Untersuchung zur Optimierung der Luftverteilung in Kanalnetzen hinsichtlich des Energieaufwands

Verfasser:
Forschungsgesellschaft
Heizung-Lüftung-Klimatechnik Stuttgart mbH

Lukas Siebler, M.Sc.
Dipl.-Ing. Armin Ruppert

Forschungsvorhaben 2020-01
gefördert vom
Verein der Förderer der Forschung im Bereich
Heizung-Lüftung-Klimatechnik Stuttgart e.V.

Verein der Förderer
der Forschung im Bereich
Heizung • Lüftung • Klimatechnik
Stuttgart e.V.

HLK
STUTTGART

April 2020

Vorwort und Danksagung

An dieser Stelle möchten wir uns recht herzlich beim Verein der Förderer der Forschung im Bereich Heizung-Lüftung-Klimatechnik Stuttgart e.V. für die Förderung des Forschungsvorhabens bedanken. In diesem Rahmen ist eine studentische Arbeit [**Lott**] betreut worden, deren Ergebnisse in den vorliegenden Endbericht eingeflossen sind.

Kurzfassung

Die Energieeffizienz der Luftverteilung (hier Konstant-VolumenstromSystem) kann im Wesentlichen durch eine Auslegung und einen Betrieb der Anlagenkomponenten nach Bedarf gesteigert werden. Kann der Bedarf jedoch nicht nur durch einen, sondern durch die Kombination mehrerer Ventilatoren an verschiedenen Positionen im Kanalnetz gedeckt werden, entsteht eine große Anzahl an Möglichkeiten, die Auslegung und den Betrieb energetisch optimal umzusetzen.

Die vorliegende Arbeit formuliert diese Thematik mit einem diskret-kombinatorischen Optimierungsproblem, bei welchem die Anzahl an Ventilatoren, deren Positionierung, Dimensionierung und Drehzahleinstellung variable Parameter sind. Zur Lösung des Problems werden drei verschiedene Methoden - die Enumeration, die sogenannte Faktorisierung und die heuristische Wahl der Kombinatorik (genetische Algorithmen) - angewandt und anhand der numerischen Berechnungssoftware MATLAB formal beschrieben.

Ziel der Methoden ist es, eine geeignete Kombination der Parameter zu finden, welche zu einem theoretisch minimalen Leistungsbedarf führt, um somit den vereinfachten und pauschalisierten Entscheidungsprozess bei der Anlagenplanung zu verbessern.

Die Eignung der Methoden wird für Probleme nachgewiesen, welche kleine Kanalnetzgeometrien und einen begrenzten Zahlenraum der Parameter aufweisen. Es werden außerdem formale Schemata erläutert, welche eine Erweiterung dieser Größen ermöglicht. Ein Auslegungsbeispiel zeigt, dass eine dezentrale Ventilator-Anordnung, gekoppelt mit der seriellen Verschaltung zweier Ventilatoren zu einer Energieeinsparung von etwa 20 % gegenüber einer zentralen Ventilator-Anordnung mit nur einem Ventilatoren führen kann. Eine Betrachtung aller kombinatorischen Zusammensetzungen ist daher notwendig, um das globale energetische Minimum von Luftverteilungen zu finden.

Abstract

The energy efficiency of the air distribution (here constant volume flow system) can essentially be increased by designing and operating the system components as required. If, however, the demand can be met not only by one but also by a combination of several fans at different positions in the duct network, a large number of possibilities will arise to optimally implement the design and operation in terms of energy efficiency.

The present thesis formulates this topic with a discrete-combinatorial optimization problem, where the number of fans, their positioning, dimensioning and rotational speed setting are variable parameters. To solve the problem, three different methods - enumeration, so-called factorization and heuristic selection of combinatorics (genetic algorithms) - are applied and formally described using the numerical calculation software MATLAB.

The aim of the methods is to find a suitable combination of parameters that leads to a theoretically minimum power requirement, in order to improve the simplified and generalised decision-making process in plant planning.

The suitability of the methods is proven for problems describing small duct network geometries

and a limited number space of parameters. In addition, formal schemes are explained which allow a scaling of these magnitudes. A design example shows that a decentralised fan arrangement coupled with the serial connection of two fans can lead to an energy saving of about 20% compared to a central fan arrangement with only one fan. A consideration of all combinatorial compositions is therefore necessary to find the global energetic minimum of air distributions.

Inhaltsverzeichnis

Bildverzeichnis

Tabellenverzeichnis

Abkürzungen

i	Strangnummer
j	Komponentennummer
$numDamp$	Klappenanzahl
$numDuct$	Kanalstückanzahl
$numEl$	Komponentenanzahl
$numFan$	Ventilatoranzahl
$numMesh$	Stranganzahl
ID	Identität der Komponente
interp2	2 dimensionale Interpolationsfunktion
K	Klappe
KVS	Konstant-Volumenstrom-System
NaN	Not a number
NEOS	Network-Enabled Optimization System
NP	nicht polynomiale Laufzeit - Komplexitätsklasse des Optimierungsproblems
P	polynomiale Laufzeit - Komplexitätsklasse des Optimierungsproblems
R	Kanalstück
RAM	Random Access Memory
V	Ventilator
VVS	Variabel-Volumenstrom-System
VZR	Typenbezeichnung des Ventilators

Formelzeichen

Symbol	Beschreibung	Einheit
k	Teilmenge ausgewählter Elemente	-
n	Menge aller Elemente; Drehzahl; Luftwechsel	-; %; h^{-1}
p	Atmosphärischer Druck	Pa
Δp_t	Gesamter Druckaufbau im Ventilator	Pa
Δp_v	Gesamter Druckabfall im Strang	Pa oder $\mathrm{N\,m^{-2}}$
Δp_{st}	Statischer Druckaufbau im Ventilator	Pa
Δp_d	Dynamischer Druckaufbau im Ventilator	Pa
w_m	Mittlere Strömungsgeschwindigkeit	$\mathrm{m\,s^{-1}}$
ρ_L	Dichte der Luft	$\mathrm{kg\,m^{-3}}$
g	Gravitationskonstante	$\mathrm{m\,s^{-2}}$
z	Geodätische Höhe	m
λ	Widerstandsbeiwert der ausgebildeten Kanalströmung	-
ζ	Widerstandsbeiwert von Einbaukomponenten	-
l	Kanallänge	m
d_h	Hydraulischer Kanaldurchmesser	m
a, b	Kanallänge Rechteckkanal; beliebige Variable	m
Q	Luftvolumenstrom	$\mathrm{m^3\,s^{-1}}$
A	Querschnittsfläche	$\mathrm{m^2}$
η	Gesamtwirkungsgrad	%
η_V	Wirkungsgrad des Ventilators	%
η_A	Wirkungsgrad des Antriebs	%
η_M	Wirkungsgrad des Motors	%

V_{NE}	Luftvolumen einer Nutzereinheit	m^3
i	Strangnummer	-
P_{el}	Elektrischer Leistungsbedarf	W
P_{fan}	Elektrischer Leistungsbedarf Ventilator	W
P	Elektrischer Leistungsbedarf Ventilatorkombination	W
W_V	Elektrischer Energiebedarf	J
W	Elektrischer Energiebedarf Luftverteilung	J
t_B	Betriebszeit	s
tp	Ventilatortyp	
$numFan$	Maximale Anzahl an Ventilatoren im Luftverteilsystem	
pos	Ventilator-Position, Angabe in Tupel	-
α	Klappen-Öffnungswinkel	$^\circ$
$numn$	Anzahl möglicher Drehzahleinstellungen	-
$numtp$	Anzahl möglicher Ventilator-Typen	-
$numFanPos$	Anzahl möglicher Ventilator-Positionen	-
π	Parameter-Permutation, Anordnung	-
$num\pi_D$	Anzahl Drehzahl-Variationen	-
$num\pi_T$	Anzahl Ventilator-Typ-Variationen	-
$num\pi_P$	Anzahl Positions-Tupel-Kombinationen	-
$num\pi_{Druck}$	Anzahl Druck-Variationen	-
$Q_{V,ges,FL}$	Für den Feuchteschutz erforderlicher Luftvolumenstrom	$m^3\,s^{-1}$
$Q_{V,Inf,Konzept}$	Luftvolumenstrom durch Infiltration	$m^3\,s^{-1}$
Δq	Diskretisierungsbreite	$m^3\,s^{-1}$
Δp_d	Diskretisierungshöhe	Pa
WP_s	Energieeinsparpotential	%

1 Einleitung und Motivation

Laut aktuellen Energieprognosen der Internationalen Energieagentur IAE wird der weltweite Energiebedarf bis 2040 um über 25% steigen. Verursacht wird der starke Anstieg hauptsächlich durch die rasant wachsende Weltbevölkerung in Entwicklungsländern und deren Urbanisierung. Mit einem etwa doppelt so hohen Anstieg wird jedoch gerechnet, wenn die Energieeffizienz in Schwellen- und Industrieländern nicht wie geplant kontinuierlich verbessert wird [**InternationalEnergyAgency**].

Die im November 2016 festgelegten Klimaziele Deutschlands verfolgen bis 2050 eine weitgehende Dekarbonisierung. Das bedeutet, dass die nationalen Treibhausgas-Emissionen um 90% bis 95% gesenkt werden sollen. Verschiedene Szenarien zeigen, dass dieses Ziel ohne Steigerung der Energieeffizienz und entsprechende Technologien sowohl in Deutschland als auch weltweit keinesfalls zu erreichen ist [**InstitutfurEnergieundUmweltforschungHeidelberg**].

Zahlreichen Studien zufolge ist das Raumklima in Büro- und Wohngebäuden für Leistungsfähigkeit und Wohlbefinden der Nutzer ausschlaggebend. Weiterhin wird die Luftdichtheit der Gebäude und die Anforderung an eine automatisierte Regelung der Raumluftfeuchte mit dem stetig steigenden Wärmedämmstandard erhöht. Die Folge ist ein steigender Einsatz von maschinellen Lüftungsanlagen [**ArbeitskreisLuftungamUmweltbundesamt**], [**LiangZhou**]. In Deutschland entstehen 35% des Endenergieverbrauches in Gebäuden und ein nicht unerheblicher Teil des dort vorliegenden Stromverbrauchs wird für Lüftungsanlagen benötigt [**BundesministeriumfurWirtschaf** [**MarkusSironi**].

Energieeffizienz steht daher auch bei der Entwicklung von Lüftungsanlagen zunehmend im Fokus. Da die Energieoptimierung einzelner Lüftungskomponenten oft nicht zur Energieoptimierung der Lüftungsanlage führt, sollte das gesamte System betrachtet werden [**C.SCHANZLEL**]. Einer der energieintensivsten Prozesse einer Lüftungsanlage ist, neben der Luftkonditionierung, der Lufttransport [**MarkusSironi**]. Insbesondere in großen Gebäuden wächst der Energiebedarf des Lufttransportes in Luftverteilsystemen mit umfangreicherer Kanalgeometrie.

Bei der Planung von Lüftungsanlagen geben geltende Richtlinien und Normen, abhängig von der Raumnutzung, notwendige Hygiene- und Behaglichkeitsanforderungen, wie den Luftwechsel vor [**DIN16798**], [**DIN1946**]. Der erforderliche Luftwechsel wird durch den zu- und abgeführten Luftvolumenstrom realisiert.

Die Gebäudegeometrie und architektonische Anforderungen bestimmen den Umfang der Kanalnetzgeometrie. Kanalnetzabschnitte mit Form- und Verbindungsstücken bilden die Kanalgeometrie, die bei Durchströmung einen Druckabfall verursacht. Andere Komponenten wie Klappen sorgen für einen weiteren Druckabfall. Ventilatoren kompensieren mit einem Druckanstieg den Druckabfall im Kanalnetz und transportieren die Luft dadurch in die gewünschte Richtung. Für den Druckaufbau benötigt der Ventilator elektrische Leistung.

Während der Planung einer Lüftungsanlage liegen trotz vordefiniertem Luftvolumenstrom (Konstant-Volumenstrom-System KVS) und gegebener Kanalnetzgeometrie zahlreiche Möglichkeiten vor, die Luftverteilung zu realisieren. Über die Anzahl, die Positionierung, die Dimensionierung und die Drehzahleinstellung der Ventilatoren sowie weitere Freiheitsgrade kann frei entschieden werden. Jede einzelne Entscheidung kann den Leistungsbedarf des Luftverteilsystems we-

sentlich beeinflussen. Der Entscheidungsprozess des Planers wird durch die Anzahl kombinatorischer Zusammensetzungen der Parameter bereits für sehr kleine Luftverteilsysteme sehr komplex. In der Praxis entstehen daher durch Zeit- und Informationsmangel oft Lösungen, die der Aufgabenstellung der Lüftungsanlage nicht genügen, oder zu überdimensionierten Anlagen führen [**AerotechnikE.SigwartGmbH**], [**STIEBELELTRONGmbH**]. Motiviert ist diese Arbeit durch das Ziel, Planern einen einfachen Entscheidungsprozess bei theoretisch minimalem Leistungsbedarf der Luftverteilung zu ermöglichen.

2 Handlungsbedarf

Im Folgenden wird der Stand der Technik im Planungsprozess von Lüftungsanlagen erläutert und durch den Stand der Forschung zur Optimierung der Luftverteilung erweitert.

Dieses Kapitel beschreibt durch einen Vergleich der bestehenden Optimierungsansätze die Notwendigkeit für diese Arbeit.

2.1 Stand der Technik - Planung von Lüftungsanlagen

Das Planen von Lüftungsanlagen erfolgt in der Regel durch Ingenieur- oder Planungsbüros und ausführende Unternehmen. Wird die Lüftungsanlage für ein Wohn- oder Nicht-Wohngebäude vom Bauherren in Auftrag gegeben, sind zahlreiche Schritte zu beachten [**AerotechnikE.SigwartGmbH**] [**STIEBELELTRONGmbH**].

Seit Januar 2018 muss im Gebäudebereich, nach aktuellen Normen und Richtlinien, sowohl für Neubauten als auch für Sanierungen mit lüftungstechnisch relevanten Änderungen, zunächst ein Lüftungskonzept erstellt werden. Mit diesem soll sichergestellt werden, dass nutzerunabhängig ein Feuchteschutz zur Schimmelpilzvermeidung erfolgt. So werden lüftungstechnische Maßnahmen nach DIN 1946-6 spätestens erforderlich, sobald der für den Feuchteschutz erforderliche Luftvolumenstrom, den durch Infiltration überschreitet [**DIN1946**].

$$Q_{V,ges,FL} > Q_{V,Inf,Konzept} \tag{2-1}$$

$Q_{V,ges,FL}$ Erforderlicher Luftvolumenstrom für den Feuchteschutz in $\mathrm{m^3\,s^{-1}}$

$Q_{V,Inf,Konzept}$ Luftvolumenstrom durch Infiltration in $\mathrm{m^3\,s^{-1}}$

Neben dieser ausschlaggebenden bauphysikalischen Anforderung werden, wie in Kapitel 3.2 erläutert, abhängig von Schadstoffquellen, Nutzung und Dichtheit der Gebäude, weitere Hygiene- und Behaglichkeitsanforderungen nach *Fanger* berücksichtigt [**Maas**]. Der erforderliche Luftvolumenstrom wird daran angepasst. DIN EN 16798-7:2017 stellt Berechnungsmethoden zur Bestimmung der Luftvolumenströme in Gebäuden einschließlich Infiltration zur Verfügung. Dabei ist für Nichtwohngebäude beispielsweise der Mindestluftwechsel nach DIN EN 15251:2012 (Tabellen B.1 und B.2) einzuhalten [**DIN16798**] [**DIN15251**].

Im nächsten Schritt wird entsprechend dieses erforderlichen Luftvolumenstroms ein Lüftungssystem ausgewählt. Dabei kann zwischen einer freien (Fensterlüftung und Infiltration), einer ventilatorgestützten und einer kombinierten Lüftung gewählt werden. Jedes der Lüftungssysteme bietet weitere Umsetzungsmöglichkeiten, die in Bild Bild 2-1 dargestellt sind. Speziell erhöhte Anforderungen an die Energieeffizienz der ventilatorgestützten Lüftungssysteme werden durch die ErP Richtlinie empfohlen, sind jedoch nicht verpflichtend umzusetzen. DIN EN 16798-3:2015 legt weiterhin Anforderungen an die Leistung von Lüftungs- und Klimaanlagen und Raumkühlsystemen fest. Nach DIN EN 18599-3:2018 erfolgt die Berechnung des Nutzenergiebedarfs für die energetische Luftaufbereitung [**DIN18599**],[**EuropaischenUnion**].

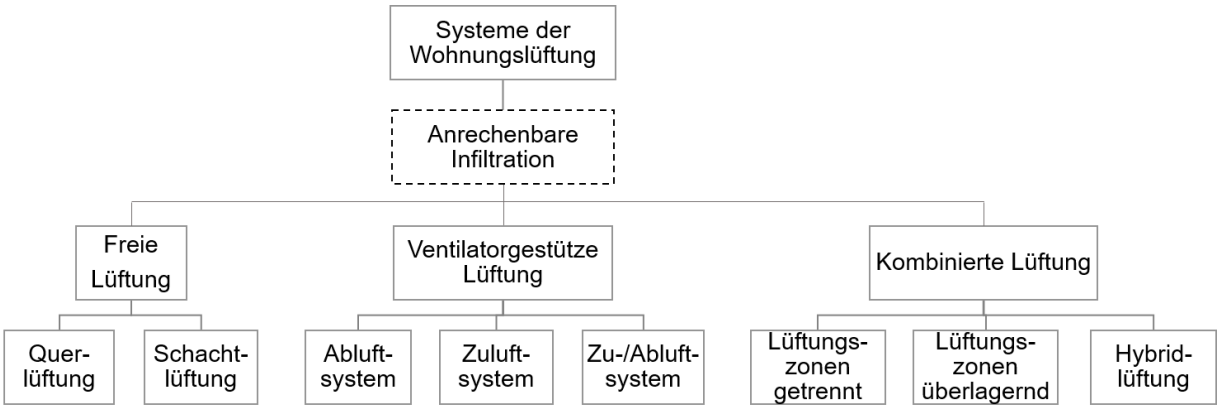


Bild: Bild 2-1 Umsetzungsmöglichkeiten der Lüftungssysteme nach DIN 1946-6 [DIN1946]

Wird eine ventilatorgestützte Lüftung mit vorgegebenen Sollvolumenströmen eingesetzt, wird der Ventilator für den größten erforderlichen Volumenstrom und den Strang mit dem maximalen Druckabfall ausgelegt. Neben Normen und Richtlinien bieten Lüftungsanlagenhersteller Vorgaben für die Auslegung. Einige Praxisleitfäden führen durch den Planungsprozess, der ähnlich wie in Bild Bild 2-2 dargestellt abläuft [**AerotechnikE.SigwartGmbH**] [**STIEBELELTRONGmbH**]. Sie stellen Produktkataloge und Produktkennfelder sowie Auslegungssoftware zur Verfügung, welche beispielsweise die Außenluftvolumenströme pro Nutzungseinheit berechnen und nach der Auswahl der Lüftungskomponenten entsprechende Materiallisten nach DIN 1946-6 [**DIN1946**] ausgeben.

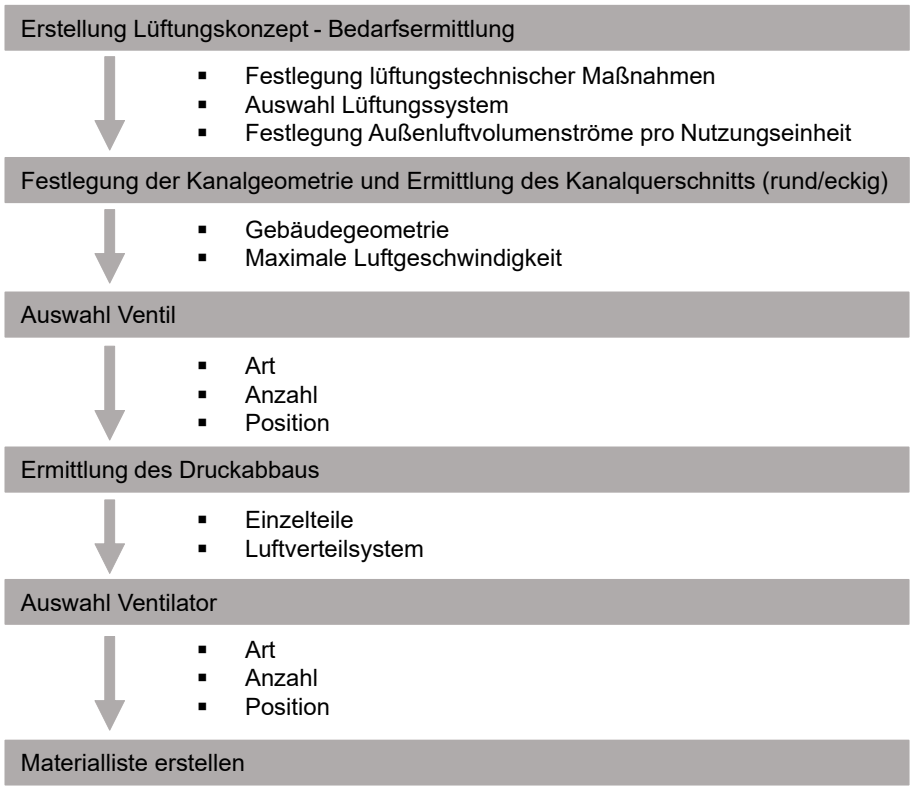


Bild: Bild 2-2 Flussdiagramm zur Auslegung lufttechnischer Anlagen

In dieser Arbeit wird davon ausgegangen, dass lüftungstechnische Maßnahmen erforderlich

sind und eine ventilatorgestützte Lüftung zum Einsatz kommt. Der Planer muss im Vorfeld, unter Berücksichtigung aller Bedingungen und Anforderungen, solange abwägen, bis eine geeignete Topologie des Luftverteilsystems gefunden ist, welche zur Erfüllung des Lufttransportes die minimale Leistung benötigt. Aufgrund der Vielzahl an kombinatorischen Zusammensetzungen und meist vorhandenem Zeitdruck werden Entscheidungen oft auf Basis von Erfahrungswerten getroffen. Dadurch ist die Wahrscheinlichkeit hoch, dass jeder Planer individuell entscheidet und dabei nicht die optimale Systemlösung gefunden wird. Werden Computer und intelligente Algorithmen zu Hilfe genommen, kann die optimale Systemlösung mit einer weitaus höheren Wahrscheinlichkeit gefunden werden [**C.SCHANZLEL**], [**SandyJorensIvanVerhaertKennethSorensen**].

2.2 Stand der Forschung - Optimierung der Luftverteilung

Betriebsweisen und Regelstrategien zentraler Ventilatoren werden bereits 2005 im Forschungsvorhaben BOLKA-II untersucht. Dabei wird unter anderem der Einsatz von Konstant-Volumenstrom-Systemen (KVS) und Variablen-Volumen-strom-Systemen (VVS) betrachtet [**TechnischeUniversitatDresden**] Aufgrund der variierenden Anforderungen an die Luftqualität in Abhängigkeit der Personenanzahl und Zonennutzung (siehe Kapitel 3.2 und 2.1) wird der erforderliche Volumenstrom in VVS daran angepasst geregelt. Ziel dieser Systeme ist es, durch eine geringere Betriebszeit, als die der KVS, Energie einzusparen.

Es existieren weiterhin verschiedene Ansätze, die Energieeffizienz von Lüftungsanlagen zu optimieren:

Klimmt und Arlsen [**TobiasKlimmt**] [**Niklas**] setzen an der Wahl einer Kombination zwischen zentralen und dezentralen Ventilatoren innerhalb einer Luftverteilung an (schematisch siehe Bild 2-3). Die Bewertung dieses Systems findet sowohl energetisch als auch wirtschaftlich statt. Es wird mit Hilfe einer physikalischen Simulationsumgebung und Messungen im Realmaßstab dargestellt, wie die Positionierung und Anzahl der Ventilatoren Einfluss auf die Anzahl der Klappen, hier Volumenstromregler, hat. Diese sorgen dafür, einen eingestellten Volumenstrom mit Hilfe einer Regelklappe konstant zu halten. Die Regelung beinhaltet eine Veränderung des Klappenstellwinkels, wodurch sich der Druckabfall in der Klappe verändert. Für verschiedene erforderliche Strang-Volumenströme müssen die Volumenstromregler die Differenz der Druckabfälle zwischen den Strängen ausgleichen. Erfolgt ein geeignet gewählter Einsatz der kombinierten zentralen und dezentralen Ventilatoren kann das System mit weniger Volumenstromreglern oder wie in Bild 2-3 dargestellt ohne Volumenstromregler ausgeführt werden. Im gesamten System muss dadurch ein geringerer Druckabfall kompensiert werden. Die Leistung, die durch Druckabfälle in den zusätzlichen Volumenstromreglern im System mit einem zentralen Ventilator hätte aufgewendet werden müssen, wird eingespart. Untersucht werden Nichtwohngebäude mit bedarfsgerechter Lüftung. Die dezentralen Ventilatoren werden für diesen Ansatz bereits im Voraus festgelegt, weshalb mit der Modellierung und Simulation des Systems keine Aussage über deren optimale Anzahl und Position getroffen werden kann.

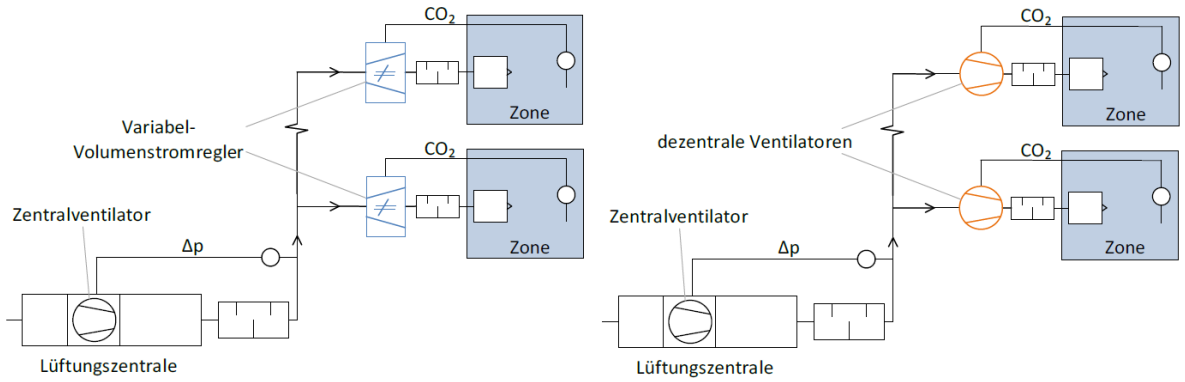


Bild: Bild 2-3 Schematische Darstellung zweier Luftverteilungen mit und ohne dezentrale Ventilatoren nach Klimmt [TobiasKlimmt]

Zahlreiche Arbeiten befassen sich mit der kombinatorischen Optimierung von Luftverteilungen durch computergestützte Programme und Algorithmen. Der Fokus ist dabei immer unterschiedlich gesetzt. Mossolly [**MOSSOLLY.2009**] entwickelt mit Hilfe eines Genetischen Algorithmus (GA) Luftauslassstrategien der Luftverteilung in Abhängigkeit der Raumluftqualität.

Taecheol [**TaecheolKim.2001**] entwickelt wirtschaftlich optimale Netzgeometrien für VVS anhand verschiedener Optimierungsalgorithmen. Anhand dieser wird hier ein Ventilator gewählt, welcher nicht für einen maximalen Volumenstrom, sondern für den VVS-Betrieb (Teillast) optimiert ist. Dabei wird die Anzahl oder Position des Ventilators nicht variiert.

Eine durch die American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) anerkannte Auslegungsmethode für Luftverteilungen ist die sogenannte T-Methode - eine kombinatorische Optimierung mit mehreren Zielfunktionen. Optimiert wird hier sowohl hinsichtlich der Druckverteilung im Luftverteilsystem, als auch hinsichtlich minimaler Lebenszykluskosten. Auf Basis dieser Zielfunktionen werden optimale Kanal- und Ventilator-Größen ausgegeben. Dabei wird das gesamte Jahr über von einem konstanten, maximal benötigten Volumenstrom ausgegangen. Die Kennfelder der gewählten Ventilatoren können dabei keine Teillast abbilden. Die Wirkungsgrade bei verschiedenen Teillasten werden mit dieser Methode nicht berücksichtigt. Außerdem sind die Positionen der Komponenten, genau wie in anderen Auslegungsmethoden, in der Regel bereits im Voraus festgelegt [**TaecheolKimJeffreyD.Spitler**].

Eine ähnliche Betrachtung erfolgt durch Jorens [**SandyJorensIvanVerhaertKennethSorensen**]. Der Fokus liegt hier auf der Optimierung der Netzgeometrie. Dabei wird einem GA [**Jorens.2017**] neben der Dimensionierung des Luftverteilsystems die Position und Anzahl der Form- und Verbindungsstücke in großen Luftverteilsystemen optimiert. Nach aktuellem Forschungsstand wird jedoch eine lokale Suche (ein anderer heuristischer Algorithmus) angewendet. Die Position und Anzahl der Ventilatoren bleibt in den beschriebenen Beispielmodellen jedoch konstant.

Schänzle [**C.SCHANZLEL**] betrachtet die energetische Optimierung einer Luftverteilung. Diese wird als VVS mit drei quasistationären Volumenströmen betrieben. Dabei wird die Netzgeometrie im Vorfeld festgelegt, sowohl die Position (innerhalb einer Lüftungszentrale) und Anzahl der Ventilatoren als auch deren Dimensionierung und Arbeitsbereich werden variiert. Die Abbildung dieser Größen erfolgt näherungsweise mit Hilfe von Affinitäts- und Skalierungsgesetzen, womit eine lineare Zielfunktion formuliert wird. Mit Hilfe eines intelligenten Algorithmus des Bereichs Operation Research wird das gemischt-ganzzahlige kombinatorische lineare Optimierungsproblem gelöst. Eine Erläuterung des Optimierungsproblems und der zugehörigen Begriffe erfolgt

in Kapitel 4.1. Ermittelt wird das globale Optimum. Der Fokus liegt dabei auf der Effizienzsteigerung durch eine zentrale parallele, serielle oder gemischte Verschaltung verschiedener Ventilatoren in einer Lüftungszentrale (siehe Bild Bild 2-4). Die Nutzung dezentraler Ventilatoren in Luftverteilsystemen wird nicht berücksichtigt.

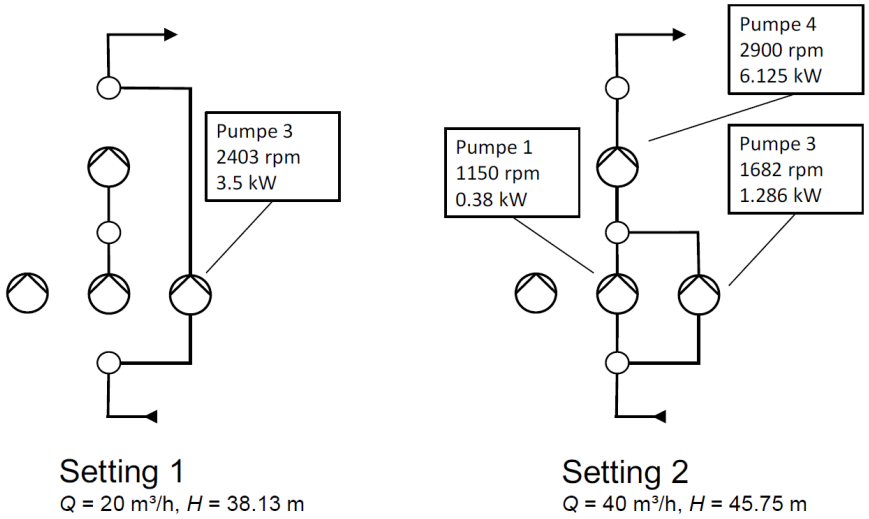


Bild: Bild 2-4 Schematische Darstellung zweier Lösungen des energetischen Optimierungsproblems nach Schänzle [C.SCHANZLEL]

Die Dimensionierung einer energieeffizienten und/oder kosteneffizienten Luftverteilung kann also mit einem breiten Spektrum an Optimierungsmethoden erfolgen. Grundlage für jede dieser Methoden ist es, den Druckabfall im gesamten System zu reduzieren. Einige der Arbeiten berücksichtigen dabei die Kombination von dezentralen und zentralen Ventilatoren (hybride Ventilatoranordnung), andere die Auswahl von Anzahl, Position, Dimensionierung und Arbeitsbereich der Komponenten.

Mit jeder dieser Arbeiten kann eine mehr oder weniger große Energie- oder Kosteneinsparung nachgewiesen werden. Im Rahmen dieser Arbeit soll daher ein kombinatorisches Optimierungsproblem, wie es in Kapitel 4.1.1 und 4.1.2, beschrieben wird, behandelt werden. Der Entscheidungsspielraum des Problems soll dabei gleichzeitig dezentrale Ventilatoren in zentralen Luftverteilsystemen und eine dezentrale parallele, serielle oder gemischte Verschaltung dieser zulassen. Es sollen Optimierungsmethoden entwickelt und verglichen werden, mit welchen eine Aussage darüber getroffen werden kann, für welche Ausgangssituation (Volumenströme, Netzgeometrien und zur Verfügung stehende Komponenten) eine solche Kombination eine Energieeinsparung bringt. Die Methoden sollen weiterhin aufzeigen, wie genau diese Kombination aufgebaut sein soll, um eine maximale Energieeinsparung zu generieren. Dabei sollen die gängigen Planungsschritte und der dabei bestehende Zeitdruck berücksichtigt werden.

3 Grundlagen

In diesem Kapitel werden Begriffe definiert und physikalische Zusammenhänge erläutert, welche Basis für das Verständnis der Beschreibung von Optimierungsproblem und -methode sind. Dabei wird zunächst auf allgemeine Begriffe der mathematischen Optimierung und Kombinatorik eingegangen. Im Anschluss werden Gleichungen und Kennlinien der Luftverteilung betrachtet.

3.1 Begriffe der mathematischen Optimierung und Kombinatorik

Die Optimierung ist ein wichtiges Teilgebiet der angewandten Mathematik, welches in verschiedenen Anwendungen in Wirtschafts-, Ingenieur- und Naturwissenschaften stetig an Bedeutung gewinnt. Die Bestimmung eines größten oder kleinsten Wertes einer Zielfunktion ist die Aufgabe eines Optimierungsproblems [**MartinPieper**]. Weitere, in dieser Arbeit verwendete Begriffe der mathematischen Optimierung, werden im Folgenden kurz aufgegriffen.

Die Mengenlehre beschreibt eine Grundmenge G als die Zusammenfassung von bestimmten, unterschiedlichen Elementen zu einem Ganzen. Dabei wird beispielsweise ein möglicher Zahlenraum eines Parameters beschrieben. Die mathematische Schreibweise für ein Element x - hier Parameter - welches in der Grundmenge enthalten ist, lautet $x \in G$ bzw. $G = \{x\}$. Eine Lösungsmenge oder zulässige Menge L ist dagegen eine Zusammenfassung von bestimmten, unterschiedlichen Elementen, welche zu einer gültigen Lösung eines Problems führen. Damit wird eine Teilmenge der Grundmenge beschrieben. Die Schreibweise ist analog $L = \{x\}$ [**OliverVornbergerOlafMuller**].

Ein Zielfunktionswert f_{min} ist eine gültige Lösung, wenn der eingesetzte Parameter in der Lösungsmenge ($x \in L$) enthalten ist und alle Bedingungen eingehalten werden.

Ein Optimierungsproblem ist die Definition einer Instanz und einer Aufgabe. Die Instanz setzt sich zusammen aus einer Zielfunktion und einer Lösungsmenge. Die Aufgabe definiert, wann die Zielfunktion optimal ist. Mathematisch kann die Instanz mit Zielfunktionswert in Abhängigkeit des Parameters x mit der Zielfunktion $f(x)$ und der Bedingungen $x \in L$ formuliert werden. Dabei ist die Zielfunktion $f(x)$ die mathematische Gleichung zur Berechnung des Zielfunktionswertes, welcher den zu optimierenden Wert darstellt. Eine Bedingung sind Gleichungen ($3 = x^2 * 8$), Ungleichungen ($x > 0$) oder Zahlenräume ($x \in \mathbb{N}$), welche die Grund- und Lösungsmengen begrenzen. Liegt die Lösung eines Problems genau auf der Bedingung, wird sie als Randbedingung bezeichnet. Ansonsten handelt es sich um eine Nebenbedingung. Die Aufgabe der Optimierung ist es beispielsweise, den Zielfunktionswert durch Parameteränderung zu minimieren ($min(f(x))$) und gleichzeitig die Bedingungen einzuhalten [**BernhardKorte**].

Ein Optimierungsproblem wird nach seinen Eigenschaften benannt. Es kann linear oder nichtlinear sein. Die Zuordnung erfolgt anhand der Ziel- und Bedingungsfunktion. Ein nicht-lineares Problem wird daher mit nicht-linearen Ziel- und Bedingungsfunktionen (z.B. $f(x) = x^2$ und $x^3 + x = 18$) beschrieben. Wird der Anspruch an die Lösung betrachtet, wird zwischen globalem und lokalem Minimum unterschieden. Für eine Zielfunktion höheren Grades können lokale (relative) Minima auftreten, welche nicht dem globalen (absoluten) Minimum entsprechen. Wird die Lösungsmenge des Parameters begrenzt, ändert sich das Problem von kon-

tinuierlich ($x \in \mathbb{R}$) zu diskret (Beispielsweise ganzzahlig $x \in \mathbb{N}$ oder $x = \{0, 10, 20, 30\}$). Für Letzteres ist die Lösungsmenge betrachteter Parameter endlich oder abzählbar unendlich. Alle kombinatorischen Probleme haben endliche Lösungsmengen [**TUBergakademieFreiburg**] [**ChristophHelmberg**].

Bild Bild 3-1 dient zum Verständnis der Begriffe. Auf der linken Seite bildet die schwarze Kurve eine nicht-lineare kontinuierliche Funktion ab. Die endliche Lösungsmenge wird durch Kreuze gekennzeichnet. Beide - Neben- und Randbedingung - sind Ungleichungen, welche durch blaue Geraden abgebildet werden. Die Aufgabe, das Minimum des Problems zu finden, ist ausschließlich bei dem Parameterwert $x = 0$ erfüllt. Auf der rechten Seite in Bild Bild 3-1 wird sowohl eine gestrichelte Gerade als auch eine schwarze Kurve abgebildet. Erstere wird als lineare Zielfunktion verstanden, letztere stellt eine Funktion vierter Ordnung dar. Hier kann ein lokales und ein globales Minimum unterschieden werden.

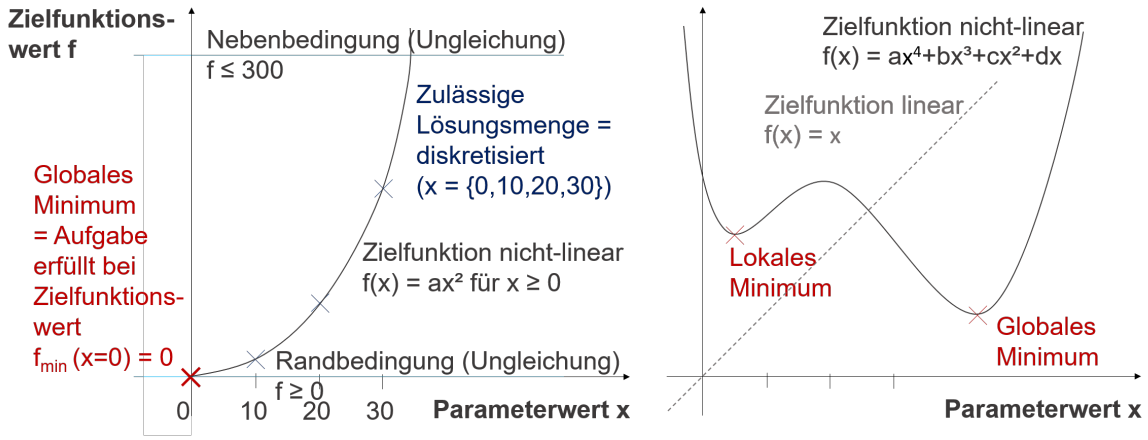


Bild: Bild 3-1 Graphische Darstellung eines diskreten nicht-linearen Optimierungsproblems (links). Vergleich globales und lokales Minimum sowie lineare und nicht-lineare Zielfunktion (rechts).

Die Kombinatorik bearbeitet das Grundproblem des Abzählens von Elementen einer gegebenen endlichen Menge. Ein kombinatorisches Element ist dabei entweder eine Anordnung (Permutation), eine Auswahl (Kombination, Variation), eine Verteilung oder eine Zerlegung (Partitionen). Das explizite Auflisten und Zählen der kombinatorischen Elemente der endlichen Menge - die Enumeration - liefert eine Lösung des Grundproblems. Wird eine große endliche Menge betrachtet, wird das Zählen der kombinatorischen Elemente sehr zeitaufwendig, weshalb die Anzahl der Elemente mit Hilfe von speziellen Folgen ganzer Zahlen berechnet werden (beispielsweise Binomialkoeffizienten Vgl. Bild 3-2). Bild Bild 3-2 veranschaulicht die Unterschiede der kombinatorischen Elemente Kombination und Variation. Diese sind gegeben, wenn nur eine Teilmenge k aller Elemente n einer endlichen Menge in einer beliebigen Anordnung π verwendet werden. Für beide wird unterschieden, ob die Elemente wiederholt ausgewählt werden können (mit Zurücklegen) oder nicht. Die Variation unterscheidet sich von der Kombination durch die Relevanz der Reihenfolge der ausgewählten Elemente [**FernUniversitatinHagen**], [**Tittmann**]. In dieser Arbeit kommen die Kombination mit Wiederholung und die Variation mit Wiederholung zum Einsatz (Gleichung 3-1 und 3-2).

Anzahl Variationen mit Wiederholung : n^k (3-1)

Anzahl Kombinationen mit Wiederholung : $\binom{\mathrm{n}+\mathrm{k}-1}{\mathrm{k}}$ (3-2)

k Einheitslose Teilmenge aller Elemente

n Einheitslose Menge aller Elemente z.B. Lösungsmenge

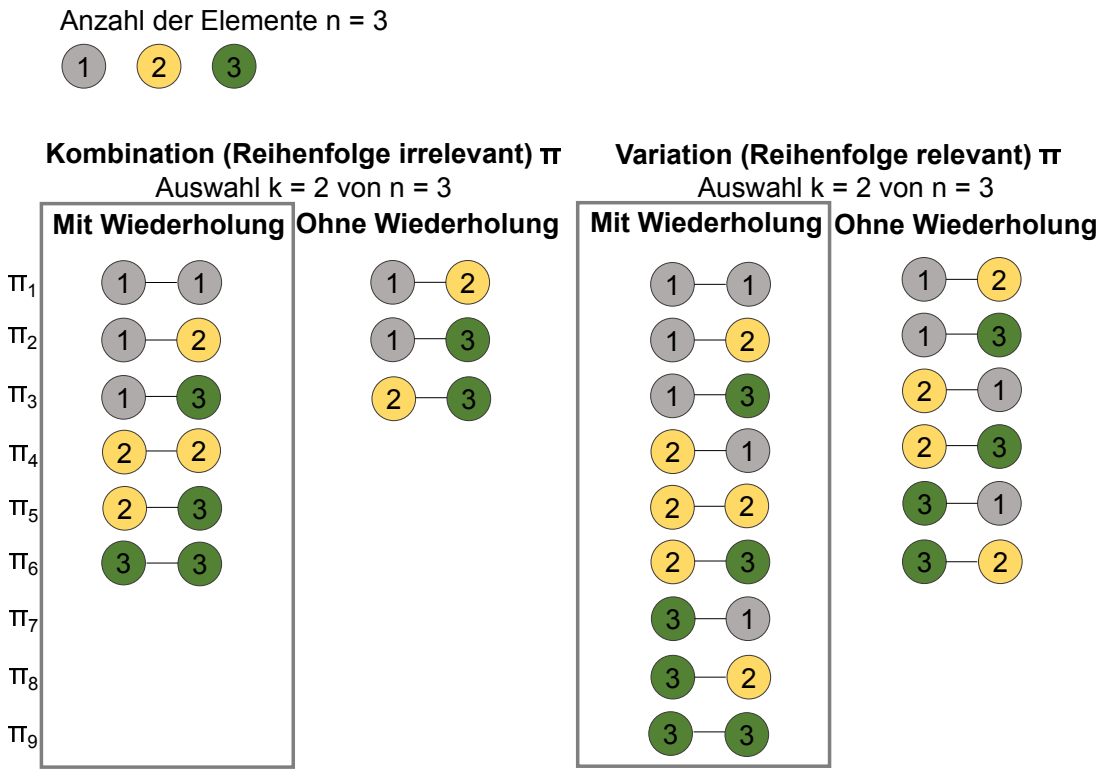


Bild: Bild 3-2 Graphische Darstellung der kombinatorischen Elemente Kombination und Variation

Wird die Aufgabe des Problems stochastisch-heuristisch gelöst, so kann der Zielfunktionswert bei einer Wiederholung des Lösungsverfahrens variieren. Dadurch kann nicht garantiert werden, dass das globale Minimum gefunden wird. Das Lösungsverfahren ist nicht bei jeder Wiederholung exakt gleich, sondern passt sich laufend verändernden Arbeitsanweisungen an und findet näherungsweise Lösungen (möglicherweise lokale Minima). Wird die Aufgabe dagegen deterministisch gelöst, bleibt das Lösungsverfahren immer exakt gleich, womit die Lösung und alle Zwischenzielfunktionswerte ebenfalls immer exakt gleich sind [**Rimscha**].

Für die Bewertung eines Optimierungsproblems wird unter anderem die Laufzeit des Lösungsverfahrens unter Berücksichtigung des Wachstums des Rechenaufwands mit der Problemgröße betrachtet. Die Laufzeit betrachtet die Anzahl der ausgeführten elementaren Rechenoperatoren (formal bedeutet dies z.B. ein Symbol lesen/schreiben, Bytes addieren/ multiplizieren/ vergleichen). Sie wird in die Komplexitätsklassen polynomial (P) und nichtdeterministisch polynomial (NP) eingeteilt. Ist ein Problem der Klasse NP zu lösen, wächst der Rechenaufwand

so schnell, dass mit heute verfügbaren Rechnern schon eine geringe Problemgröße nicht mehr in einer überschaubaren Zeit gelöst werden kann. Die Klasse NP umfasst dabei alle Entscheidungsprobleme und die meisten kombinatorischen Probleme [**ChristophHelmberg**] [**TUDresden**], [**Rimscha**].

Zur Lösung eines Optimierungsproblems der Klasse NP werden daher Algorithmen verwendet. Ein Algorithmus muss in seinen Arbeitsanweisungen sowohl allgemeingültig (auf verschiedene Probleme anwendbar) und ausführbar (endliche, eindeutige, verständliche, Anweisungen in einer definierten Reihenfolge) sein und er muss zu einem Ergebnis kommen [**Rimscha**]. In dieser Arbeit werden zwei Algorithmen (Enumeration und Genetischer Algorithmus) verwendet und auf die Problemstellung angewendet. Sie werden in Kapitel 5 erläutert.

Neben den Begriffen der mathematischen Optimierung soll an dieser Stelle außerdem der Begriff Tupel definiert werden. Ein Tupel wird in der Mathematik als eine Zusammenfassung von Elementen - ähnlich der Mengen - verstanden. Die Elemente der Tupel sind im Vergleich zu denen der Mengen nicht notwendigerweise unterschiedlich. Die Länge eines Tupels wird durch n in n-Tupel beschrieben. So hat ein 2-Tupel zwei Einträge [**BernhardKorte**], [**OliverVornbergerOlafMuller**].

3.2 Luftverteilung

An einen Ventilator einer Lüftungsanlage werden verschiedene Anforderungen gestellt. Diese beziehen sich auf die Betriebsgrößen Luftvolumenstrom und spezifische Förderarbeit. Um diese an den Bedarf anzupassen, erfolgt entweder eine Veränderung der Geometrie des Ventilators, wie Leitrad- oder Laufradverstellung, eine Änderung der Drehzahl oder eine Parallel-, Reihen- oder Bypass-Schaltung. Das Betriebsverhalten der Ventilatoren und Anlagenkomponenten wird mit Kennlinien oder Kennfeldern dargestellt. Diese geben relevante Betriebsgrößen in Tabellen oder Kurvendiagrammen an und werden meist messtechnisch durch die Komponentenhersteller bestimmt. Im Folgenden werden die physikalischen Zusammenhänge von Anlagenkennlinien (Kanalnetzkennlinien) und Ventilatorkennlinien betrachtet.

3.2.1 Kennlinien und Druckverlauf

Die beiden Kennlinien - Anlagenkennlinie und Ventilatorkennlinie - geben den Zusammenhang zwischen der erforderlichen spezifischen Förderarbeit des Ventilators und dem durch die Anlage zu fördernden Luftvolumenstrom wieder. Die thermodynamische Grundlage für diesen Zusammenhang liefert die Energieerhaltungsgleichung für strömende Fluide in durchströmten Maschinen und Bauteilen nach Bernoulli [**Weber**]. Sie gilt sowohl für inkompressible als auch für kompressible Fluide, wie Luft bis zu einer Druckdifferenz von $3{,}0\,\mathrm{kPa}$ oder einer Strömungsgeschwindigkeit von bis zu $100\,\mathrm{m\,s^{-1}}$ [**Weber**]. Bei der Förderung eines Fluids von einem Zustand 1 in einen anderen Zustand 2 über die Systemgrenze hinweg, entstehen Strömungsverluste in den Anlagenkomponenten. Ihr Ursprung liegt hauptsächlich in Dissipationsenergie, welche in Wärme übergeht. Die Verluste werden mit einem Druckabfall im Kanalnetz berücksichtigt. Um unter Berücksichtigung dieser Strömungsverluste eine Luftförderung zu ermöglichen, muss die spezifische Förderarbeit $\frac{\Delta p_t}{\rho_L}$ geleistet werden. Dies kann in Form einer Erweiterung der Bernoulli-Gleichung abgebildet werden (siehe Gleichung 3-3) [**Weber**].

$$p_1 + \Delta p_t + \frac{w_{m1}^2}{2} * \rho_L + g * z_1 * \rho_L = p_2 + \Delta p_v + \frac{w_{m2}^2}{2} * \rho_L + g * z_2 * \rho_L \tag{3-3}$$

p Atmosphärischen Druck des Zustands 1 oder 2 in Pa

ρ_L (Temperatur- und druckabhängige) Dichte der Luft in $\mathrm{kg\,m^{-3}}$

w_m Mittlere Strömungsgeschwindigkeit des Zustands 1 oder 2 in $\mathrm{m\,s^{-1}}$

g Gravitationskonstante in $\mathrm{m\,s^{-2}}$

z Geodätische Höhe über Bezugsebene des Zustands 1 oder 2 in m

Δp_v Druckabfall im Kanalnetz in Pa

Δp_t Druckaufbau durch Ventilatoren in Pa

Für offene Systeme - wie das einer Lüftungsanlage - entspricht der atmosphärische Druck in Zustand 1 dem in Zustand 2. Wird Gleichung 3-3 nach dem Druckaufbau Δp_t umgestellt, entfällt somit der Term $p_2 - p_1$. Bei einem Ventilator im Umwälzbetrieb wird außerdem die geodätische Höhendifferenz vernachlässigt [**Weber**]. Die Druckgleichung reduziert sich zu Gleichung 3-4. Allgemein kann Gleichung 3-4, wie nachfolgend erläutert, mit dem statischen Druck p_{st} und dem dynamischen Druck p_d gleichgesetzt werden. Daraus resultiert Gleichung 3-5, welche zur Auslegung von Ventilatoren verwendet wird.

$$\Delta p_t = \Delta p_v + \frac{w_{m2}^2 - w_{m1}^2}{2} * \rho_L \tag{3-4}$$

$$\Delta p_t = \Delta p_{st} + \Delta p_d \tag{3-5}$$

Δp_{st} Statischer Differenzdruck in Pa

Δp_d Dynamischer Differenzdruck in Pa

Eine graphische Darstellung (Bild Bild 3-3) des Druckverlaufes in einem Kanalnetz mit Anlagenkomponenten und Ventilator soll deren Wirkungsweise verdeutlichen. Um einen Luftvolumenstrom zu fördern, wird ersichtlich, dass ein Ventilator im stationären Betrieb den Druck aufbauen muss, der durch den Druckabfall im Kanalnetz abgebaut wird. Der Differenzdruck ist dabei die treibende Kraft.

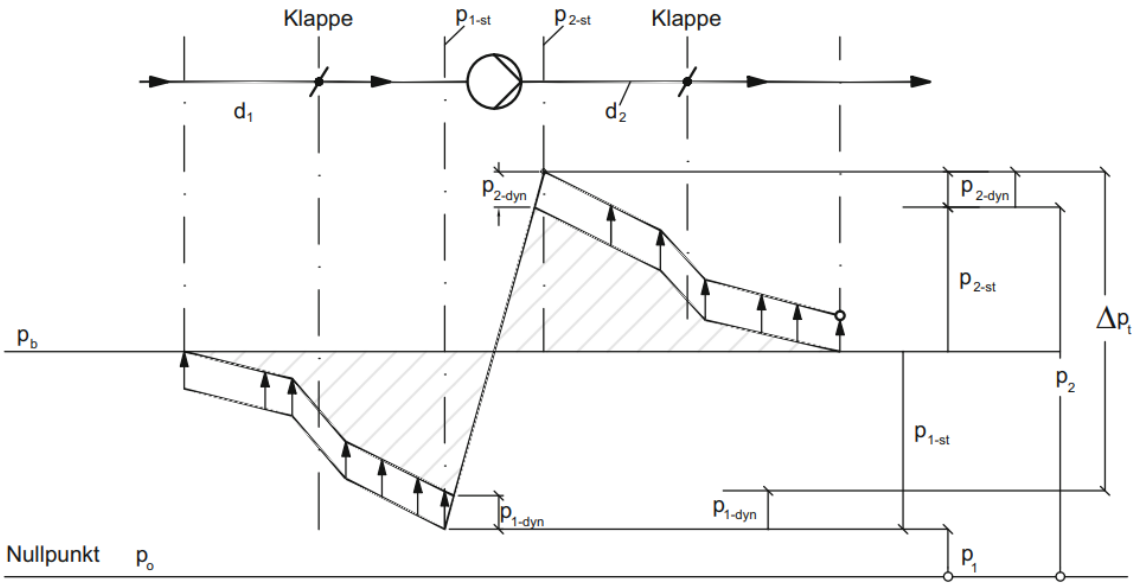


Bild: Bild 3-3 Druckverlauf eines Kanalnetzes mit Druckabfall in Kanalstücken und Klappen sowie Druckaufbau im Ventilator. Druck- und Saugseite des Ventilators werden mit statischem und dynamischem Druck dargestellt [Weber].

Der statische Differenzdruck ist der nutzbare bzw. externe Druck des Ventilators, welcher für die Kompensation des Druckabfalls im Kanalnetz zur Verfügung steht. Der dynamische Differenzdruck beschreibt die ventilatorbedingten Strömungsverluste, welche hauptsächlich durch die Einbausituation und die erforderliche Strömungsgeschwindigkeit am Luftauslass zustande kommen. Für die Auslegung eines Ventilators wird daher typischerweise ausschließlich der statische, nutzbare Differenzdruck berücksichtigt. Der dynamische Differenzdruck wird vernachlässigt und in der Berechnung der Ventilator-Antriebsleistung mit einem Wirkungsgrad η_V berücksichtigt (siehe Kapitel 3.2.2) [**Niklas**], [**Dipl.Ing.HeinzJackmann**]. Somit kann Gleichung 3-4 zu Gleichung 3-6 reduziert werden.

$$\Delta p_t = \Delta p_v \text{ oder } \Delta p_t - \Delta p_v = 0Pa \tag{3-6}$$

Bei der Berechnung des Druckabfalls im Kanalnetz Δp_v wird die Serienschaltung der Kanalstücke und Klappen mit dem Summen-Symbol berücksichtigt [**Weber**]:

$$\Delta p_v = \sum (\zeta + \lambda * \frac{l}{d_h}) * \rho_L * \frac{w_m^2}{2} \tag{3-7}$$

Δp_v Druckabfall im Kanalnetz in Pa

λ Dimensionsloser Widerstandsbeiwert der ausgebildeten Kanalströmung (Bestimmung mit Diagrammen wie Moody/Colebrook oder mit Gleichungen nach [**Boswirth**])

ζ Dimensionsloser Widerstandsbeiwert von Einbaukomponenten (Bestimmung messtechnisch durch Hersteller und Auslesen aus Tabellen)

ρ_L Temperatur- und druckabhängige Dichte der Luft in $\mathrm{kg\,m^{-3}}$

l Länge des Kanals ohne Komponenten in m

d_h Hydraulischer Kanaldurchmesser in m (Berechnung für Rechteckkanal der Kantenlänge $a * b$, $d_h = \frac{2*a*b}{a+b}$)

w_m Mittlere Strömungsgeschwindigkeit in m s^{-1}

Die mittlere Strömungsgeschwindigkeit kann durch das Verhältnis von Luftvolumenstrom Q zur durchströmten Querschnittsfläche A (Gleichung 3-8) berechnet werden. Durch Einsetzen dieses Verhältnisses in Gleichung 3-7, wird in Gleichung 3-9 ersichtlich, dass der Druckabfall - in Klappen und Kanalstücken - quadratisch mit dem Volumenstrom durch die Komponente abnimmt, der Druckaufbau - im Ventilator - dagegen quadratisch mit dem Volumenstrom steigt.

$$w_m = \frac{Q}{A} \tag{3-8}$$

$$\Delta p_v \sim Q^2 \text{ oder } \Delta p_t \sim Q^2 \tag{3-9}$$

w_m Mittlere Strömungsgeschwindigkeit in m s^{-1}

Q Luftvolumenstrom in m^3 s^{-1}

A Durchströmte Querschnittsfläche in m^2

Δp_v Druckabfall im Kanalnetz in Pa

Δp_t Druckaufbau durch Ventilatoren in Pa

Die Darstellung einer Anlagenkennlinie erfolgt unter Berücksichtigung des gesamten Druckabfalls Δp_v über veränderliche Volumenströme Q bei konstantem Klappen-Öffnungswinkel α. Der Öffnungswinkel beeinflusst dabei den Widerstandsbeiwert ζ der Klappe. Die Darstellung einer Ventilatorkennlinie erfolgt äquivalent mit dem statischen Druckaufbau Δp_t über veränderliche Volumenströme Q bei konstanter Drehzahl n. In Bild Bild 3-4 werden beide Kennlinien zueinander in Verhältnis gesetzt. Die Schnittstelle Ventilatorkennlinie-Anlagenkennlinie stellt gleichzeitig den Ergebniswert von Gleichung 3-6 dar - den Betriebspunkt - und der im nachfolgenden Kapitel beschriebene Wirkungsgrad η_V des Ventilators wird damit festgelegt.

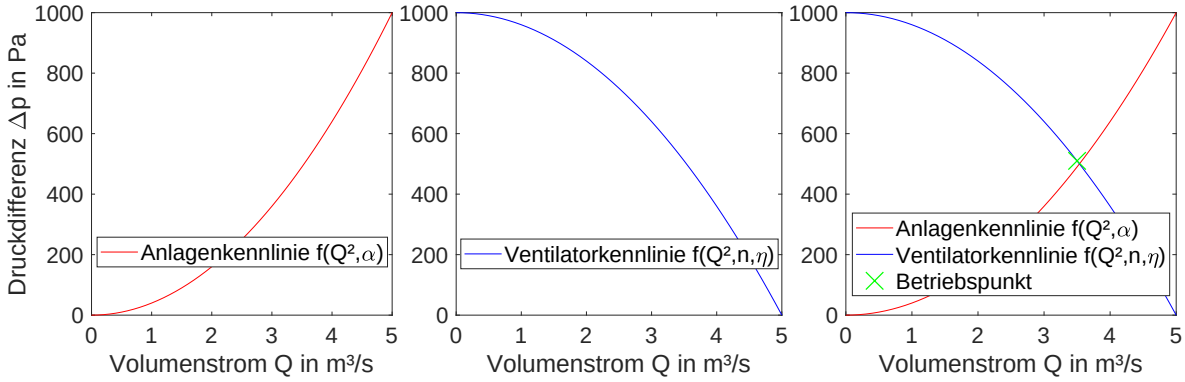


Bild: Bild 3-4 Schematische Darstellung einer Anlagenkennlinie, einer Ventilatorkennlinie und eines Betriebpunktes

Bei gleicher Geometrie und Größe der Anlagenkomponenten entstehen aufgrund veränderlicher Drehzahlen des Ventilators sowie veränderlicher Klappen-Öffnungswinkel - sowohl für die Anlagenkennlinie als auch für die Ventilatorkennlinie - Kurvenscharen. So können beispielsweise für einen erforderlichen Volumenstrom verschiedene Drücke aufgebaut werden bzw. abfallen. Anhand der Drehzahl kann der Betriebspunkt somit hin zu einem höheren Wirkungsgrad verschoben werden. Bei der Veränderung der Drehzahl sind allerdings Modellgesetze - hier Affinitätsgesetz - (siehe Gleichung 3-10) zu beachten. Wird ein geringerer Volumenstrom Q_2 benötigt, kann proportional die Drehzahl n_1 zu n_2 reduziert werden. Entsprechend Gleichung 3-10 ändert sich dadurch der Differenzdruck [**Weber**].

$$\frac{\Delta p_{t1}}{\Delta p_{t2}} = \left(\frac{n_1}{n_2}\right)^2 = \left(\frac{Q_1}{Q_2}\right)^2 \tag{3-10}$$

Δp_t Druckaufbau durch Ventilatoren des Zustands 1 oder 2 in Pa

n Drehzahl des Ventilators in Zustand 1 oder 2 in s^{-1}

Q Luftvolumenstrom des Zustands 1 oder 2 in $m^3\,s^{-1}$

Der Druckaufbau ist abhängig von der Geometrie des Ventilators. Grundlegend werden Radial-, Axial- oder Diagonalventilatoren unterschieden (Vergleich Bild Bild 3-5). In Axialventilatoren wird Druck aufgebaut, indem einströmende Luft durch Schaufeln umgelenkt wird. Die ausströmende Luft verlässt den Ventilator in spiralförmigen Bahnen. Wie groß der Druckaufbau ist, hängt vom relativen Winkel der Luftströmung zum Schaufelprofil ab. Je größer der Winkel ist, desto größer ist der Druckaufbau. Wird dieser zu groß, reißt die Profilströmung ab und der Ventilator arbeitet ineffizient. Weiterhin unterliegt die Luft in rotierenden Systemen - wie dem Ventilator - Zentrifugalkräften, welche diese nach außen drängt. Die hindurchströmende Luft wird dadurch in radiale Bahnen gezwungen, was zunehmend zum Druckaufbau beiträgt. In Radialventilatoren dominieren Zentrifugalkräfte. Wird also relativ zum Volumenstrom ein großer Druckaufbau benötigt, werden Radial- oder Diagonalventilatoren eingesetzt. In dieser Arbeit werden ausschließlich Radialventilatoren betrachtet, da in der kanalgebundenen Gebäudelüftung relativ zum Volumenstrom große Drücke aufgebaut werden müssen und sie daher am häufigsten eingesetzt werden [**IGTEUniversitatStuttgart**] [**Weber**].

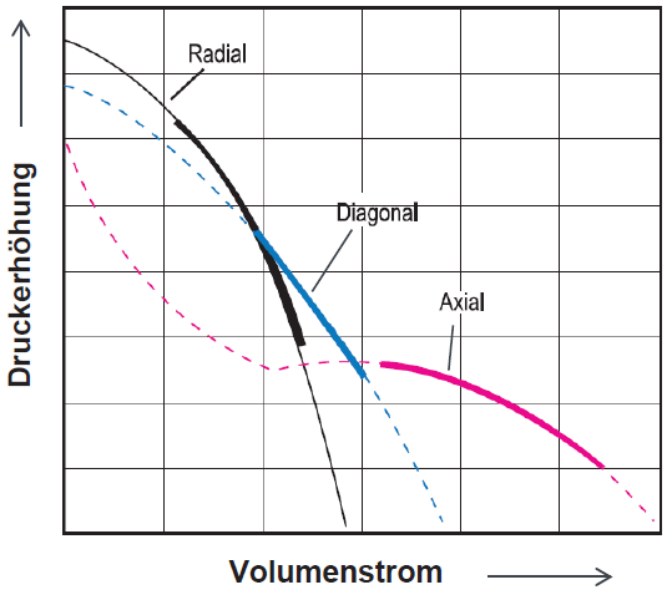


Bild: Bild 3-5 Vergleich der Ventilator-Kennlinien in Abhängigkeit der Bauarten Radialventilator, Axialventilator, Diagonalventilator [IGTEUniversitatStuttgart]

Werden Ventilatoren seriell oder parallel miteinander verschalten, verändern sich die Ventilatorkennlinien idealerweise (bei entsprechendem Abstand) nach Bild Bild 3-6. So werden für die serielle Verschaltung der Ventilatoren die Drücke der einzelnen Ventilatoren aufsummiert. Der Volumenstrom bleibt dabei für alle Ventilatoren gleich. Erfolgt eine parallele Verschaltung verändert sich der Druckaufbau der einzelnen Ventilatoren nicht. Die geförderten Volumenströme werden jedoch aufsummiert. Für die Betriebsgrößen gelten somit folgende Regeln:

Serienschaltung

$\Delta p = \Delta p_1 + \Delta p_2 + ... + \Delta p_n$

$Q = Q_1 = Q_2 = ... = Q_n$

Parallelschaltung

$\Delta p = \Delta p_1 = \Delta p_2 = ... = \Delta p_n$

$Q = Q_1 + Q_2 + ... + Q_n$

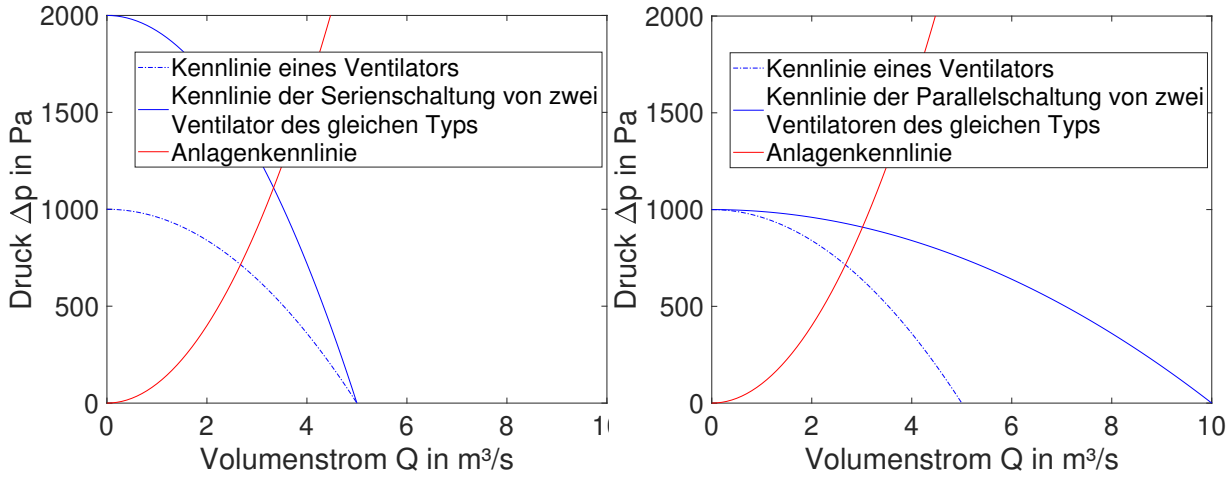


Bild: Bild 3-6 Kennlinien von zwei seriell (links) oder parallel (rechts) geschalteten Radial-Ventilatoren

3.2.2 Wirkungsgrad, Volumenstrom und Energiebedarf

Für die Berechnung des elektrischen Energiebedarfs eines Ventilators ist neben dem benötigten Druckaufbau auch die Kenntnis des Volumenstroms, bei welchem der Ventilator betrieben wird, und die des Gesamtwirkungsgrades erforderlich.

Der Gesamtwirkungsgrad η ist das Produkt aus den Wirkungsgraden des Motors η_M, des Antriebs η_A (falls der Ventilator nicht auf der Motorwelle befestigt ist) und des Ventilators η_V. Der Wirkungsgrad des Ventilators beschreibt die Verluste, die mit dem bereits erwähnten dynamischen Druck zusammenhängen. Hier werden unter anderem Minderleistungseffekte aufgrund einer endlichen Schaufelzahl, der Kanalreibung in allen strömungsführenden Bauteilen, der Spaltverluste im Laufradspalt und der Stoßverluste am Eintritt vom Laufrad oder Leitrad berücksichtigt. Da Druck und Volumenstrom einen unterschiedlich starken Einfluss auf die Verluste haben, ändert sich der Wirkungsgrad des Ventilators entlang der Konstant-Drehzahl-Ventilatorkennlinie. Der maximale Wirkungsgrad liegt dort, wo die Stoßverluste am kleinsten sind [**IGTEUniversitatStuttgart**] [**Weber**]. Ein reales Ventilatorkennfeld (Abbildung mehrerer Kennlinien) enthält daher neben dem Druckaufbau über dem variablen Volumenstrom auch alle Teillastbetriebe in Abhängigkeit von der Drehzahleinstellung und dem Wirkungsgrad (siehe Bild Bild 3-7).

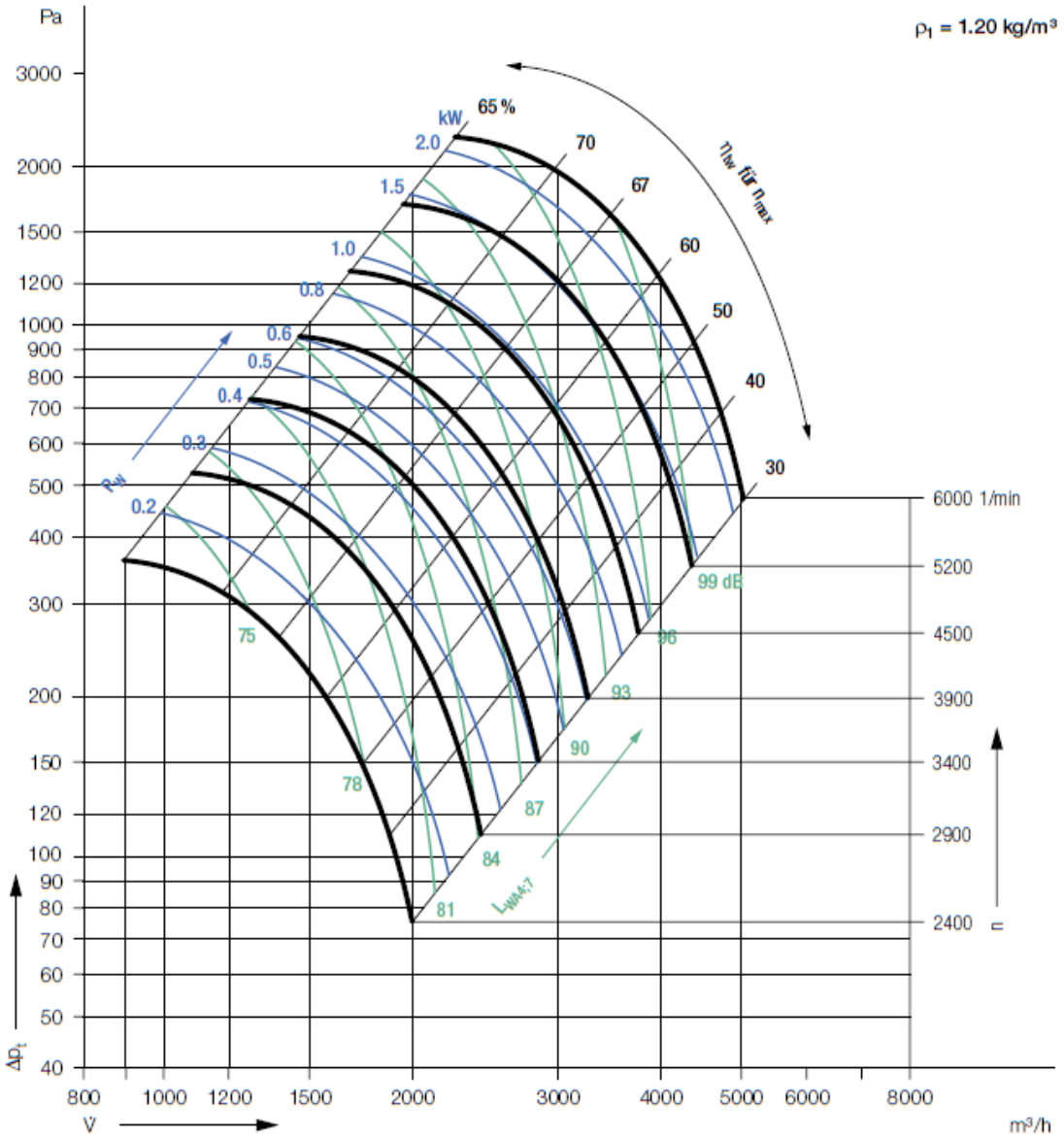


Bild: Bild 3-7 Ventilator-Kennfeld verschiedener Drehzahlen gemessen vom Hersteller; charakteristisch für Radialventilatoren mit rückwärts gekrümmten Schaufeln [GebhardtVentilatoren]

In der Gebäudebelüftung ist das Minimum eines Luftvolumenstroms meist durch erforderliche Hygiene- und Behaglichkeits-Anforderungen vorgegeben. Für den Außenluftbedarf spielen die Schadstoffabfuhr oder der Feuchteschutz eine wichtige Rolle. Das Maximum des Luftvolumenstroms leitet sich wiederum z.B. aus Anforderungen an den Wärmeschutz, Schallschutz, die maximale (behagliche) Luftgeschwindigkeit oder den elektrischen Energieaufwand der Luftverteilung ab. Der Außenluftvolumenstrom wird anhand eines normativ gegebenen Luftwechsels berechnet. Dieser gibt an, wie häufig das Zonenvolumen pro Stunde mit der Außenluft ausgetauscht wird. In Wohngebäuden werden Luftwechsel n von $0{,}4\,\mathrm{h}^{-1}$ bis $1{,}0\,\mathrm{h}^{-1}$ empfohlen, in Nichtwohngebäuden sind höhere Luftwechsel anzusetzen. Sie variieren hier deutlich in Abhängigkeit der Anzahl der Personen und der Zonennutzung (siehe Tabelle B.2 DIN EN 15251:2012) [**Maas**] [**DIN15251**]. Der gesamte Luftwechsel der Zone setzt sich zusammen aus dem Luftwechsel durch Infiltration, welche durch Luftundichtigkeiten der Gebäudehülle entsteht und der, die durch die Fenster- oder ventilatorgestützte Lüftung erfolgt. In dieser Arbeit ist ausschließlich letztere relevant und in Gleichung 3-11 berücksichtigt.

$$Q = V_{NE} * n_{mech} \tag{3-11}$$

V_{NE} Volumen einer Nutzungseinheit (Zonenvolumen) in m^3

n_{mech} Fenster- oder ventilatorgestützter Luftwechsel in h^{-1}

Q Erforderlicher Luftvolumenstrom in $m^3\,h^{-1}$

Daneben wird für den Volumenstrom im Kanalnetz - analog der ersten Kirchhoff´schen Regel bei elektrischem Strom im elektrischen Netz - die Knotenregel beachtet.

Knotenregel
Für einen Knoten im Kanalnetz, in den die Volumenströme $Q_i, (i = 1, ..., n)$ hinein fließen gilt:

$$\sum_{i=1}^{n} Q_i = 0 \tag{3-12}$$

Dabei ist die Fließrichtung zu beachten. Fließt der Volumenstrom aus dem Knoten heraus, ist er negativ, fließt er hinein, ist er positiv.

Der elektrische Leistungsbedarf des Ventilators wird dann nach Gleichung 3-13 berechnet. Der elektrische Energiebedarf ergibt sich aus dem Integral der Ventilatorleistung über die Betriebszeit der Lüftungsanlage bzw. des Ventilators. Für KVS wird das Integral zu Gleichung 3-14.

$$P_{el} = \frac{100 * Q * \Delta p_v}{\eta} \tag{3-13}$$

P_{el} Elektrischer Leistungsbedarf eines Ventilators in W

Q Erforderlicher Luftvolumenstrom in $m^3\,h^{-1}$

Δp_v Druckabfall im Kanalnetz in $N\,m^{-2}$

η Gesamtwirkungsgrad eines Ventilators in %

$$W_V = P_{el} * t_B \tag{3-14}$$

W_V Elektrischer Energiebedarf des Ventilators in W s bzw. J

P_{el} Elektrischer Leistungsbedarf eines Ventilators in W

t_B Ventilator-Betriebszeit in s

Für eine energieeffiziente Luftverteilung muss daher darauf geachtet werden, dass sowohl die zu überwindenden Strömungsverluste im Kanalnetz möglichst gering sind, als auch die Verluste des Ventilators, des Motors und des Antriebs. Bei einem vorgegebenen Luftvolumenstrom kann dies erfolgen, indem möglichst wenig Anlagenkomponenten eingebaut werden ($\Delta p_{v,min}$) und anschließend die Größe, Drehzahl sowie Verschaltung der Ventilatoren so gewählt werden,

dass die Ventilatoren bei möglichst maximalem Wirkungsgrad betrieben werden können.

4 Optimierungsproblem

In diesem Kapitel wird das in dieser Arbeit betrachtete Optimierungsproblem beschrieben.

4.1 Beschreibung des Optimierungsproblems

Zur Beschreibung des Optimierungsproblems werden zwei unterschiedliche Formulierungen verwendet. Das Problem wird zunächst technisch formuliert, um es ingenieurwissenschaftlich mit Anwendungsbezug abzubilden. Im Anschluss wird es mathematisch formuliert, um die formale Beschreibung der Optimierungsmethoden besser nachvollziehen zu können. Die verwendeten mathematischen und kombinatorischen Begriffe werden bereits in den Grundlagen erläutert.

4.1.1 Technische Beschreibung

Um die zu betrachtenden Optimierungsmethoden mit möglichst geringem Zeitaufwand modellieren und testen zu können, wird der Entscheidungsspielraum einer Luftverteilung - nachfolgend Referenzluftverteilung genannt - klein gewählt. Der Begriff Entscheidungsspielraum beschreibt hier die Anzahl der Freiheitsgrade des Optimierungsproblems. In diesem Zusammenhang sind folgende 13 Freiheitsgrade zu nennen und im Weiteren zu berücksichtigen:

13 Freiheitsgrade

1. Die Netzgeometrie
2. Die zeitliche Änderung der Volumenströme
3. Die Anzahl der Ventilatoren
4. Die Anzahl der Klappen
5. Die Anzahl der Kanalstücke
6. Die Position der Ventilatoren
7. Die Position der Klappen
8. Die Position der Kanalstücke
9. Die Dimensionierung der Ventilatoren
10. Die Dimensionierung der Klappen
11. Die Dimensionierung der Kanalstücke
12. Der Arbeitsbereich der Ventilatoren (Drehzahl)
13. Der Arbeitsbereich der Klappen (Öffnungswinkel)

Es besteht ausschließlich dann ein Optimierungsproblem (eine Optimierungsaufgabe), wenn mindestens einer dieser Freiheitsgrade vorhanden ist.

Die Netzgeometrie wird durch Strangabschnitte (graue Strich-Punkt-Linie) und Stränge (rote Linie) bestimmt (siehe Bild Bild 4-1). Die Referenzluftverteilung besteht aus zwei Strängen. Ein Strang ist dabei definiert durch einen möglichen Weg des Fluids vom Eingang bis zum Ausgang der Netzgeometrie. Für jeden Strang ist genau ein zunächst konstanter Volumenstrom Q_1 und Q_2 vorgegeben. Dieser wird durch den bereits erwähnten erforderlichen Volumenstrom der jeweiligen Zone bestimmt und nicht verändert (siehe Kapitel 3.2). Geltende Normen und Richtlinien sowie ein weiteres Vorgehen der Volumenstrombestimmung werde bereits in Kapitel 2.1 erläutert.

Wie Bild Bild 4-1 zeigt, wird ein Strangabschnitt vom Eingang des Stranges bis zu einem Knotenpunkt (hier Abzweigung) definiert. Die Indizes der Bezeichnung der Volumenströme der Stränge sind arabische Zahlen, für die Strangabschnitte werden römische Zahlen verwendet. Der Volumenstrom in Strangabschnitt I Q_I entspricht der Summe der Volumenströme in Strang I und Strang II ($Q_I = Q_1 + Q_2$). Analog entspricht der Volumenstrom in Strangabschnitt II Q_{II} dem Volumenstrom in Strang I Q_1 und der in Strangabschnitt III Q_{III} dem Volumenstrom in Strang II Q_2. Dadurch wird die Knotenregel (Kapitel 3.2) berücksichtigt. Bei der Berechnung des Druckabfalls oder -aufbaus in einer Komponente ist der Volumenstrom im Strangabschnitt von Bedeutung, nicht der im Strang.

Die Netzgeometrie beinhaltet eine definierte Anzahl an Komponenten. Diese sind begrenzt auf Ventilator V, Kanalstück R und Klappe K. Der Index der Komponenten in Bild Bild 4-1 (z.B. V_1-V_9) bezeichnet die in diesem Beispiel möglichen Positionen, nicht die definierte Anzahl. Auf deren Dimensionierung wird später in diesem Kapitel eingegangen. In dieser Betrachtung wird angenommen, dass das Kanalstück R für die Strömungsverluste in allen Wandelementen im jeweiligen Strangabschnitt steht und die Ströumgsverluste in der Abzweigung (im T-Stück) vernachlässigt werden.

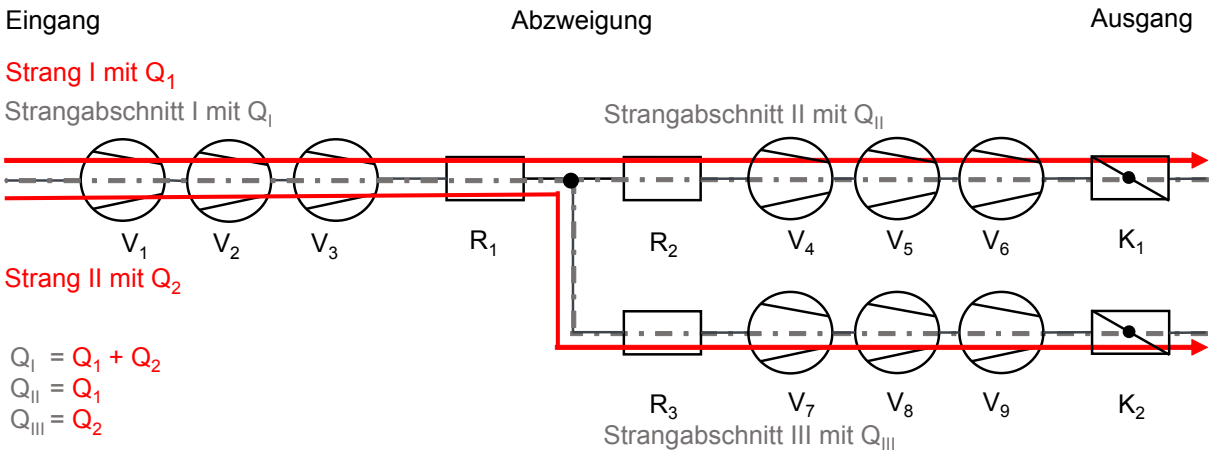


Bild: Bild 4-1 Schematische Darstellung der Referenzluftverteilung mit Strang und Strangabschnitt, möglichen Komponenten und deren Positionen

In der gewählten Referenzluftverteilung kommen ein bis drei Ventilatoren, drei Kanalstücke und zwei Klappen zum Einsatz. Pro Strangabschnitt muss genau ein Kanalstück eingesetzt werden. Beispielhafte Anordnungen der Referenzluftverteilung sind in Bild Bild 4-2 dargestellt. In der Darstellung ist die Positionierung so gewählt, dass mindestens ein Ventilator und eine Klappe pro Strang enthalten sind. Jeder Strangabschnitt enthält keinen bis alle Ventilatoren.

Bisher sind die Positionen und Anzahl der Klappen und Kanalstücke sowie die Anzahl an Strängen und Strangabschnitten (Netzgeometrie) festgelegt. Weiterhin bleibt eine zeitliche Änderung des Volumenstroms unberücksichtigt. Dadurch wird das Optimierungsproblem um sechs Freiheitsgrade (1,2,4,5,7,8) reduziert, es verbleiben noch sieben: Die Position und die Anzahl des/der Ventilator/en, die Arbeitsbereiche und die Dimensionierung aller Komponenten.

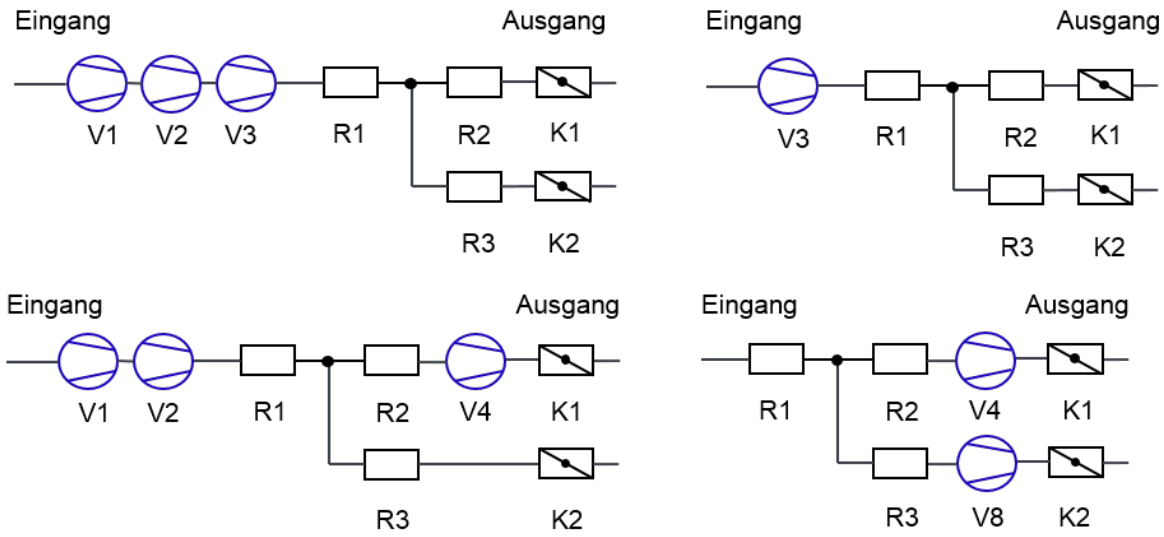


Bild: Bild 4-2 Schematische Darstellung beispielhafter Ventilator-Anordnungen im Referenzluftverteilsystem

Die Komponenten werden mit Hilfe analytischer Beschreibungen theoretischer Kennfelder abgebildet. Diese sind dabei an die in der Praxis typischen Komponenten-Eigenschaften (siehe Kapitel 3.2) angelehnt. Sie beschreiben die Dimensionierung und den Arbeitsbereich. In Bild Bild 4-3 wird das Kennfeld eines Ventilator-Typs dargestellt. Die blauen Kennlinien stellen den Druckaufbau über einem veränderlichen Volumenstrom bei einer konstanten Drehzahleinstellung dar. Letztere verändert den Arbeitsbereich entsprechend der Modellgesetze (Gleichung 3-10). Sie wird hier in Prozent angegeben. Bei der Wahl einer Drehzahl kleiner als 100 %, wird der Ventilator in Teillast betrieben. 100 % entsprechen der höchsten Drehzahleinstellung des Ventilator-Typs und damit der Volllast. Die Ventilator-Kennfelder sind nur im positiven Wertebereich (positive Drücke und positive Volumenströme) definiert. Da mindestens ein Ventilator pro Strang vorgesehen wird, ist damit die Fließrichtung des Systems festgelegt.

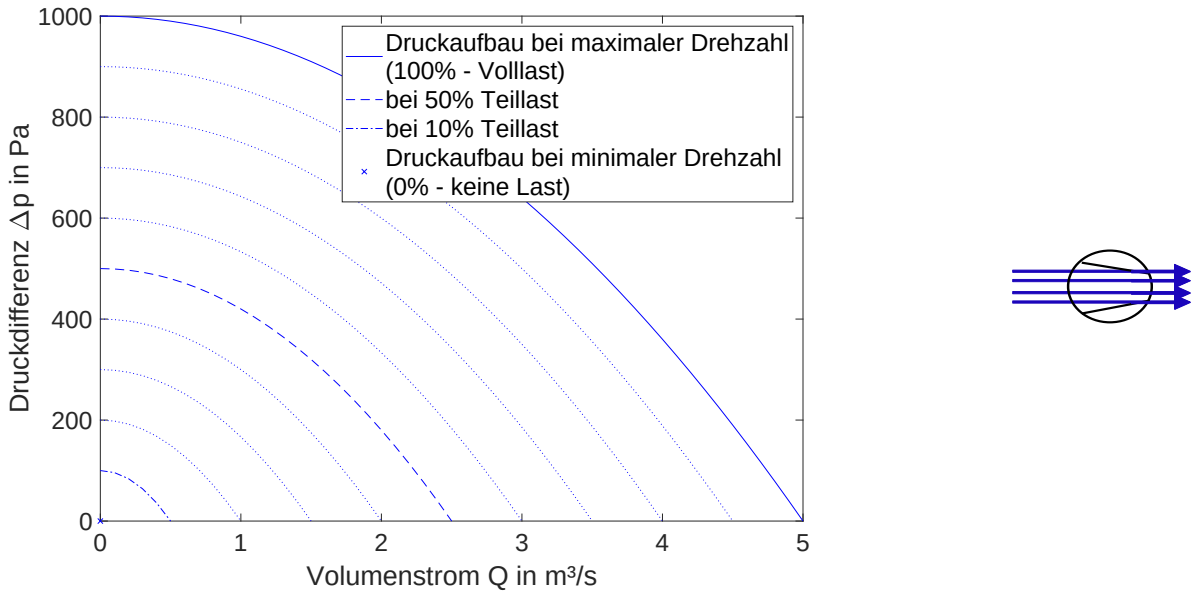


Bild: Bild 4-3 Theoretisches Kennfeld eines Ventilators

In der Betrachtung werden vier Ventilator-Typen berücksichtigt. Typ 1 baut bei gleichem Volumenstrom und gleicher Last (hier Volllast) nur ein Viertel des Druckes von Typ 4 auf. Die Ventilator-Typen unterscheiden sich also durch die Steigungen der Kennlinien im Kennfeld. Ein Faktor - im Weiteren *Skalierfaktor* genannt - sorgt für die Skalierung des Druckaufbaus. So ist bei gleichem Volumenstrom, der Druckaufbau für jeden Typ unterschiedlich. Je kleiner der Skalierfaktor, desto weniger Druck kann aufgebaut werden. Bild Bild 4-4 stellt die Kennlinien der Typen unter Volllast dar.

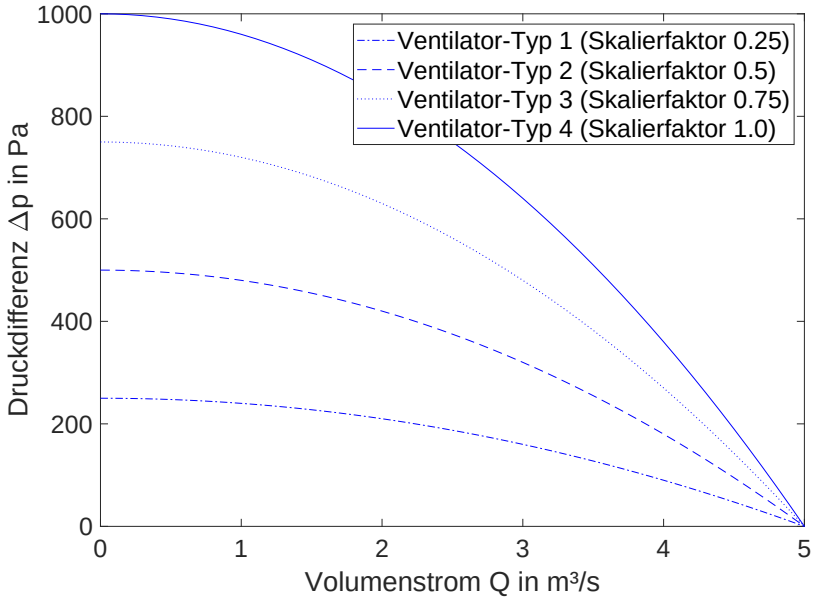


Bild: Bild 4-4 Theoretische Kennlinien der Ventilator-Typen (Freiheitsgrad Dimensionierung) unter Volllast

Der Druck-Skalierfaktor wirkt analog der Serienschaltung (siehe Kapitel 3.2) mehrerer Ventilatoren des Typs 1. Der Unterschied zwischen dem Einsatz eines Ventilators mit Skalierfaktor 1 und einer Reihenschaltung von vier Ventilatoren mit Skalierfaktor 0,25 liegt in dem, in Kapitel 3.2 beschriebenen Wirkungsgrad der jeweiligen Ventilatoren und der Strömungsverluste aufgrund nicht-vollausgebildeter Strömungsprofile. In Kapitel 4.1.2 wird näher auf den Wirkungsgrad im theoretischen Ventilator-Kennfeld eingegangen.

Bild Bild 4-5 stellt das theoretische Kennfeld der Kanalstücke graphisch dar. Die Dimensionierung wird durch den feststehenden Typ der Kanalstücke festgelegt. Diese Komponenten sind nicht veränderlich, sodass der Freiheitsgrad „Arbeitsbereich Kanalstück“ entfällt. Für den vorgegebenen Volumenstrom entsteht immer ein eindeutiger Druckabfall. Die entsprechende analytische Beschreibung ist im nachfolgenden Kapitel erläutert.

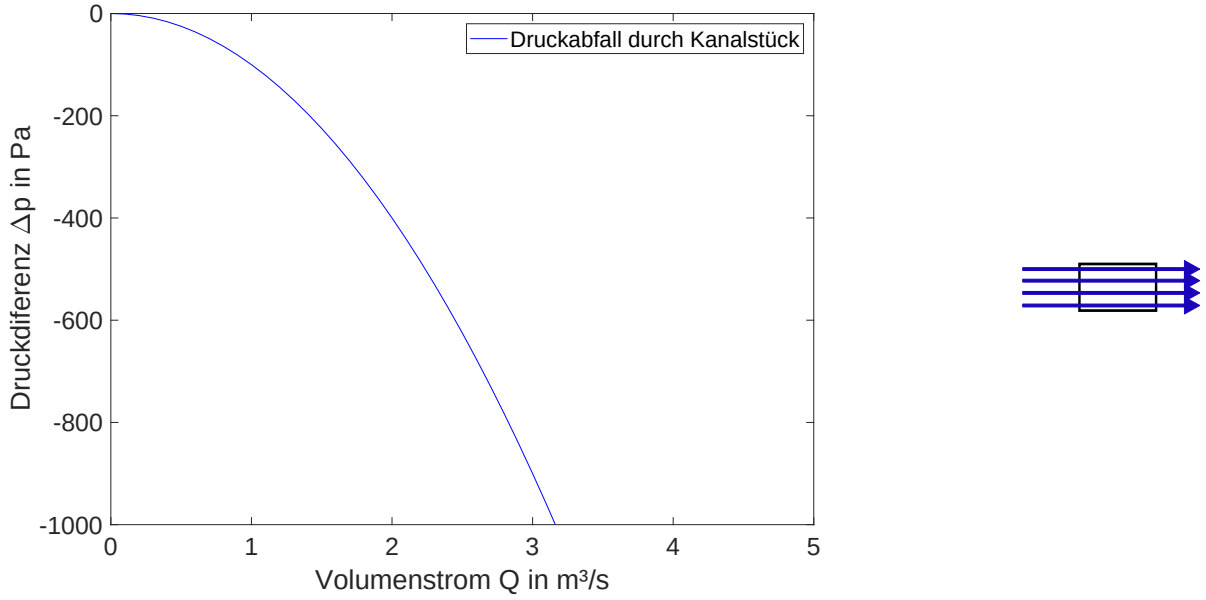


Bild: Bild 4-5 Theoretisches Kennfeld eines Kanalstücks (links). Schematische Darstellung des durchströmten Kanalstücks (rechts).

Bild Bild 4-6 stellt das theoretische Kennfeld der Klappen dar. Es wird genau wie für die Kanalstücke festgelegt, dass nur ein Typ der Klappen zur Auswahl steht. Damit ist die Dimensionierung festgelegt, die Klappen-Öffnungswinkel α bleiben jedoch, wie in Kapitel 3.2 erwähnt, variabel. Der Öffnungswinkel der Klappen kann physikalisch sinnvoll von null (geschlossen) bis neunzig (geöffnet) Grad variieren und verändert dadurch den Druckabfall, beziehungsweise den Arbeitsbereich der Klappe. Um die Druckgleichung 3-6 erfüllen zu können, kann bei festgelegtem Druckabfall der Kanalstücke (mit Position und Dimensionierung), der Arbeitsbereich der Klappe (Öffnungswinkel) entsprechend der gewählten Freiheitsgrade des/der Ventilatoren angepasst werden. Die Referenzluftverteilung wird damit um diesen weiteren Freiheitsgrad (13), bedingt durch die Wahl der Freiheitsgrade des/der Ventilatoren, reduziert.

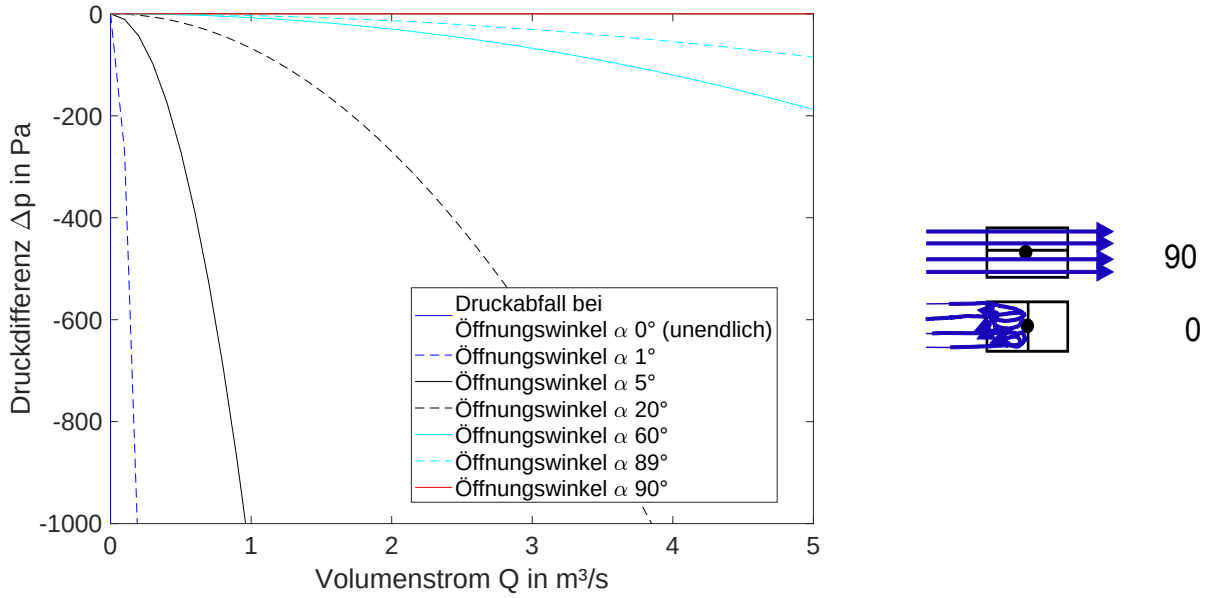


Bild: Bild 4-6 Theoretisches Kennfeld einer Klappe bei verschiedenen Öffnungswinkeln ($0°$ - geschlossen, $90°$ - geöffnet)

Ist die Klappe geschlossen, ist der Druckabfall maximal (unendlich). Hier wird idealisiert angenommen, dass die komplette Strömung unterbrochen wird und keine Leckagen vorliegen. Ist die Klappe geöffnet, wird angenommen, dass diese einen sehr geringen Strömungswiderstand darstellt. Dieser wird wesentlich geringer angenommen, als der eines Kanalstücks. So kann für nahezu jeden Druckaufbau ein Druckabfall gefunden werden, mit welchem die Druckgleichung 3-6 erfüllt wird. Die Ventilator-Dimensionierung wird dadurch weniger eingeschränkt.

Zusammengefasst ist das Optimierungsproblem durch die Beschreibung der Referenzluftverteilung und dessen Entscheidungsspielraum gegeben. Die Freiheitsgrade 1, 2, 4, 5, 7, 8, 10, 11 und 13 werden bereits im Vorfeld ausgeschlossen, wodurch der Entscheidungsspielraum wesentlich reduziert wird. Es bleiben vier Freiheitsgrade – Anzahl, Positionierung, Dimensionierung und Arbeitsbereich der Ventilatoren – und alle kombinatorischen Zusammensetzungen dieser.

Gesucht wird der minimale elektrische Leistungsbedarf der Referenzluftverteilung. Die Freiheitsgrade und deren kombinatorische Zusammensetzung verändern, wie in Kapitel 2 beschrieben, den Leistungsbedarf. Es ist zu beachten, dass bei konstanter Anzahl und Position der Komponenten innerhalb eines Strangabschnitts, die Reihenfolge der Komponenten keine Veränderung für den Leistungsbedarf darstellt. Die Position der Komponente beschreibt in diesem Zusammenhang daher nicht den exakten Ort im Kanalnetz, sondern den Strangabschnitt der Netzgeometrie, in der sich die Komponente befindet.

Gelöst wird das Optimierungsproblem mit den Methoden in Kapitel 5.

4.1.2 Mathematische Beschreibung

Um das Optimierungsproblem mathematisch beschreiben zu können, wird zunächst die allgemeine Form definiert. Anhand der Eigenschaften der Funktionen und Grundmengen sowie den Ansprüchen an die Lösung wird das Optimierungsproblem anschließend klassifiziert.

Allgemeine Form: Die allgemeine Form beinhaltet die Instanz und die Aufgabe des Optimierungsproblems. Die Instanz setzt sich zusammen aus der Zielfunktion und der Lösungsmenge (Kapitel 3.1). Die Lösungsmenge gibt unter Beachtung der Gleichungs- und Ungleichungs-Nebenbedingungen alle zur Berechnung des Zielfunktionswertes benötigten Parameter und deren Zahlenraum an. Die mathematische Lösungsmenge entspricht dem technischen Entscheidungsspielraum. Ein Parameter entspricht einem der angegebenen Freiheitsgrade für jede Komponente. Die Instanz der Referenzluftverteilung soll diese Definitionen verdeutlichen. Die Aufgabe ist es, das globale Minimum der Zielfunktion zu finden. Dies muss unter der Bedingung gelöst werden, dass alle Parameter zur Berechnung des Zielfunktionswerts innerhalb der Lösungsmenge liegen. Damit ist die Lösung dieses Problems gültig.

Nachfolgend gibt $numFan$ die Anzahl der Ventilatoren im Luftverteilsystem (Freiheitsgrad 3) und $pos_{1,...,f}$ die Position des Ventilators (Freiheitsgrad 6) an. Letztere ist ein Indikator, welcher das Vorhandensein (1) oder nicht Vorhandensein (0) des jeweiligen Ventilators $(1, ..., numFan)$ im Strangabschnitt beschreibt. Er wird, wie später in diesem Kapitel beschrieben, als 2-Tupel angegeben. Der Parameter $tp_{1,...,f}$ entspricht dem jeweiligen Ventilator-Typ (Freiheitsgrad 9), $n_{1,...,f}$ der jeweiligen Drehzahleinstellung (Freiheitsgrad 12).

Allgemeine Form Referenz-Optimierungsproblem

Instanz: Die Zielfunktion $P(numFan, pos_{1,...,numFan}, tp_{1,...,numFan}, n_{1,...,numFan})$, in Abhängigkeit der Parameter $numFan$, $pos_{1,...,numFan}$, $tp_{1,...,numFan}$, $n_{1,...,numFan}$. Die Grundmengen $\{numFan \mid numFan \in \mathbb{N}, numFan > 0\}$, $\{pos \mid pos = 0, pos = 1\}$, $\{tp \mid tp \in \mathbb{N}, tp > 0\}$, $\{n \mid n \in \mathbb{R}_+ , n \leq 100\}$. Wobei die Grundmengen begrenzt werden auf die Lösungsmengen $numFan, f \varepsilon \{1, 2, 3\}$, $pos \varepsilon \{0, 1\}$, $tp \varepsilon \{1, 2, 3, 4\}$ und $n \varepsilon \{0, 10, 20, 30, ..., 100\}$. Die Ungleichungs-Nebenbedingung lautet $P(numFan, pos_{1,...,numFan}, tp_{1,...,numFan}, n_{1,...,numFan}) \geq 0$.

Aufgabe: Finden des globalen Minimums von $P(numFan, pos_{1,...,numFan}, tp_{1,...,numFan}, n_{1,...,numFan})$ (Zielfunktionswert) und Bestimmung zugehöriger Parameter innerhalb der Lösungsmenge.

Aufstellung der Zielfunktion: Bevor die Klassifizierung des Optimierungsproblems stattfinden kann, muss die Zielfunktion aufgestellt werden. Hierzu ist die analytische Beschreibung der theoretischen Ventilator-Kennfelder notwendig. Für die analytische Beschreibung des theoretischen Kennfeldes der Ventilatoren in Bild Bild 4-3 wird die im Ventilator erzeugte Druckdifferenz Δp_t in Abhängigkeit der Drehzahleinstellung, des Ventilator-Typs und des Strangabschnittvolumenstroms berechnet.

$$\Delta p_t(n, tp, Q) = tp * 10 * n - (\frac{tp * 10 * n}{(\frac{n}{20})^2} * Q^2) \tag{4-1}$$

Δp_t Druckaufbau durch einen Ventilator in Pa

tp Einheitsloser Ventilator-Typ

n Ventilator-Drehzahleinstellung in %

Q Strangabschnittsvolumenstrom in $m^3\,s^{-1}$

$$P_{fan}(\Delta p_t, Q, \eta) = \frac{Q * \Delta p_t(n, tp, Q)}{\eta} \tag{4-2}$$

P_{fan} Elektrischer Leistungsbedarf eines Ventilators in W

Q Strangabschnittsvolumenstrom in $m^3\,h^{-1}$

Δp_t Druckaufbau durch einen Ventilator in Pa

η Gesamtwirkungsgrad eines Ventilators (einheitslos)

$$P(numFan, pos_{1,...,numFan}, tp_{1,...,numFan}, n_{1,...,numFan}) = \sum_{m=1}^{numFan} P_{fan}(Q, \Delta p_t, \eta)_m * pos_m \tag{4-3}$$

P	Elektrischer Leistungsbedarf aller Ventilatoren einer Luftverteilung in Abhängigkeit der Optimierungsproblem-Parameter in W
P_{fan}	Elektrischer Leistungsbedarf eines Ventilators in W
Q	Strangabschnittsvolumenstrom in $m^3\,h^{-1}$
Δp_t	Druckaufbau durch einen Ventilator in Pa
η	Gesamtwirkungsgrad eines Ventilators (einheitslos)
pos	Indikator zur Position des Ventilators (Vorhandensein (1) oder nicht Vorhandensein (0))

Gleichung 4-1 beschreibt stetige Kurven für jede Drehzahleinstellung - Parabeln zweiten Grades, welche eindeutig bestimmbar sind, da sie ausschließlich im Wertebereich des ersten Quadranten definiert sind. Der Leistungsbedarf P_{fan} kann mit Gleichung 4-2 analog der Grundlagen-Gleichung 3-13 berechnet werden. Dieser ist nicht-linear abhängig von den Parametern n, tp und Q. Der Leistungsbedarf der gesamten Luftverteilung P ergibt sich aus der Summe der Leistungsbedarfe der eingesetzten Ventilatoren.

Der Wirkungsgrad η wird mit Hilfe der in MATLAB integrierten 2D-Interpolation (interp2) berechnet. Die Interpolation findet für Wirkungsgrade zwischen 80 % und 30 % statt. Dabei wird jedem Punkt $(Q|\Delta p)$ eines Gitters entsprechend Bild Bild 4-7 ein Wirkungsgrad zugewiesen. Zwischen den Punkten findet die Interpolation statt. Die Zuweisung erfolgt für jeden Ventilator-Typ separat. Bild Bild 4-7 stellt das Gitter für Ventilator-Typ 4 dar. Das formale Modell der Interpolation ist dem Anhang (Kapitel 10.3.1) beigefügt. Der Wirkungsgrad ist somit eine Funktion von Typ, Druckaufbau und Volumenstrom. Dieser ergibt sich durch die Angabe der Druckdifferenz aus Gleichung 4-1 und dem erforderlichen Strangabschnittvolumenstrom.

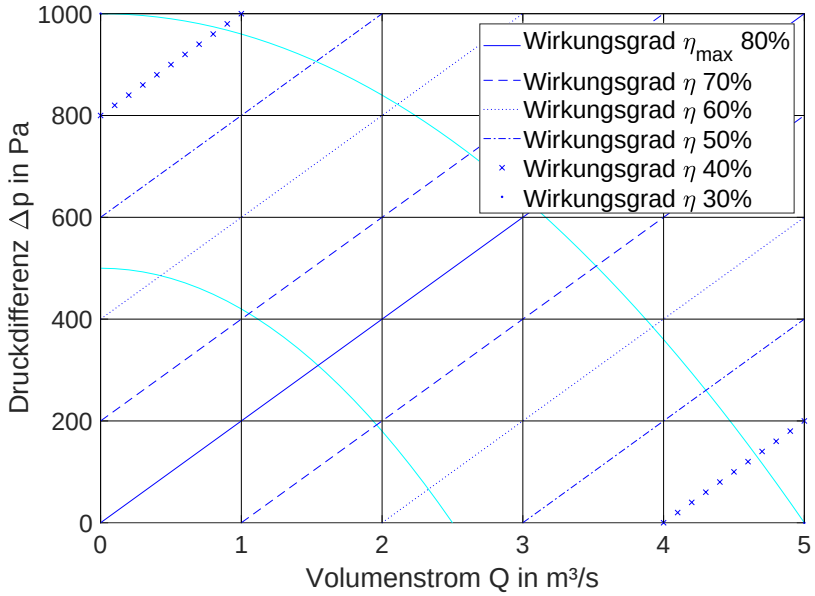


Bild: Bild 4-7 Theoretisches Kennfeld eines Ventilators mit Angabe eines Gitters zur Wirkungsgradbestimmung η mit Hilfe einer zweidimensionalen Interpolation

Die Komponenten, Klappen und Kanalstücke, sind für die Klassifizierung des Optimierungsproblems nicht relevant. Sie werden in Kapitel 5.1.2 bei einer Prüfung des Klappen-Öffnungswinkels benötigt. Deren analytische Beschreibungen, Gleichung 4-5 und Gleichung 4-6, werden in Abhängigkeit des Strangabschnittvolumenstroms Q berechnet. Alle eingesetzten Klappen und Kanalstücke werden aufsummiert (4-4) und in die bereits erwähnte Druckgleichung eingesetzt.

$$\Delta p_v = \sum \Delta p_{v,K} + \sum \Delta p_{v,R} \tag{4-4}$$

Δp_v Druckabfall im Kanalnetz in Pa

$\Delta p_{v,K}$ Druckabfall in einer Klappe in Pa

$\Delta p_{v,R}$ Druckabfall in einem Kanalstück in Pa

$$\Delta p_{v,K}(\alpha, Q) = -\frac{300}{\frac{(\alpha^2)}{90}} * Q^2 \tag{4-5}$$

$\Delta p_{v,K}$ Druckabfall in einer Klappe in Pa

α Klappen-Öffnungswinkel in °

Q Strangabschnittsvolumenstrom in $m^3\,s^{-1}$

$$\Delta p_{v,R}(Q) = -100 * Q^2 \tag{4-6}$$

$\Delta p_{v,R}$ Druckabfall in einem Kanalstück in Pa

Q Strangabschnittsvolumenstrom in $m^3\,s^{-1}$

Klassifizierung des Problems: Mit Hilfe der allgemeinen Form und der analytischen Beschreibung der Ventilator-Kennfelder, kann das Referenz-Optimierungsproblem Klassen zugeordnet werden [**CzyzykJ**] [**ElizabethD.**] [**WilliamGroppandJorgeJ.More**].

Optimierungsproblem-Klassen

- kontinuierliche oder diskrete Optimierungsprobleme
  - diskret-kombinatorische Optimierungsprobleme
  - diskret-ganzzahlige Optimierungsprobleme (Integer Programming IP)
  - gemischt-diskret-kontinuierliche Optimierungsprobleme (Mixed Integer Programming MIP)
- Optimierungsprobleme mit oder ohne Bedingungen
- Optimierungsprobleme mit keiner, einer oder vielen Zielfunktionen
- deterministische oder stochastische Optimierungsprobleme

Innerhalb jeder dieser Klassen kann wiederum ein lineares oder nichtlineares, globales oder lokales Optimierungsproblem definiert werden. Eine Kombination aus mehreren Klassen be-

schreibt das Optimierungsproblem voll. Eine der möglichen Taxonomien (Zweig der Systematik) der Optimierungsprobleme ist in Bild Bild 4-8 dargestellt.

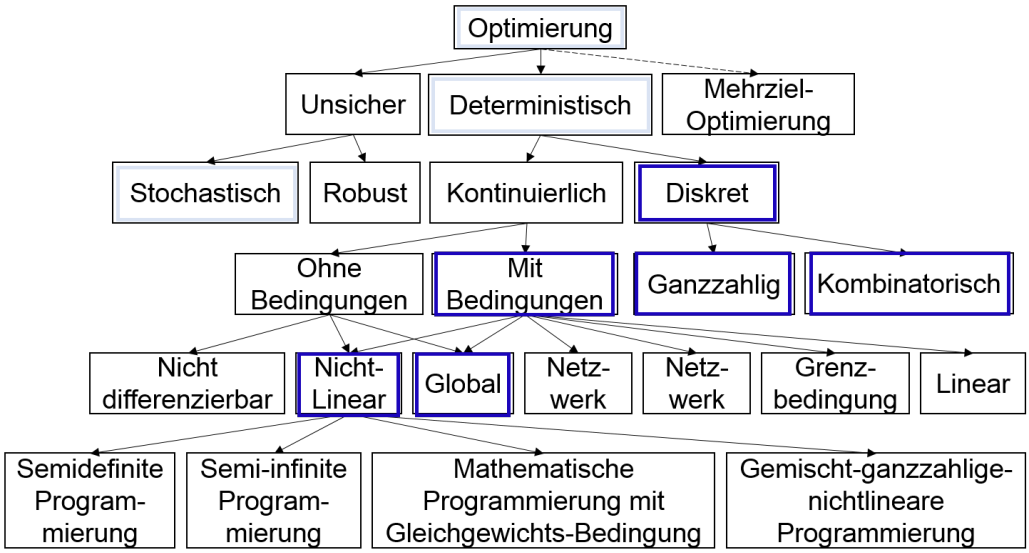


Bild: Bild 4-8 Eine mögliche Taxonomie der Optimierungsprobleme entsprechend dem "NEOS Optimization Guide"[CzyzykJ] [ElizabethD.] [WilliamGroppandJorgeJ.More]

Das in dieser Arbeit zu betrachtende Optimierungsproblem hat eine nicht-lineare Zielfunktion, für welche genau ein globales Minimum gefunden werden soll. Die Zielfunktion ist begrenzt durch eine Ungleichungs-Nebenbedingung. Die Parameter werden im ganzzahligen Zahlenraum gesucht und sind auf endliche Werte begrenzt, wobei nur die kombinatorische Zusammensetzung der Parameter zu einer Lösung führt (diskret-kombinatorisch). Die gültige Lösung beinhaltet den minimalen Zielfunktionswert P_{min} und die kombinatorische Zusammensetzung der Parameter, welche innerhalb der Lösungsmenge liegen. Es gilt daher mathematisch ein nichtlineares, globales Optimierungsproblem der Klasse diskret-kombinatorisch mit genau einer Zielfunktion und unter Bedingungen zu lösen. Dabei wird noch nicht festgelegt, ob das Problem deterministisch oder stochastisch zu lösen ist. Diese Zuordnung erfolgt abhängig von den Optimierungsmethoden in Kapitel 5.

Kombinatorik des Problems: Nachdem das Optimierungsproblem klassifiziert ist, kann auf die Kombinatorik als Teilgebiet der Mathematik eingegangen werden. Anhand der Formeln der Kombinatorik und gegebener Instanz, kann die Anzahl an kombinatorischen Zusammensetzungen der Parameter berechnet werden. Wie in Kapitel 3.1 erläutert, unterscheidet die Kombinatorik grundlegend in Anordnungen (Permutationen) und Auswahlen (Kombinationen,Variationen). Die kombinatorischen Zusammensetzungen der Referenzluftverteilung können mit Auswahlen beschrieben werden. Es sind genauer, geordnete Auswahlen mit Wiederholung (Variationen) für die Drehzahleinstellungen und für die Ventilator-Typen. So findet beispielsweise eine Auswahl von $numFan$ Ventilatoren aus der Lösungsmenge der Ventilator-Typen $tp\ \varepsilon\ \{1,2,3,4\}$ ($numtp = 4$) statt. Genauso wird eine Auswahl von $numFan$ Ventilatoren aus der Lösungsmenge der Drehzahleinstellungen $n\ \varepsilon\ \{0,10,20,30,...,100\}$ ($numn = 11$) getroffen. Die Auswahl beschreibt dabei nicht einen Wert (z.B. $tp = 1$), sondern das kombinatorische Element der Werte (z.B. $tp_1 = 3, tp_2 = 3, tp_3 = 4$). Da die Reihenfolge beachtet wird, ist die Variation (3,3,4) eine andere als beispielsweise die Variation (3,4,3). An diesem Beispiel wird auch deutlich,

was Wiederholung in diesem Zusammenhang bedeutet: In einer Auswahl kann Typ 3 mehr als einmal vorkommen.

Für die Positionen der Ventilatoren werden die beiden Stränge der Referenzluftverteilung als 2-Positions-Tupel betrachtet. So kann erkannt werden, in welchem Strangabschnitt der jeweilige Ventilator positioniert ist und damit, bei welchem Volumenstrom der Ventilator betrieben wird. Ist der Ventilator in beiden Strängen vorhanden (z.B. in Strangabschnitt I, Bild Bild 4-1), ist die Position des Ventilators mit dem Positions-Tupel $pos = \left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]$ angegeben. Ist der Ventilator nur in einem der Stränge vorhanden (z.B. Strangabschnitt II oder III), ist die Position mit dem Positions-Tupel $\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$ oder $\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$ gegeben. Wenn kein Ventilator vorhanden ist, entspricht das Positions-Tupel der Angabe $\left[\begin{smallmatrix}0\\0\end{smallmatrix}\right]$. Die Anzahl der Ventilator-Positions-Tupel $numFanPos$ beträgt für die Referenzluftverteilung mit zwei Strängen somit insgesamt vier. Gleiches wird für die Kanalstücke und Klappen durchgeführt. Diese Positions-Tupel werden, im Vergleich zu den Ventilator-Positions-Tupel, einmal festgelegt und bleiben unveränderlich.

Für jeden der $numFan$ Ventilatoren muss jedes der Positions-Tupel zur Auswahl stehen. Für die Berechnung des Leistungsbedarfes ist es dabei nicht von Bedeutung, in welcher Reihenfolge die $numFan$ Positions-Tupel aus den vier möglichen ausgewählt werden, weshalb es kombinatorisch eine ungeordnete Auswahl mit Wiederholung, genauer eine Kombination mit Wiederholung ist. Die Kombination $\left[\begin{smallmatrix}0&1&1\\1&1&1\end{smallmatrix}\right]$ ist hier die gleiche wie die Kombination $\left[\begin{smallmatrix}1&0&1\\1&1&1\end{smallmatrix}\right]$.

Da die Auswahlen von der Anzahl der Ventilatoren abhängig sind, muss dieser Freiheitsgrad (Nr. 3), bevor die Auswahlen getroffen werden, festgelegt werden. Für die Berechnung der Auswahlen werden folgende Gleichungen der Kombinatorik genutzt:

$$num\pi_D = numn^{numFan} \tag{4-7}$$

$num\pi_D$	Anzahl der Drehzahl-Variationen
$numn$	Einheitslose Menge aller zur Auswahl stehenden Drehzahleinstellungen
$numFan$	Maximal zulässige Anzahl der Ventilatoren im System

$$num\pi_T = numtp^{numFan} \tag{4-8}$$

$num\pi_T$	Anzahl der Ventilator-Typ-Variationen
$numtp$	Einheitslose Menge aller zur Auswahl stehenden Ventilator-Typen

$$num\pi_P = \binom{\text{numFanPos} + \text{numFan} - 1}{\text{numFan}} \tag{4-9}$$

$num\pi_P$	Anzahl der Positions-Tupel-Kombinationen
$numFanPos$	Einheitslose Menge aller zur Auswahl stehenden Positions-Tupel

Gleichung 4-10 wird dabei als Binomialkoeffizient dargestellt. Nachfolgendes Beispiel stellt die Berechnung dieser vereinfachten Schreibweise dar. Für $numFanPos = 3$ und $numFan = 2$

gilt:

$$\binom{\text{numFanPos} + \text{numFan} - 1}{\text{numFan}} = \binom{3+2-1}{2} = \binom{4}{2} = \frac{4!}{2!(4-2)!} = 6 \tag{4-10}$$

Die Tabellen in Bild Bild 4-9 stellen die Variationen und Kombinationen schematisch als Matrizen dar. Jede Variation der Ventilator-Typen (Auswahl π_{T1} bis $\pi_{Tnum\pi_T}$) kann wiederum mit jeder Variation der Drehzahleinstellungen (Auswahl π_{D1} bis $\pi_{Dnum\pi_D}$) und jeder Kombination der Positions-Tupel (Auswahl π_{P1} bis $\pi_{Pnum\pi_P}$) kombiniert werden. Eine Anordnung der Auswahlen wird hier als Parameter-Permutation bezeichnet. Dabei muss beachtet werden, dass keine Permutation zweimal auftritt. Die Permutation ist außerdem begrenzt durch die Bedingung, dass in jeder Anordnung genau eine von jeder Auswahl vorhanden ist, die Reihenfolge der Auswahlen ist dabei festgelegt ([Auswahl π_P, Auswahl π_T, Auswahl π_D]). Damit wird die Anzahl der Permutationen $numParPerm$ hier nicht mit der kombinatorischen Gleichung $n!$ berechnet. Ein Schema wie jede der Auswahlen abgebildet werden kann, ist im Anhang (Bild Bild 10-1) zu finden.

Für die Referenzluftverteilung resultiert daraus:

$numn^{numFan} * numtp^{numFan} * \binom{\text{numFanPos}+\text{numFan}-1}{\text{numFan}} = 1.331 * 64 * 20 = 1.703.680$

Parameter-Permutationen $numParPerm$. Unter der Bedingung, dass mindestens ein Ventilator pro Strang vorhanden sein muss, reduziert sich die Anzahl der Positions-Tupel-Kombinationen im Referenzluftverteilsystem von 20 auf 13. Trotzdem bleiben über eine Million Permutationen, welche unterschiedliche Leistungsbedarfe liefern können.

Name der Auswahl	Ventilator 1 bis numFan		
	V1	V2	V3
$\pi_{P\,1}$	pos = 1 1	pos = 1 1	pos = 1 1
$\pi_{P\,2}$	pos = 1 1	pos = 1 1	pos = 0 0
…	…	…	…
$\pi_{P\,num\pi P}$	pos = 0 1	pos = 0 1	pos = 0 1

a)

Name der Auswahl	Ventilator 1 bis numFan		
	V1	V2	V3
$\pi_{T\,1}$	tp = 1	tp = 1	tp = 1
$\pi_{T\,2}$	tp = 2	tp = 1	tp = 1
…	…	…	…
$\pi_{T\,num\pi T}$	tp = 4	tp = 4	tp = 4

b)

Name der Auswahl	Ventilator 1 bis numFan		
	V1	V2	V3
$\pi_{D\,1}$	n = 0%	n = 0%	n = 0%
$\pi_{D\,2}$	n = 10%	n = 0%	n = 0%
…	…	…	…
$\pi_{D\,num\pi D}$	n = 100%	n = 100%	n = 100%

c)

Bild: Bild 4-9 a) Kombination der Positions-Tupel, b) Variationen der Ventilator-Typen und c) Variationen der Ventilator-Drehzahlen schematisch dargestellt in Matrizenform.

Nachfolgend werden die entwickelten Methoden zur Lösung dieses Optimierungsproblems beschrieben.

5 Optimierungsmethoden für Luftverteilsysteme mit Optimierungsaufgabe

Um das Optimierungsproblem zu lösen, wird es, inklusive der Arbeitsanweisungen zur Lösung, mit der in MATLAB integrierten Programmiersprache, in ein formales Modell übersetzt. Solche formalen Modelle führen jedoch zu sehr langen und aufwendigen Beschreibungen, weshalb in diesem Kapitel jedes Modell - im Folgenden Optimierungsmethode genannt - informell, mit Worten und Graphiken beschrieben wird. Das komplette formale Modell findet sich im Anhang.

Eine Optimierungsmethode setzt sich aus der Modellierung des Optimierungsproblems und aus dem angewandten Lösungsverfahren zusammen. Es werden drei Optimierungsmethoden vorgestellt und beschrieben. Sie sind nach ihren Lösungsverfahren - Enumeration, Faktorisierung und Heuristische Wahl der Kombinatorik - benannt. Die Modellierung des Optimierungsproblems wird an jedes der Lösungsverfahren angepasst.

Sowohl die Enumeration als auch die Faktorisierung sind als deterministische Lösungsverfahren zu bezeichnen. Mit ihnen wird entsprechend ihrer Lösungsmenge eindeutig der globale, minimale elektrische Leistungsbedarf bestimmt. Die dritte Methode ist ein stochastisches, heuristisches Lösungsverfahren (siehe Kapitel 3.1). Da nur wenige Zielfunktionswerte mit variierenden Parameter-Permutationen berechnet werden, liefert diese Methode nicht mit Sicherheit den globalen, minimalen elektrischen Wert des Leistungsbedarfs.

Im Folgenden wird kurz darauf eingegangen, wieso diese drei Optimierungsmethoden ausgesucht werden. Ferner werden ihre Arbeitsanweisungen und ihre derzeitigen Grenzen erläutert.

5.1 Modellierung kleiner Luftverteilsysteme – Enumeration der Kombinatorik

Eine vollständige Enumeration ermittelt alle Zielfunktionswerte eines Optimierungsproblems, durch das Einsetzten jeder Parameter-Permutation und bestimmt durch Vergleich dieser den minimalen Wert.

5.1.1 Grund für die Wahl der Enumeration

Im Allgemeinen wird diese Art von Algorithmus zur Lösung von Optimierungsproblemen eingesetzt, für die ein endlicher Lösungsraum definiert wird. Beispielprobleme beruhen auf Entscheidungen mit endlichen Möglichkeiten, weshalb sie als kombinatorische Probleme bezeichnet werden. Damit ist die Voraussetzung für die Anwendung auf das diskret-kombinatorische Optimierungsproblem gegeben.

Die Enumeration wird gewählt, wenn ein eindeutiges globales Minimum des Optimierungsproblems bestimmt werden soll. Dieses Minimum dient unter gleichen Simulationsbedingungen als Referenz für andere Lösungsverfahren. Bei abweichenden Minima wird eine Begründung für die Abweichung gesucht.

5.1.2 Beschreibung der Modellierung

Bild Bild 5-1 stellt schematisch den Aufbau des formalen Modells der Enumeration dar. Dies soll einen ersten Überblick der aufeinander folgenden Arbeitsanweisungen bieten, welche nachfolgend erläutert werden.

Ziel des Lösungsverfahrens ist es, die optimale Kombination der Eigenschaften und Parameter

der Luftverteilung zu finden, welche zum minimalen Leistungsbedarf der Referenzluftverteilung führt. Die Eigenschaften stellen dabei der Volumenstrom und der Druck der jeweiligen Komponente dar sowie, je nach Komponente, der Wirkungsgrad oder der Öffnungswinkel. Die Parameter sind, wie in Kapitel 4.1.2 erläutert, der Ventilator-Typ in Form des Skalierfaktors, die Ventilator-Drehzahl und die Position der Komponenten.

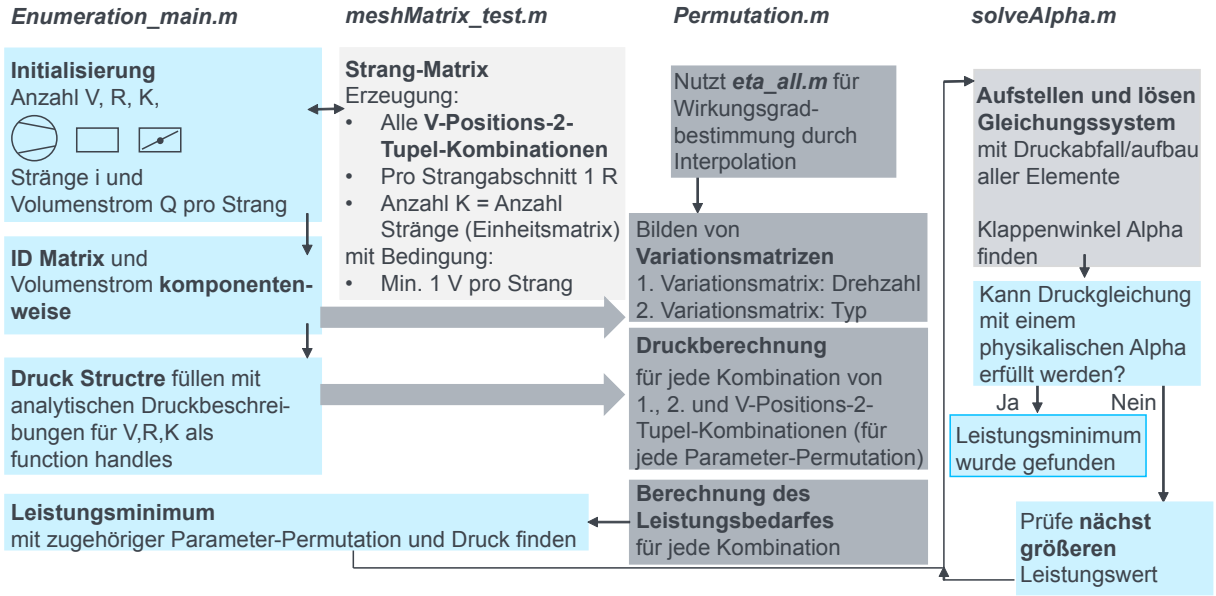


Bild: Bild 5-1 Schematische Darstellung des formalen Modells der Enumeration

Initialisierung Referenzluftverteilung: Um die Referenzluftverteilung abzubilden, ist die Enumeration so aufgebaut, dass zunächst die Anzahl der Stränge ($numMesh = 2$), die Anzahl der Kanalstücke ($numDuct = 3$) und Klappen ($numDamp = 2$) sowie die maximal mögliche Anzahl der Ventilatoren ($numFan = 3$) vorgegeben wird. "Maximal mögliche Anzahl", bedeutet in diesem Zusammenhang, dass die Lösung des Problems auch eine geringere Anzahl an Ventilatoren beinhalten kann. Die Anzahl der Komponenten wird aufsummiert und mit $numEl$ bezeichnet (im gegebenen Problem 8). Außerdem wird für jeden Strang ein konstanter Volumenstrom Q festgelegt.

Strang- und ID-Matrix: Die Position der Komponenten pos (Angabe in Positions-2-Tupel) gibt vor, in welchem Strangabschnitt sie sich befinden. Für jeden Strangabschnitt ist ein Volumenstrom definiert. Der Volumenstrom durch die Komponente wird daher über die Position bestimmt.

Über die Art der Komponente wird eine entsprechende analytische Beschreibung zur Berechnung der Druckdifferenz zugeordnet. Der Druckauf- oder -abbau wird daher über die Art der Komponente bestimmt. Aus diesem Grund muss formal ein eindeutiger Zugriff auf die Position und die Art der einzelnen Komponenten erfolgen können. Dies geschieht mit Hilfe der Aufstellung einer Identitäts(ID)- und einer Strang-Matrix. Die Matrizen und zugehörige Luftverteilung sind beispielhaft in Bild 5-2 dargestellt.

Strang-Matrix

$$\left[\begin{array}{ccc|ccc|cc} 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{array}\right]$$

Strang I

Strang II

ID-Matrix

$$\left[\begin{array}{ccc|ccc|cc} 1 & 1 & 1 & 2 & 2 & 2 & 3 & 3 \end{array}\right]$$

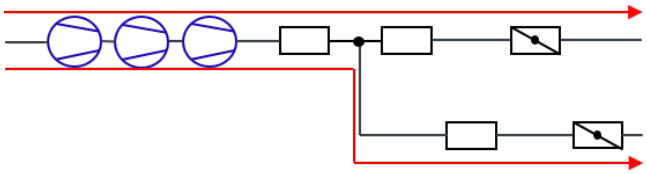


Bild: Bild 5-2 ID- und Strang-Matrix mit zugehöriger schematischer Darstellung der Luftverteilung

Über die Identitäts(ID)-Matrix kann die Art der Komponente festgelegt werden. Mit Hilfe der ID wird ein Ventilator (ID = 1), ein Kanalstück (ID = 2) oder eine Klappe (ID = 3) ausgewiesen. Jeder eingesetzten Komponente wird genau eine ID zugewiesen. Der Wert der ID kann sich wiederholen. Die Strang-Matrix gibt mit Hilfe der Positions-2-Tupel an, in welchem Strangabschnitt sich die Komponenten befinden. Ist die Komponente in beiden Strängen, bzw. in Strangabschnitt I vorhanden, erhält die Komponente, genau wie in Kapitel 4.1.2 beschrieben, den Eintrag $\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]$. Ist diese nur in einem der Stränge vorhanden erhält sie den Eintrag $\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$ oder $\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$. Fehlt die Komponente, erhält sie den Eintrag $\left[\begin{smallmatrix}0\\0\end{smallmatrix}\right]$. Mit letzterem Eintrag kann die Anzahl variiert werden, mit den übrigen die Position. Die Strang-Matrix enthält damit die dargestellten Einträge und entspricht der Größe *Anzahl-der-Stränge x maximale-Anzahl-der-Komponenten*.

Ausschließlich die blau markierten Einträge der Ventilatoren werden hier variiert. Die Komponenten sind in der Strang-Matrix entsprechend der ID-Matrix angeordnet. So stehen die Einträge für die Ventilatoren auf der linken Seite, die Einträge für die Kanalstücke mittig und die Einträge für die Klappen auf der rechten Seite. Da festgelegt wird, dass genau ein Kanalstück pro Strangabschnitt vorhanden ist, bleiben die mittleren Einträge konstant. Für die Klappen gilt: Die Einträge der rechten Seite werden als Einheits-Matrix festgelegt.

Mit Gleichung 4-10 zur Berechnung der Anzahl der Positions-Tupel-Kombinationen ($num\pi_P$) und unter Berücksichtigung, dass mindestens ein Ventilator pro Strang und einer bis alle pro Strangabschnitt eingesetzt werden müssen, entstehen unter Anwendung dieser Methode 13 verschiedene Strang-Matrizen, Bild 5-2 zeigt eine davon. Im Anhang sind die übrigen zwölf zu finden (Bild Bild 10-2).

Mit dem Laufindex i werden die Stränge der Strang-Matrix von eins bis zur Gesamtzahl der Stränge ($numMesh$) durchnummeriert. Laufindex j dient der Nummerierung der Komponenten der Strang-Matrix von eins bis zur Anzahl der Komponenten ($numEl$). So kann im nächsten Schritt für jede Strang-Matrix, auf Basis der festgelegten konstanten Volumenströme im Strang, jeder Komponente ein Volumenstrom zugewiesen werden. Dabei werden die beiden Strangvolumenströme für Strang-Matrix-Einträge von $\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]$ entsprechend der Knotenregel (siehe Kapitel 3.2.2) aufsummiert.

Erläuterung zu MATLAB Datentypen: Innerhalb einer Strang-Matrix (mit dem Laufindex $num\pi_P$) wird jeder Komponente (mit den Laufindizes i, j) strangweise eine analytische Beschreibung der Druckdifferenz zugewiesen. Für die Zuweisung wird der MATLAB-Datentypen *Structure Array* verwendet, welcher an dieser Stelle kurz erläutert wird. Übliche MATLAB-Datentypen, wie ein *Array* oder eine *Matrix*, speichern Informationen oder Daten gleichen Datentyps (z.B. des Datentyps double) und gleicher Datenlänge. Die MATLAB-Datentypen *Structure Array* und

Cell Array werden dazu verwendet, verschiedene Datentypen und -längen in einer einzigen Variablen abzuspeichern. Das Nutzen des *Structure Arrays* hat im Vergleich zum *Cell Array* den Vorteil, dass über den Namen auf die Daten zugegriffen werden kann. Im *Cell Array* geschieht dies ausschließlich über numerische Indizes. Ein *function handle* ist eine Variable, mit welcher eine Funktion indirekt aufgerufen werden kann. Der Funktionswert kann mit dem Variablen-Namen und den einzugebenden Parameter berechnet werden. Wird ein *function handle* in einem *Cell Array* gespeichert, kann nicht gleichzeitig auf die Daten im *Cell Array* zugegriffen und die Parameter des *function handles* eingegeben werden. Ist das erforderlich, so kann ein *Structure Array* genutzt werden [**MathWorks**].

Im Folgenden wird außerdem der Datentyp *Symbolic* erwähnt. Dieser wird genutzt, um mit symbolischen Variablen Berechnungen anstellen zu können. So wird bisher vor der Ausführung einer Rechenoperation jeder Variablen ein numerischer Wert zugewiesen, welcher den Speicherplatz der Variablen einnimmt. Mit dem Datentyp *Symbolic* ist es möglich, den Speicherplatz mit einen algebraischen Wert zu belegen. Dadurch können analytische Gleichungssysteme symbolisch umgestellt und vereinfacht werden, bevor den Variablen numerische Werte zugewiesen werden. Eine Gleichung der Form $y = a * x + b$ kann beispielsweise nach der symbolischen Variablen a aufgelöst werden, wobei die Variablen y, x und b Zahlenwerte annehmen können [**MathWorks**] [**TutorialsPoint**].

Berechnung aller Leistungsbedarfe: Ein Druck-*Structure* der Größe *Anzahl-der-Stränge x Anzahl-der-Komponenten x Anzahl-der-Strang-Matrizen* wird erstellt. Dieses beinhaltet für jede der 13 Strang-Matrizen die analytischen Beschreibungen entsprechend ihrer ID (Gleichungen 4-1, 4-5, 4-6) in Form von *function handles*. Dabei wird das *function handle* nur zugewiesen, wenn der Eintrag der Strang-Matrix 1 ist. Ansonsten bleibt das *Structure*-Feld leer, um keinen weiteren Speicherplatz zu beanspruchen.

Mit der programmierten Funktion *permutation.m* werden alle Leistungsbedarfe berechnet, welche für dieses Optimierungsproblem auftreten können. Dazu werden zunächst Matrizen erstellt, welche für jede Drehzahl- und Ventilator-Typ-Variation einen Eintrag für jeden Ventilator beinhalten. Die Matrizen entsprechen somit der Größe Anzahl-der-Variationen x Anzahl-der-Ventilatoren. Nach Gleichung 4-7 entstehen 1.331 Drehzahl-Variationen und nach Gleichung 4-8 64 Typ-Variationen.

Im Anschluss werden für jeden Ventilator alle Parameter - formal in Schleifen - miteinander kombiniert. Innerhalb jeder der 13 Strang-Matrizen $num\pi_P$ wird jede der 64 Typ-Variationen $num\pi_T$ und innerhalb dieser jede der 1.331 Drehzahl-Variationen $num\pi_D$ genutzt, um in jedem der zwei Stränge i den Druckaufbau, den Wirkungsgrad und den Leistungsbedarf jedes Ventilators j zu berechnen. Es entstehen zwei 4-dimensionale Matrizen - Wirkungsgrad und Leistungsbedarf mit der Größe j x $num\pi_D$ x $num\pi_T$ x $num\pi_P$ - und eine 5-dimensionale Matrix - Druckaufbau mit der Größe j x i x $num\pi_D$ x $num\pi_T$ x $num\pi_P$. So wird sicher gestellt, dass jede Parameter-Permutation genau einmal vorkommt und dass eindeutig auf jede dieser zugegriffen werden kann.

Der Druckaufbau wird mit Hilfe der Gleichung 4-1 des Druck *Structures* berechnet. Dabei wird der Druck für einen Ventilator des Positions-2-Tupels $\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]$ doppelt berechnet. Der Grund dafür liegt im Druck *Strcture*. Das erwähnte *function handle* kommt in beiden Strängen vor und wird für beide Stränge mit dem gesamten Volumenstrom des Ventilators berechnet. Dies ist jedoch notwendig, damit der Druckaufbau der Ventilatoren der Positions-2-Tupel $\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$ oder $\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$ zuge-

wiesen werden kann. Bei der Berechnung des Wirkungsgrades und des Leistungsbedarfes wird dies berücksichtigt. Sie werden mit dem einfachen Druck ermittelt und beziehen sich ausschließlich auf den Ventilator - nicht auf jeden Strang - wodurch die Reduktion der 5D-Matrix auf die 4D-Matrizen entsteht. Für kleine Drehzahlen kann Gleichung 4-1 abhängig vom Volumenstrom negativ werden. Negative Drücke werden an dieser Stelle formal auf den Wert null gesetzt, da sie nicht weiter berücksichtigt werden sollen.

Für jeden berechneten Druckaufbau in einem Ventilator und den zugehörigen Volumenstrom wird der Wirkungsgrad mit der in Kapitel 4.1.2 beschriebenen 2-dimensionalen Interpolationsfunktion berechnet. Ein Wirkungsgrad wird nur dann gefunden, wenn der Druckaufbau innerhalb der Grenzen des minimalen und des maximalen Druckaufbaus liegt. Letzterer ist von dem Ventilator-Typ abhängig (siehe Bild Bild 4-4). Liegt der Druckaufbau außerhalb der Grenzen, nimmt η den Wert *Null* an. Da der Wirkungsgrad eines Ventilators bei jedem Punkt (Druckaufbau|Volumenstrom) einen anderen Wert annimmt, muss der zugehörige Leistungsbedarf für jeden Ventilator nach Gleichung 4-2 separat bestimmt werden. Erst im Anschluss werden die Leistungsbedarfe entsprechend ihrer Parameter-Permutation nach Gleichung 4-3 aufsummiert. Eine 3-dimensionale Matrix des kombinierten Leistungsbedarfes entsteht ($num\pi_D$ x $num\pi_T$ x $num\pi_P$). Da ein Druckaufbau mit einer aufgebrachten Leistung kleiner als null Watt oder genau null Watt physikalisch nicht möglich ist, werden diese als unzulässige Zahl (NaN) deklariert. So wird nach der Summation sicher gestellt, dass ausschließlich positive Leistungsbedarfe berücksichtigt werden. Aus diesen Werten wird durch Vergleich der minimale Wert - die Lösung des Optimierungsproblems - bestimmt.

Strangweise Prüfung der Druckgleichung und gültige Lösung: Im nächsten Schritt wird geprüft, ob mit dieser Lösung die Druckgleichung 3-6 für beide Stränge gleichzeitig eingehalten werden kann (siehe Bild Bild 5-3). Beide Gleichungen werden dabei mit Hilfe des Druck *Structures* in der Funktion solveAlpha.m aufgestellt. Es entsteht ein Gleichungssystem des MATLAB Datentyps $Symbolic$ mit zwei Gleichungen und zwei Unbekannten (Klappen-Öffnungswinkel α_I des Stranges I und α_{II} des Stranges II). Sobald der Druckaufbau der Lösung Δp_t in beiden Strängen gleichzeitig ausreicht, um mindestens den Druckabfall der Kanalstücke $\sum \Delta p_{v,R}$ zu kompensieren, ist die Lösung für diese Anwendung gültig. Reicht der Druckaufbau nicht aus, wird der Leistungsbedarf der Lösung als keine Zahl (NaN) deklariert und die nächstgrößere Lösung wird bilanziert und auf Gültigkeit geprüft.

Bild Bild 5-3 zeigt, dass sobald in Strang I und Strang II unterschiedliche Volumenströme erforderlich sind, in demjenigen mit geringerem Volumenstrom (hier Strang II) ein weiterer Druckabfall ($\Delta p_{v,K}$) erfolgen muss, um die Druckbilanz einhalten zu können. Dieser wird durch den Einsatz einer Klappe in Abhängigkeit ihres Öffnungswinkels erzeugt. Ist die Klappe nicht erforderlich (hier in Strang I), beträgt der Öffnungswinkel $90°$. Ist die Klappe wie in Strang II erforderlich, kann der Öffnungswinkel, wie in Bild Bild 4-6 gezeigt, einen Wert zwischen $0°$ und $90°$ annehmen. Der genaue Winkel wird mit dem symbolischen Gleichungssystem bestimmt.

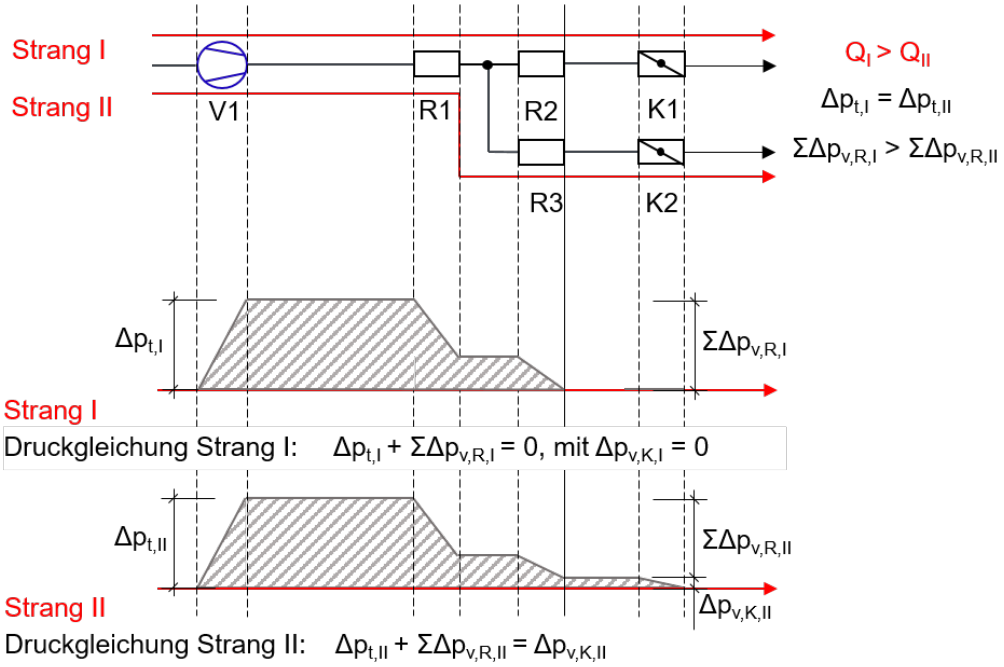


Bild: Bild 5-3 Schematische Darstellung einer Strang-Matrix mit zwei Strängen unterschiedlichen Volumenstroms und zugehöriger Druckgleichung

Aufgrund der geringen Abtastrate der Drehzahleinstellung des Ventilators (Schrittweite = 10%), wird die Druckgleichung auch in Strang I des Beispiels in Bild Bild 5-3 nie null. Dieser Zustand - Druckaufbau ist genau gleich dem Druckabfall - kann nur für bestimmte Volumenströme erreicht werden (siehe Bild Bild 5-4, Schnittpunkte der roten Linie mit den blauen). Daher erhalten die meisten Lösungen der Enumeration zwei Klappen und geben damit nicht den absolut minimalen Leistungsbedarf an. Jede Lösung der Enumeration hat als Ergebnis jedoch den absolut minimalen Leistungsbedarf innerhalb der Grenzen der allgemeinen Form des Optimierungsproblems.

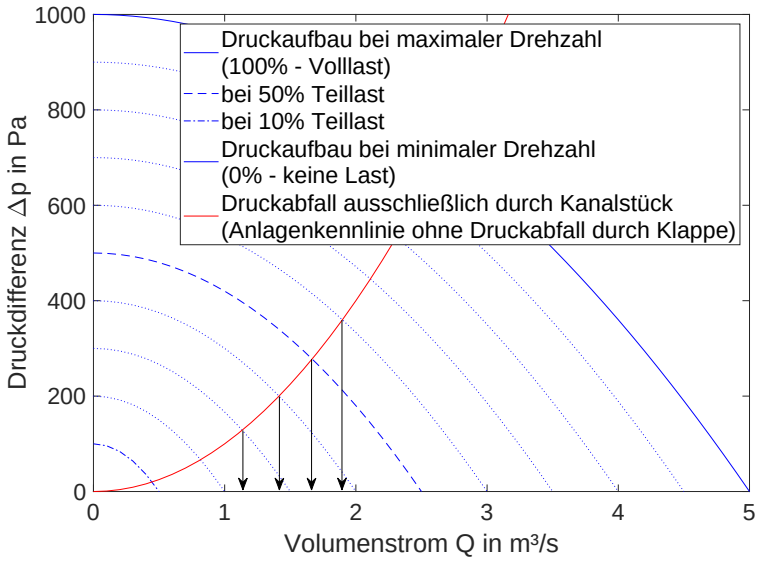


Bild: Bild 5-4 Abbildung der Abtastrate der Drehzahleinstellung des Ventilators von 10% und zugehöriger Druck über variablen Volumenströmen in blau. Abbildung der Anlagenkennlinie ohne Einsatz von Klappen (ausschließlich Kanalstücke) in rot.

Der gesamte in diesem Kapitel beschriebene Vorgang wird für jede konstante Volumenstromangabe (Angabe des Volumenstroms in Strang I und in Strang II) wiederholt.

5.1.3 Grenzen der Modellierung

Die getestete Enumeration kann Luftverteilungen mit:

- genau zwei Strängen
- maximal ein bis drei Ventilatoren
- vier verschiedenen zur Auswahl stehenden Ventilator-Typen entsprechend der Beschreibung in Kapitel 4.1.1
- Drehzahleinstellungen mit einer maximalen Abtastrate von 10%
- keiner Klappe oder genau einer Klappe pro Strang
- einem Klappen-Typ mit variablen Öffnungswinkeln α zwischen $0°$ und $90°$
- null bis einer nicht definierten, konstanten Anzahl an Rohrstücken eines Typs

abbilden.

Mit der beschriebenen Modellierung dieser Methode werden große Matrizen benötigt, um jede Parameter-Permutation, den Volumenstrom und die Druckdifferenz pro Strang und Komponente eindeutig zuweisen zu können. Je mehr Parameter-Permutationen möglich sind, desto größer werden diese Matrizen. Der genutzte Speicherplatz einer Matrix ist abhängig von deren Größe. Ab einer bestimmten Größe (abrufbar mit dem Befehl *(*memory)) erreicht MATLAB in Abhängigkeit des zugewiesenen physikalischen Arbeitsspeichers (RAM) des Computers sein maximales Speichervermögen [**MathWorks**]. Das Programm ist nicht weiter ausführbar und bricht ab.

Die Anzahl an Parameter-Permutationen steigt mit der Anzahl an Strängen (erhöhte Anzahl an Positions-x-Tupel), mit der Anzahl an Typ- und Drehzahl-Variationen, mit der Anzahl an Freiheitsgraden (beispielsweise wenn Anzahl und Position von Klappen und Kanalstücken variabel sind, da die Anzahl an Positions-x-Tupel erhöht wird) oder mit einer größeren Anzahl an Ventilatoren. Wird einer dieser Faktoren verringert, kann ein anderer erhöht werden. Die oben aufgelisteten Maximalwerte der getesteten Enumeration gelten, wenn alle gleichzeitig maximal ausgeschöpft werden.

Mit diesem Setup kann die Abtastrate der Drehzahleinstellung mit dem genutzten Computer und der getesteten Luftverteilung nicht weiter verfeinert werden, da die Anzahl der Drehzahl-Variation mit dem Faktor $\left(\frac{100}{prozentualeAbtastrate}+1\right)^3$ steigt. Für eine Abtastrate von 1 % folgt daraus die nahezu achthundertfache Anzahl der Drehzahl-Variationen (von $numn$ = 11, $numFan$ = 3; $numn^{numFan}$ = 1.331, zu $numn$ = 101, $numFan$ = 3; $numn^{numFan}$ = 1.030.301). Die Größe der Matrizen und die zugehörigen auszuführenden elementaren Rechenoperatoren übersteigen bei der derzeit bestehenden Modellierung das maximale Speichervermögen.

Für eine Erhöhung der Stranganzahl muss außerdem berücksichtigt werden, dass die Strang-Matrix mit den Positions-2-Tupeln ($pos = \begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}$) erzeugt wird. Soll diese Anzahl auf x Stränge erhöht werden, muss neben der Anzahl der Positions-Tupel-Kombinationen außerdem die Aufstellung der Positions-x-Tupel verändert werden.

Für $x = 3$ gilt $pos = \begin{bmatrix}1\\1\\1\end{bmatrix}, \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}1\\0\\1\end{bmatrix}, \begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}$.

Es entstehen 71 statt 13 Strang-Matrizen. Bleiben die weiteren Angaben der getesteten Luftverteilung gleich, können drei Stränge mit der bestehenden Modellierung simuliert werden. Die

Programm-Laufzeit erhöht sich jedoch von mehreren Stunden auf mehrere Tage. Bild Bild 5-5 zeigt den Anstieg der Programm-Laufzeit in Abhängigkeit der Modellgröße.

Hierbei entspricht die Laufzeit nicht der Definition in Kapitel 4.1.2, sondern einer rechnerspezifischen Programm-Laufzeit, welche mit der MATLAB-Funktion $tic - toc$ gemessen wird. Alle Simulationen werden dabei an demselben Rechner durchgeführt. Es finden keine parallelen Simulationen statt. Die Qualität der Programmierung wird hier mit berücksichtigt.

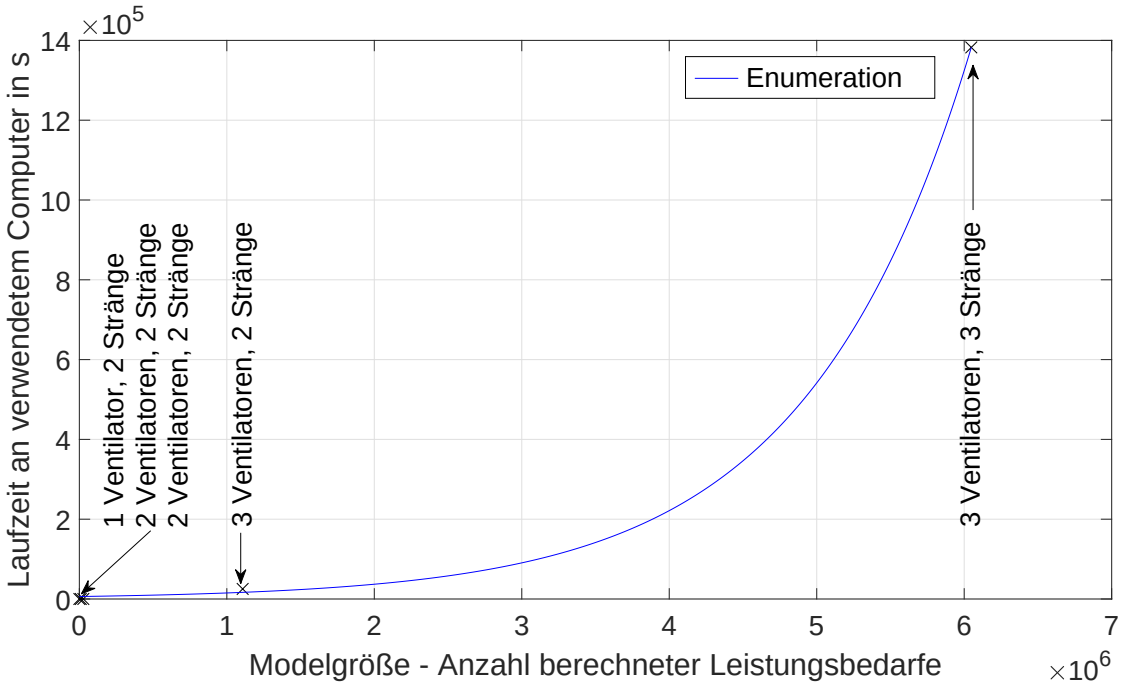


Bild: Bild 5-5 Schematische Darstellung des überproportionalen Anstiegs der Programm-Laufzeit mit steigender Modellgröße

Es wird ersichtlich, dass das betrachtete diskret-kombinatorische Optimierungsproblem der Komplexitätsklasse NP zugewiesen werden muss. Die Enumeration ist bereits für sehr kleine Netzgeometrien begrenzt, weshalb für größere Luftverteilsysteme alternative Optimierungsmethoden notwendig sind. Bild Bild 5-5 dient also dazu, eine erste Abschätzung treffen zu können, ab welcher Netzgeometrie ein Lösungsverfahren eingesetzt werden muss, welches nicht jeden Zielfunktionswert ermittelt, sondern auf eine andere Herangehensweise beziehungsweise auf die Statistik/Heuristik zurückgreift.

5.2 Modellierung kleiner Luftverteilsysteme – Faktorisierung der Kombinatorik

Die Faktorisierung beschreibt in der Mathematik einen Vorgang, bei welchem ein Objekt in mehrere Faktoren zerlegt wird. Ein Datenverarbeitungsverfahren für NP schwere kombinatorische Optimierungsprobleme ist beispielsweise durch die Matrix-Faktorisierung gegeben. Vor der eigentlichen Datenverarbeitung, bei der Lösungen ermittelt werden, findet eine Zerlegung großer Datenmatrizen in mehrere kleine Matrizen statt. Durch a priori (von vornherein bekanntem) Wissen wird jeder Eintrag der kleinen Matrizen nach Relevanz gewichtet. Nur Einträge höchster Relevanz werden zur eigentlichen Datenverarbeitung verwendet. Somit können dieselben Informationen unter Verwendung von wesentlich kleineren Datenmengen beibehalten werden. Die Dimensionalität und der Speicherplatz der Datenmatrizen wird reduziert [**UniversitatBremen**] [**IBMKnowledgeCenter**].

Im Rahmen der vorliegenden Arbeit werden zwar keine klassischen Verfahren der Matrix-Faktorisierung angewendet, angelehnt an deren Logik wird jedoch eine Faktorisierung vorgenommen, indem an jedem Punkt ($\sum_1^{numFan} \Delta p|Q$) aus allen Zielfunktionswerten eines kombinatorischen Ventilator-Kennfeldes (große Datenmatrizen) ausschließlich die Lösung (minimaler Zielfunktionswert und Parameter-Permutation, kleinere Matrizen) gespeichert wird. Die relevante Information - im Weiteren Lösungs-Kennfeld genannt - wird damit beibehalten, nicht weiter benötigte Informationen werden gelöscht. Sobald das Lösungs-Kennfeld vorliegt, kann im nächsten Schritt die Lösung für Luftverteilungen unterschiedlicher Druckabfälle und Volumenströme mit geringem Rechenaufwand bestimmt werden. Das Optimierungsproblem wird an dieses Lösungsverfahren angepasst.

5.2.1 Grund für die Wahl der Faktorisierung

Während die Enumeration, wie in Kapitel 5.1.3 erläutert, schon für geometrisch sehr kleine Luftverteilungen bzw. geringe Freiheitsgrade an die Grenzen des zugewiesenen physikalischen Arbeitsspeichers stößt, kann die Faktorisierung ein Lösungsverfahren für größere Luftverteilungen bieten.

Das Lösungs-Kennfeld wird dabei genutzt, um sowohl die Drehzahl-Variationen (in Form von Druck-Variationen), als auch die Ventilator-Typ-Variationen bereits im Vorfeld zu bestimmen. Die Berechnung des Lösungs-Kennfeldes soll nur einmalig erfolgen und für jegliche Luftverteilungen - der gleichen zur Auswahl stehenden Parameter - anwendbar sein. So ist beispielsweise kein fester Strangvolumenstrom für ein spezifisches Anwendungsbeispiel vorgegeben. Die Position-Tupel-Kombination wird im Lösungs-Kennfeld außen vor gelassen. Im Folgenden wird zunächst die Modellierung und im Anschluss die Grenzen dieser beschrieben.

In einer erweiterten Teilmodellierung, welche nicht mehr Teil dieser Arbeit ist, sollen die Strang-Matrizen, genau wie die der Enumeration variieren. Die Lösung zu jeder Strang-Matrix soll aus den Lösungen des Lösungs-Kennfeldes abgelesen werden. So soll das Optimierungsproblem, welches in der erweiterten Modellierung bearbeitet wird, um die Freiheitsgrade Dimensionierung und Arbeitsbereich der Ventilatoren (9 und 12) reduziert werden, um Arbeitsspeicherplatz einzusparen und damit für größere Luftverteilungen anwendbar zu sein. Überlegungen zur Umsetzung dieser Teilmodellierung sind in Kapitel 8 zu finden.

5.2.2 Beschreibung der Modellierung

Während in der Enumeration für einen festgelegten Volumenstrom die Lösung der Referenzluftverteilung gefunden werden soll, ist das Ziel der Faktorisierung die Lösung für mehrere verschiedene Volumenströme zu erhalten. Die Position der Ventilatoren (Freiheitsgrad 6) bleibt dabei zunächst unberücksichtigt. Die Anzahl kann jedoch weiterhin variieren. Für den in diesem Kapitel beschriebenen Teil der Faktorisierung erfolgen weitere Einschränkungen: Die Netzgeometrie (Strangabschnitt und Strang) kann noch nicht abgebildet werden und die Komponenten Klappe und Kanalstück bleiben zunächst unberücksichtigt.

Die Erstellung des Lösungs-Kennfeldes erfolgt mit Hilfe der schematisch dargestellten, aufeinanderfolgenden Arbeitsanweisungen in Bild Bild 5-6. Nachfolgend wird auf die Reihenfolge der Anweisungen eingegangen.

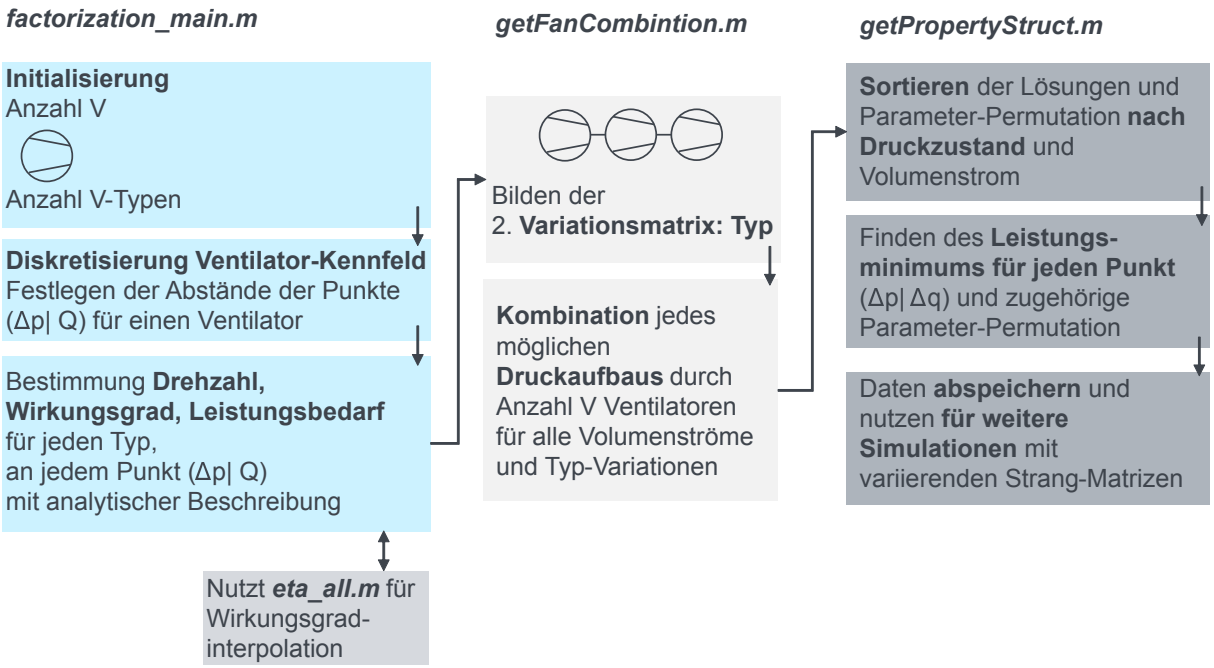


Bild: Bild 5-6 Schematische Darstellung des formalen Modells der Faktorisierung.

Erstellung des Lösungs-Kennfeldes: Das obere Kennfeld in Bild Bild 5-7 bildet das diskretisierte Kennfeld eines einzelnen Ventilators ab. Diskretisierung bedeutet in diesem Zusammenhang das Einteilen eines Ventilator-Kennfeldes in endlich große Kontrollkennfelder, der Breiten $\Delta q = Q_{i+1} - Q_i$ und Höhen $\Delta p_d = \Delta p_{t,j+1} - \Delta p_{t,j}$. Die hier blau oder rot gefärbten Knotenpunkte der Kontrollfelder werden mit $(\Delta p_{t,j}|Q_i)$ bezeichnet. So steht die Diskretisierung (Δp = 20 Pa, Δq = 0,5 $m^3\,s^{-1}$) für Ventilator-Typ 1 mit einem maximalen Druckaufbau von 1,000 Pa und einem maximalen Volumenstrom von 5,0 $m^3\,s^{-1}$ beispielsweise für 51 (1.000/20+1) Teildrücke und deren Abbildung auf 10 (5/0,5) Teilvolumenströme. Mit dem Summand "+1", der Teildruckberechnung, wird berücksichtigt, dass der Druck jedes Ventilators den Wert Null annehmen kann. Dadurch wird die Anzahl der Ventilatoren verändert, indem idealisiert angenommen wird, dass der Ventilator bei einem Druck von 0 Pa nicht vorhanden ist. Es entstehen 510 Knotenpunkte. Die roten beschreiben dabei die physikalisch unzulässigen Werte, da eine Drehzahleinstellung von 100% überschritten wird. Die blauen sind zulässig.

Für jeden Knotenpunkt wird im ersten Schritt der Faktorisierung sowohl der Wirkungsgrad (mit Hilfe der 2D-Interpolation) als auch der Leistungsbedarf und die Drehzahleinstellung berechnet. Letztere wird mit der inversen Funktion von Gleichung 4-1 bestimmt. Die Drehzahleinstellung ist somit nicht auf die angegebenen Lösungsmengen (siehe 4.1.2) begrenzt, sondern durch die Diskretisierung des Druckaufbaus. Der Leistungsbedarf wird nach Gleichung 4-2 mit gewählten Δp_t, welcher unabhängig von Drehzahl, Typ und Volumenstrom n, tp, Q ist, berechnet. Wird eine Drehzahleinstellung von über 100 % erreicht, wird der zugehörige Druck, der Wirkungsgrad und der Leistungsbedarf als unzulässige Zahl (NaN) definiert. Somit werden unphysikalische Werte ausgeschlossen. Dieser Vorgang wird für jeden zur Auswahl stehenden Ventilator-Typ wiederholt.

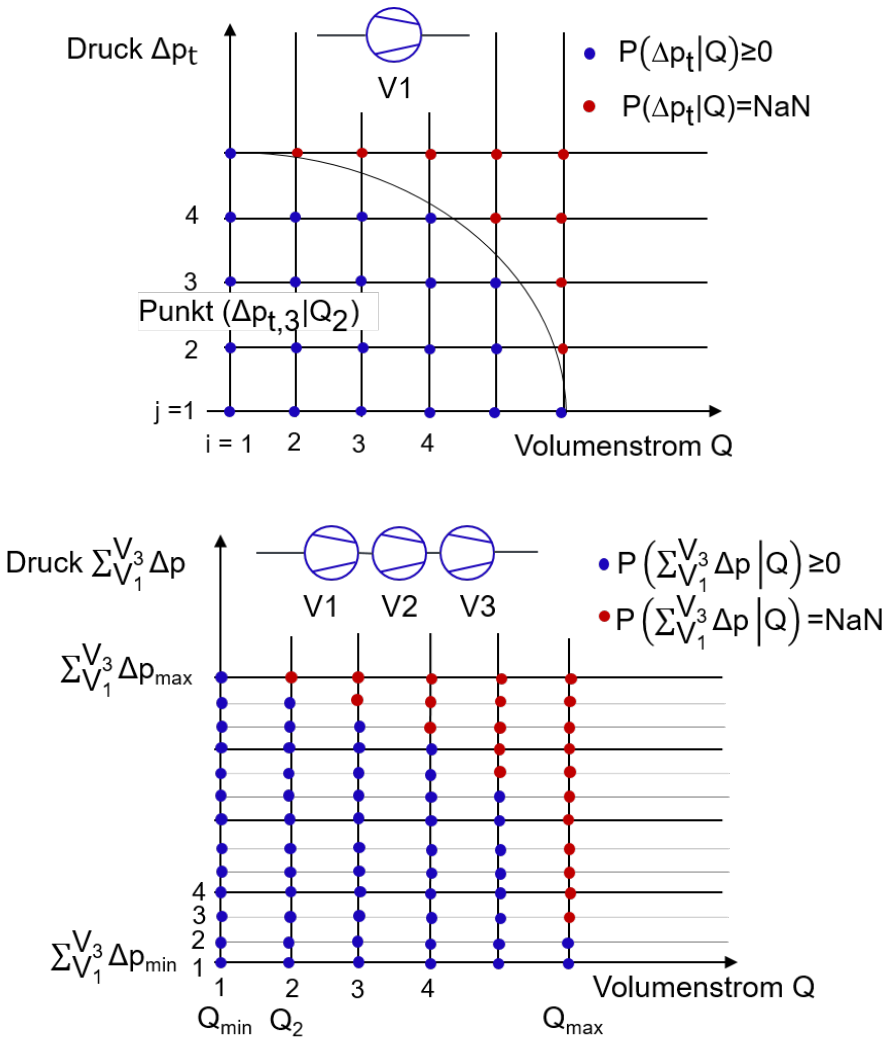


Bild: Bild 5-7 Schematische Darstellung der Diskretisierung des Kennfeldes eines Ventilators (oben) sowie der des Lösungs-Kennfeldes (unten)

Analog zur Enumeration werden im nächsten Schritt (in Funktion *getFanCombination.m*) drei (bzw. $numFan$) Ventilatoren $V_{1,\ldots,numFan}$ miteinander kombiniert. Statt der Drehzahleinstellungen $n_{1,\ldots,numFan}$ werden bei der Modellierung der Faktorisierung die aufgebauten Drücke $\Delta p_{1,\ldots,numFan}$ für jeden Teilvolumenstrom variiert und kombiniert. So erfolgt die Summation aller Teildrücke nach den Druck-Variationen π_{Druck} (siehe Tabelle in Bild Bild 5-8), statt wie bei der Enumeration die der Drücke aus den Drehzahl-Variationen (vgl. Bild Bild 4-9 Tabelle c). Unten in Bild Bild 5-7 ist das Lösungs-Kennfeld mit summierten, kombinierten Teildrücken abgebildet. Dieser Vorgang wird für jede Ventilator-Typ-Variation wiederholt.

Gesamtdruck-aufbau ΣΔp	Ventilator 1 bis numFan		
	V1	V2	V3
	0 Pa	0 Pa	0 Pa
	0 Pa	0 Pa	10 Pa
...	...	...	...
$\Sigma\Delta p_{max}$	1000	1000	1000

Bild: Bild 5-8 Variation des Druckaufbaus der Faktorisierung schematisch in Matrizenform

Die Anzahl der Druck-Variationen kann äquivalent zu Gleichung 4-7 mit

$$num\pi_{Druck} = \left(\frac{\Delta p_{t,max}}{\Delta p_d} + 1\right)^{numFan} \tag{5-1}$$

$num\pi_{Druck}$ Anzahl der Druck-Variationen

$\Delta p_{t,max}$ Maximaler Druckaufbau durch die Summe aller Ventilatoren in Pa (in der Referenzluftverteilung 3,000 Pa)

Δp_d Höhe bzw. Teildruck eines Kontrollkennfeldes in Pa

$numFan$ Maximale Anzahl der Ventilatoren im System

berechnet werden.

Die Diskretisierung im Volumenstrom- und Druckaufbaubereich bleibt dabei erhalten. Da jedoch die Summe aller Teildrücke berücksichtigt wird, kann ein maximaler Gesamtdruck von 3000 Pa aufgebaut werden. Außerdem kann für verschiedene Druck-Variationen der gleiche Gesamtdruck aufgebaut werden. Nimmt dieser einen Wert von 100 Pa an, kann für drei Ventilatoren die Druck-Variante [V1 = 0 Pa V2 = 100 Pa V3 = 0 Pa] beispielsweise den gleichen gesamten Druck aufbauen, wie die Druck-Variante [V1 = 50 Pa V2 = 0 Pa V3 = 50 Pa] und viele weitere. Alle zugehörigen Leistungsbedarfe werden für jeden Knotenpunkt des Lösungs-Kennfeldes ($\sum_1^{numFan} \Delta p|Q_i$) ebenfalls aufsummiert. Dadurch entstehen mehrere Leistungsbedarfe an einem Knotenpunkt. Aufgrund der verschiedenen Ventilator-Typen und entsprechenden Wirkungsgrade variiert der Wert dieser Leistungsbedarfe jedoch. Matrizen entsprechender Drehzahlen, Wirkungsgrade und Skalierfaktoren der einzelnen Ventilatoren werden für jede Druck- und Ventilator-Typ-Variation abgespeichert.

Lösungen Knotenpunkten zuordnen: In einer weiteren benutzerdefinierten MATLAB Funktion *getPropertyStruct.m* werden alle gespeicherten Matrizen zu den entsprechenden Knotenpunkten des Lösungs-Kennfeldes ($\sum_1^{numFan} \Delta p|Q_i$) zugeordnet. Da mehrere Lösungen an einem Punkt möglich sind, kann so sicher gestellt werden, dass alle Leistungsbedarfe und zugehörige Parameter-Permutationen an jedem Punkt bekannt sind. Dabei werden die Matrizen-Einträge innerhalb jedes Teilvolumenstroms - formal in einer Schleife - nach den Gesamtdrücken sortiert. So wird sichergestellt, dass beispielsweise am Punkt ($\sum_1^{numFan} \Delta p = 50\,\text{Pa}|Q_i = 0{,}5\,\text{m}^3\,\text{s}^{-1}$) alle Leistungsbedarfe - also alle Zusammensetzungen aus Typ, Anzahl und Teillast

der Ventilatoren (Parameter-Permutationen) - bekannt sind, welche genau zu diesem Punkt führen.

Im Anschluss wird an jedem Punkt der minimale Leistungsbedarf und zugehörige Parameter-Permutation bestimmt. Gibt es mehrere Lösungen mit dem gleichen Leistungsbedarf aber unterschiedlichen Parameter-Permutationen, wird nur die erste gefundene Lösung gespeichert. Bild Bild 5-9 zeigt die Veränderung der Anzahl der Leistungsbedarfe pro Knotenpunkt. Ausgehend von allen Leistungsbedarfen - einer pro Parameter-Permutation - pro Knotenpunkt, die nicht den Wert *Null* oder *keine Zahl* annehmen (oben rechts), über die Anzahl an Leistungsbedarfen, die den gleichen minimalen Wert annehmen (unten links), hin zu dem einen minimalen Leistungsbedarf (unten rechts), dessen zugehörige Parameter-Permutation für das Lösungs-Kennfeld gespeichert wird. Oben links werden alle drei Zustände übereinandergelegt dargestellt.

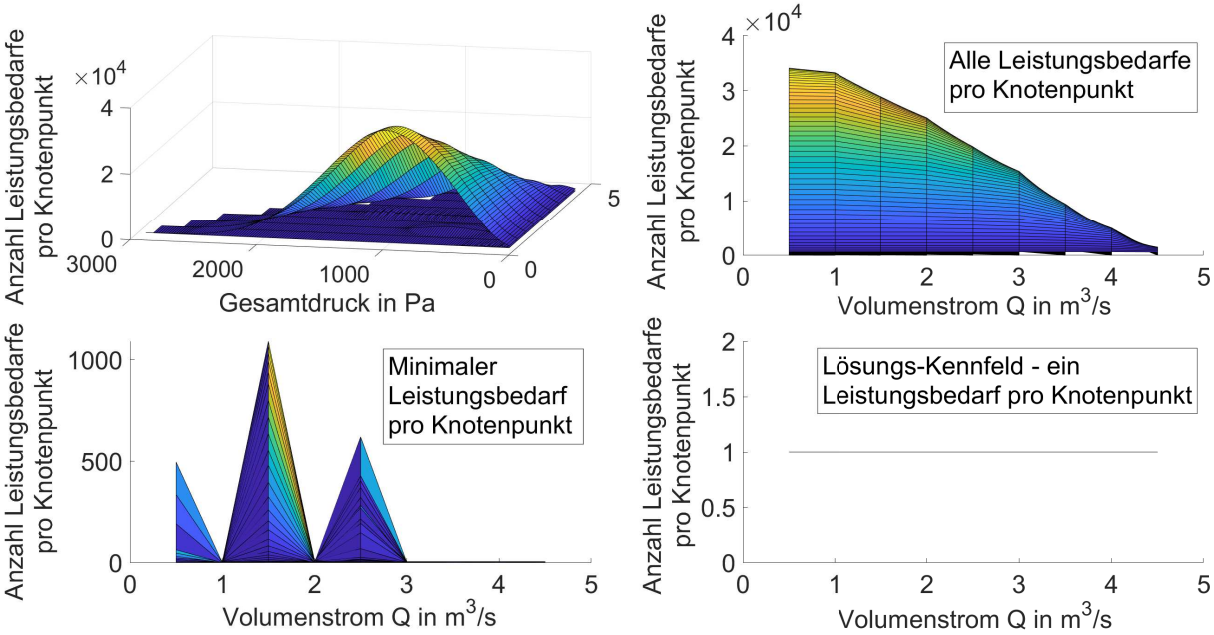


Bild: Bild 5-9 Abbildung des Prozesses: Lösungen den Knotenpunkten zuordnen. Leistungsbedarfe pro Knotenpunkt für eine Diskretisierung von (Δp = 20 Pa, Δq = 0,5 m^3 s^{-1}).

Bild Bild 5-10 bildet das Lösungs-Kennfeld sowie einzelne Kontenpunkte ($\sum_1^{numFan} \Delta p|Q$) mit Lösungs-Parameter-Permutation und Wirkungsgraden beispielhaft für eine Diskretisierung von (Δp_d = 20 Pa, Δq =0,5 m^3 s^{-1}) ab.

Die Erstellung des Lösungs-Kennfeldes ist eine andere Art von Teilenumeration. Durch die intelligente Wahl der Parameter-Permutation, d.h. Druck- statt Drehzahl-Variation, muss diese Teilenumeration nur einmalig durchgeführt werden. Das Ergebnis dieser Teilenumeration kann dann in einer zweiten Teilenumeration, welche in Kapitel 8 schematisch erläutert wird, für jegliche Kanalnetze genutzt werden. Der hier beschriebene Teil der Optimierungsmethode Faktorisierung stellt daher einen Preprocessing-Schritt dar.

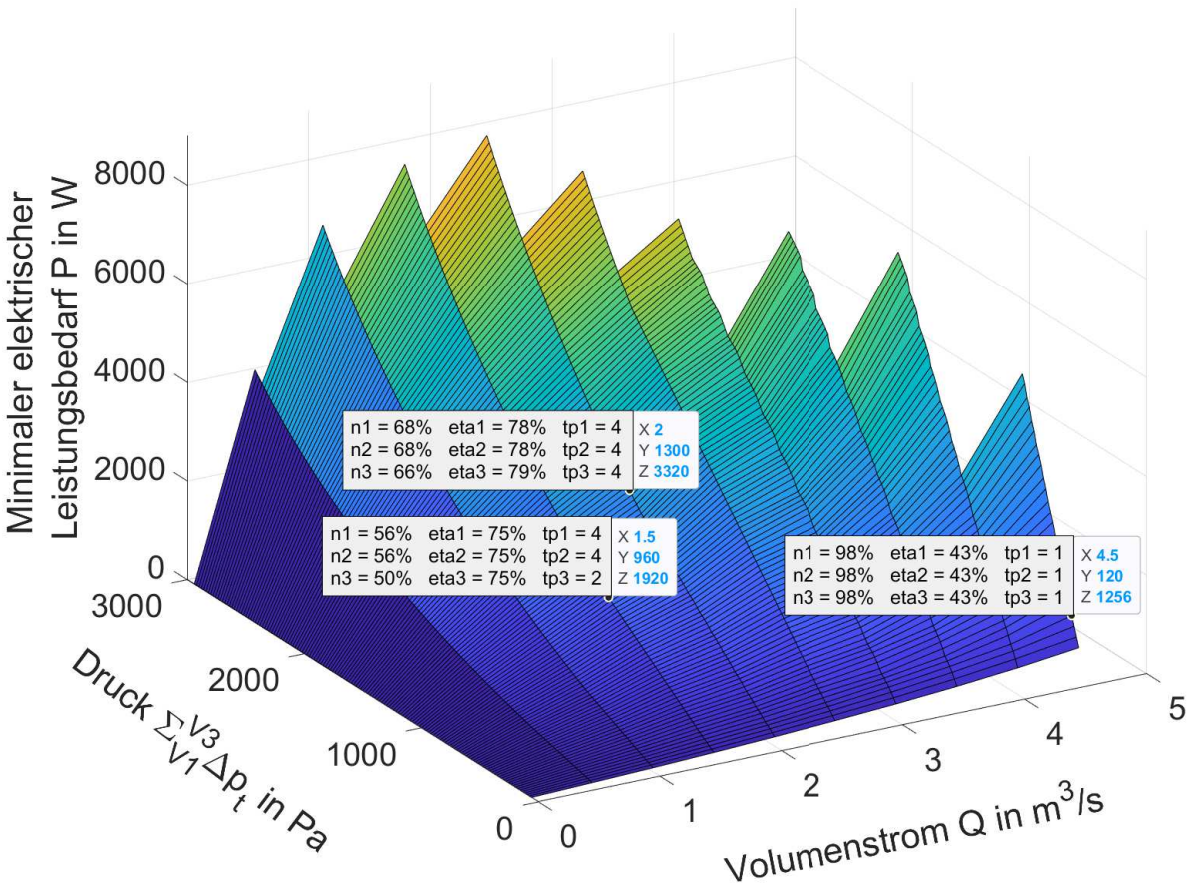


Bild: Bild 5-10 Abbildung des Lösungs-Kennfeldes der simulierten Faktorisierung für eine Diskretisierung von (Δq = 20 Pa, Δp = 0,5 m^3 s^{-1}).

5.2.3 Grenzen der Faktorisierung

Die in dieser Arbeit untersuchte Faktorisierung kann Luftverteilungen mit

- einer Kombination aus maximal ein bis vier Ventilatoren
- vier verschiedenen zur Auswahl stehenden Ventilator-Typen entsprechend der Beschreibung in Kapitel 4.1.1
- einer maximalen Diskretisierung von (Δp_d = 20 Pa, Δq =0,5 m^3 s^{-1})

abbilden.

Die notwendige Zuweisung von Volumenströmen erfolgt, im Vergleich zur Modellierung der Enumeration, mit der hier beschriebenen Modellierung der Faktorisierung nicht strang- und komponentenweise. Der Volumenstrom kann daher nicht auf zwei oder mehrere Stränge aufgeteilt werden, er muss durch die minimale Anzahl bis durch die maximale Anzahl an Ventilatoren strömen. Außerdem kommen aufgrund der fehlenden Prüfung der Druckgleichung nur Ventilatoren, keine Kanalstücke und Klappen zum Einsatz. Die Zuweisung zur ID der Komponenten (Kanalstücke und Klappen) wird damit nichtig. Es kann daher keine Angabe zur Begrenzung der beiden Komponenten oder der Stranganzahl gemacht werden.

Stattdessen werden jedem Teilvolumenstrom mehrere Ventilator-Typ-Variationen und diesen mehrere Druck-Variation zugewiesen. Die Kombination der Teilvolumenströme mit den Va-

riationen stellt für diese Modellierung die Parameter-Permutation dar. So werden trotz unberücksichtigter Position und ID, große Matrizen für eine eindeutige Zuweisung der Parameter-Permutation erzeugt, welche einen großen Speicherplatz benötigen. Erst im letzten Schritt tritt der angestrebte Vorteil dieser Methode ein: Ausschließlich die wichtigen Informationen der großen Datenmatrizen werden zur Weiterverarbeitung zur Verfügung gestellt. In allen voran gehenden Schritten sind hauptsächlich die Anzahl der Teildrücke und -volumenströme limitierende Faktoren.

Analog der Enumeration gilt, dass die angegebenen Maximalwerte der Anzahl an Ventilatoren und Typen sowie der Diskretisierung gelten, wenn alle gleichzeitig maximal ausgeschöpft werden. Werden beispielsweise weniger Teilvolumenströme betrachtet, können Lösungen auch für eine größere Anzahl an Teildrücken bestimmt werden. Dabei ist zu beachten, dass bei einer Erhöhung der Teildruck-Anzahl die Laufzeit des Lösungs-Algorithmus weitaus stärker ansteigt, als für eine Erhöhung der Anzahl der Teilvolumenströme.

5.3 Modellierung kleiner Luftverteilsysteme – Heuristische Wahl der Kombinatorik

Heuristik (griechisch heuriskein) bedeutet finden oder entdecken. In Zusammenhang mit Optimierungsproblemen wird unter Heuristik verstanden, Erfahrungswerte, Annahmen und Faustregeln zu finden, um ein Problem in begrenzter Zeit mit begrenztem Wissen zu lösen [**UniversitaetTuebingen**] [**Stickel**].

Ein genetischer Algorithmus (GA) ist eine zufällige Heuristik, welche Entscheidungsprobleme näherungsweise zu lösen versucht. Er nutzt Mechanismen der natürlichen Evolutionstheorie (Selektion, Reproduktion, Rekombination und Mutation), um die beste Lösung zu finden. In der Regel gibt der Algorithmus weder eine Garantie bezüglich einer Reduktion der benötigten Laufzeit, noch der Güte der ausgegebenen Lösung. Dennoch ist diese Art von Algorithmus für das diskret-kombinatorische Optimierungsproblem geeignet. Dessen Lösung kann jedoch wie in Kapitel 3.1 erläutert, bei jeder Wiederholung des Lösungsverfahrens variieren und um einen unbestimmten Wert vom globalen Minimum abweichen.

5.3.1 Grund für die Wahl des Genetischen Algorithmus

Generell sind exakte, deterministische Verfahren, wie die bisher beschriebenen, nur bei kleinen Optimierungsproblemgrößen anwendbar. Evolutionäre, im Besonderen genetische Algorithmen, sind dagegen gut geeignet für größere kombinatorische Optimierungsprobleme. Unter Verwendung von spezifischem Problemwissen kann deren Lösung effektiv gefunden werden [**UniversitaetTuebingen**].

Der Zielfunktionswert wird mit diesem Algorithmus auf Grund der definierten Anzahl an sogenannten Generationen und deren intelligenten Entwicklung nicht für jede Parameter-Permutation berechnet, sondern nur für eine Teilmenge davon. Wird das Optimum innerhalb dieser Anzahl gefunden, kann die Laufzeit verringert werden. Außerdem erfolgt in einem Schritt die Berechnung der Zielfunktionswerte nur für eine Population (geringe Teilmenge von Problemlösungen). Zugehörige Zwischenergebnisse und der große Speicherplatz, um für weitere Berechnungen eindeutig auf diese zugreifen zu können, werden durch nachfolgend definiertem Gütemaß und Erwartungswert ersetzt. Zwischenergebnisse und Gütemaß werden in jedem Schritt gelöscht. Insgesamt können Matrizen-Größe und der korrelierende benötigte Speicherplatz dadurch we-

sentlich reduziert werden.

Theoretisch kann somit, im Vergleich zur Enumeration und Faktorisierung, eine weitaus größere Anzahl an Strängen, Komponenten und Freiheitsgraden untersucht werden, ohne ein Speicherplatz- oder Laufzeitproblem zu generieren.

5.3.2 Beschreibung der Modellierung

Der GA ist eine von MATLAB zur Verfügung gestellte Funktion, welche zur Berechnung von gemischt-ganzzahligen Optimierungsproblemen eingesetzt wird. Bild Bild 5-11 stellt deren aufeinanderfolgende Arbeitsanweisungen schematisch dar. Die Funktion arbeitet mit einer benutzerdefinierten Zielfunktion, welche genau wie die Faktorisierung zunächst sowohl die Abbildung der Netzgeometrie (Strangabschnitt und Strang) als auch die Betrachtung der Klappen und Kanalstücke unberücksichtigt lässt.

Arbeitsweise des GAs von MATLAB: Dem Algorithmus werden neben der Zielfunktion die zur Berechnung des Zielfunktionswertes benötigten Parameter und deren Zahlenraum nach der allgemeinen Form (siehe 4.1.2) vorgegeben.

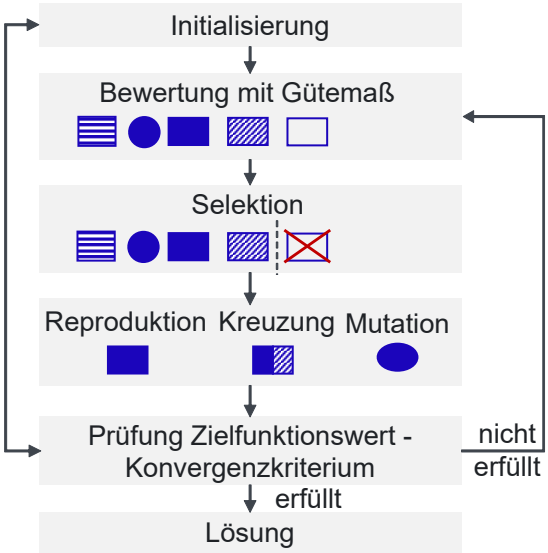


Bild: Bild 5-11 Schematische Darstellung des Vorgehens eines Genetischen Algorithmus [MathWorksb].

Im ersten Schritt initiiert der Algorithmus mit diesen Parametern zunächst willkürlich eine Anfangspopulation mit verschiedenen Individuen. Auf das hier betrachtete Optimierungsproblem angewendet, sind Individuen die verschiedenen Parameter-Permutationen [Auswahl π_D, Auswahl π_T, Auswahl $\pi_{vorhanden}$] = [n_1 n_2 n_3, tp_1 tp_2 tp_3, x_1 x_2 x_3], wobei n die Drehzahl ist, tp der Ventilator-Typ und x ein Indikator, welcher angibt, ob der Ventilator vorhanden ist (1) oder nicht (0). Die Anzahl der Auswahl ($\pi_{vorhanden}$ - $num\pi_{vorhanden}$) kann mit einer *Variation mit Wiederholung* beschrieben werden. Sie wird daher mit Gleichung 3-1 berechnet. Die Auswahl kann auch so erfolgen, dass kein Ventilator im System vorhanden ist. Diese Möglichkeit wird erst mit der Zielfunktion ausgeschlossen. Die Indizes 1 bis 3 stehen für die verschiedenen Ventilatoren. Für jedes Individuum berechnet der Algorithmus zugehörige Zielfunktionswerte. Bild Bild 5-12 bildet eine Beispielanfangspopulation ab.

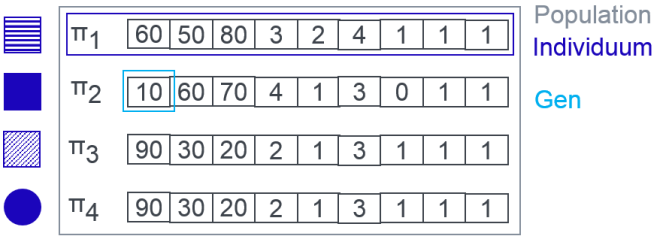


Bild: Bild 5-12 Schematische Darstellung von Population, Individuum und Gen eines Genetischen Algorithmus mit Bezug auf das in dieser Arbeit betrachtete Optimierungsproblem [VijiniMallawaarachchi].

Im nächsten Schritt wird eine Reihe neuer Populationen gebildet. Genau wie für die beiden bisher beschriebenen Optimierungsmethoden, soll aus der großen Lösungsmenge an Parameter-Permutationen diejenige gefunden werden, welche den minimalen Leistungsbedarf erzeugt. Da hier der Zielfunktionswert nicht für jede Parameter-Permutation berechnet wird, wird jeder Parameter-Permutation einer Population (Generation) eine reelle Gütemaßzahl (raw fitness score) zugewiesen. Diese bewertet, wie stark die Parameter-Permutation dazu beigetragen hat, den bisher minimalen Zielfunktionswert erreicht zu haben. Sie wird in einen Erwartungswert (expectation value) umgerechnet; hiermit findet die sogenannte Selektion statt. Je höher der Erwartungswert ist, desto höher ist die Wahrscheinlichkeit, dass diese Parameter-Permutation ein sogenanntes Elternteil (können mehr als zwei sein) oder zur Elite wird.

Allgemein bildet der Algorithmus über die Kombination von drei Mechanismen neue Populationen (sogenannte nächste Generationen), welche üblicherweise Parameter-Permutationen mit höheren Gütemaßzahlen enthalten. Bild Bild 5-13 stellt diese - Reproduktion, Kreuzung, Mutation - schematisch dar. Der erste Mechanismus - *Reproduktion* - gibt Parameter-Permutationen mit den höchsten Erwartungswerten als Elitekinder unverändert an die nächste Generationen weiter. So wird verhindert, dass bereits gefundene gute Lösungen komplett verworfen werden und erneut gefunden werden müssen. Erfolgt eine Rekombination aus zwei gleichen Anteilen der Parameter-Permutationen eines Elternpaars - *Kreuzung* -, werden gekreuzte Kinder gebildet. Mit Hilfe der Eltern - die besten der "Elite" - werden zwei weitere Arten von Kindern gebildet. Wird eines der Elternteile willkürlich verändert - *Mutation* -, werden mutierte Kinder gebildet. Es wird ein Parameterwert - *Gen* - von einer der besten Parameter-Permutationen ausgetauscht. Die neue Population besteht standardisiert aus 5% Elitekindern und 0,8% rekombinierten Kindern. Die übrigen Kinder sind mutiert [**MathWorksb**]. So verbessert der Algorithmus über die Generationen die Population und deren Zielfunktionswerte, bis eines der nachfolgend genannten Abbruchkriterien eintritt. Bild Bild 5-14 stellt die Veränderung des Leistungsbedarfs (Penalty Value in W) im Durchschnitt (mean) und des gefundenen Minimums (best) über verschiedene Generationen für einen Beispielvolumenstrom dar.

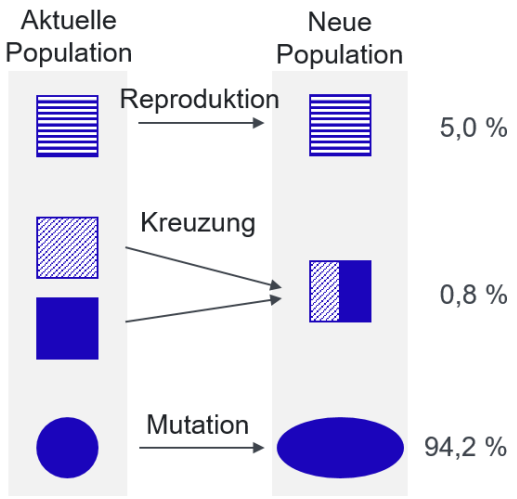


Bild: Bild 5-13 Schematische Darstellung der drei Mechanismen zur Bildung einer neuen Population durch den Genetischen Algorithmus [MathWorksb].

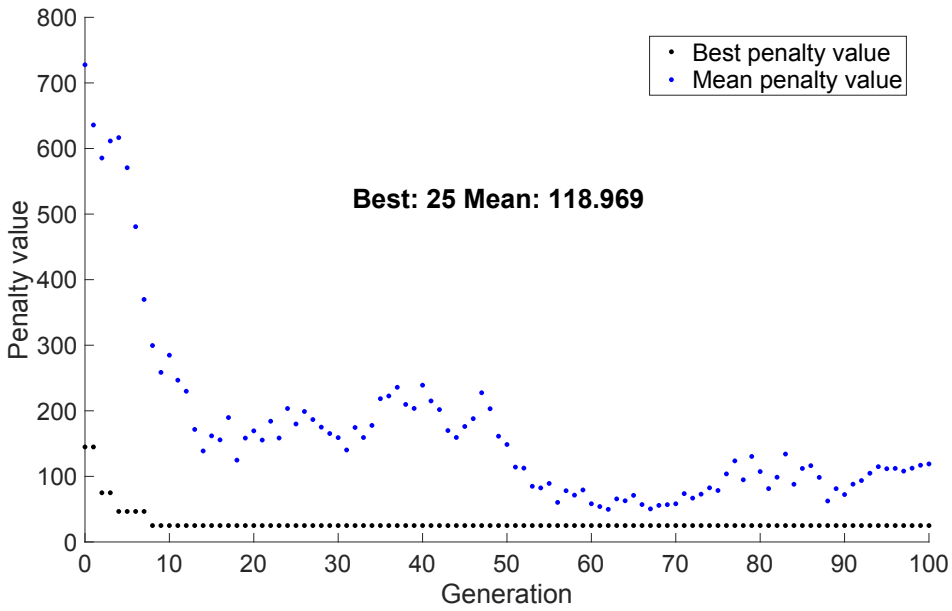


Bild: Bild 5-14 Ausgabe des Genetischen Algorithmus

Für ganzzahlige Optimierungsprobleme bildet der Algorithmus außerdem Neubildungsfunktionen (Selection Function, Creation Function, Crossover Function und Mutation Function), welche automatisiert festlegen, dass die Parameter in jedem Schritt ganzzahlig bleiben. Verändert der Nutzer die Funktionen, werden die Veränderungen ignoriert und übersprungen, weshalb sie in dem formalen Modell unverändert bleiben [**MathWorksb**] [**Deep**].

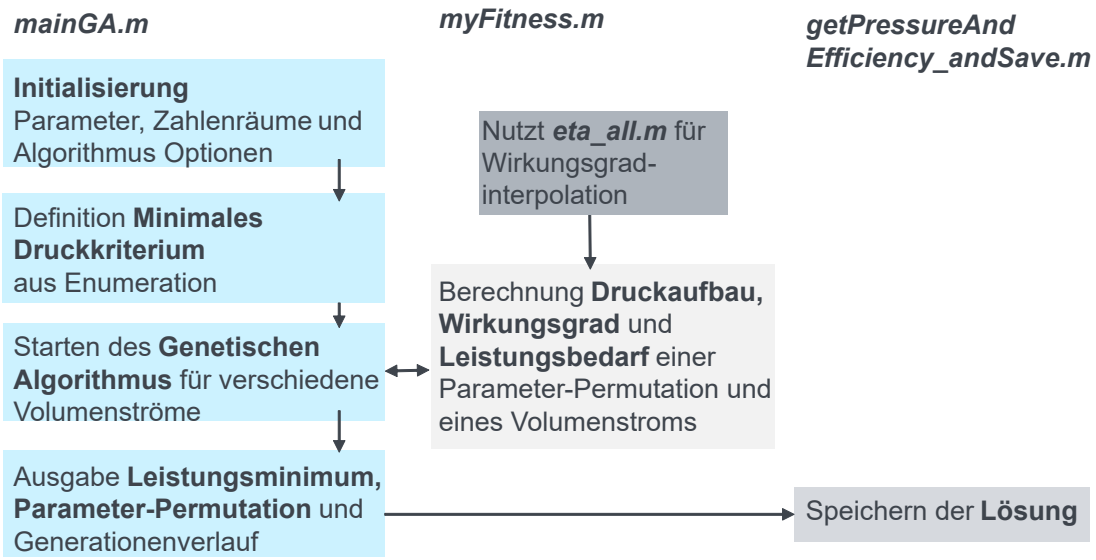


Bild: Bild 5-15 Schematische Darstellung der heuristischen Wahl der Kombinatorik

Benutzerdefinierte Zielfunktion und Abbruchkriterium: Bild Bild 5-15 stellt die Zusammenhänge des formalen Modells der heuristischen Wahl der Kombinatorik dar. Die Zielfunktion *myFitness.m* wird durch den Nutzer definiert. In Rahmen der vorliegenden Arbeit wird mit der Zielfunktion für eine Parameter-Permutation der aktuellen Population der Druckaufbau nach der analytischen Beschreibung (Gleichung 4-1) berechnet. Anschließend wird der gesamte Druckaufbau aus der Summe der einzelnen - ein bis drei - Ventilatoren berechnet. Es werden außerdem der Leistungsbedarf nach Gleichung 4-3 sowie die Wirkungsgrade der ein bis drei kombinierten Ventilatoren berechnet. Genau wie für die Enumeration und Faktorisierung wird dazu die 2D-Interpolation (Kapitel 4.1.2) verwendet.

Mit dieser Optimierungsmethode wird bisher noch keine Betrachtung der einzelnen Stränge der Referenzluftverteilung vorgenommen, sondern nur der einzelnen Ventilatoren. So wird im formalen Modell des GA nicht wie in der Enumeration mit der Strang-Matrix gearbeitet (Laufindizes i und j), sondern nur mit dem Laufindex j zur Nummerierung der Ventilatoren. Grund dafür ist, dass so die Arbeitsweise des genetischen Algorithmus vorerst verständlicher nachzuvollziehen ist. Mit dem aktuellen Wissenstand kann das formale Modell des GA jedoch nach dem Referenz-Modell der Enumeration um die strangweise Betrachtung und die damit einhergehende Prüfung der strangweisen Druckgleichung 3-6 erweitert werden.

Die strangweise Prüfung der Druckgleichung der Enumeration (siehe Bild Bild 5-3) liefert in den bisher untersuchten Luftverteilungen ersatzweise den minimalen Druckaufbau, welcher zur Einhaltung der Druckgleichung in beiden Strängen erforderlich ist. Innerhalb der Zielfunktion des GAs wird geprüft, ob dieser Druckaufbau (Druckkriterium Bild Bild 5-15) eingehalten werden kann. Nur dann stellt der zugehörige Leistungsbedarf eine Lösung dar. Dabei können ausschließlich Luftverteilungen geprüft werden, bei welchen die Positionierung der Ventilatoren identisch ist. Erfolgt diese Prüfung nicht, wird oft ein Leistungsbedarf wenig größer als $0\,/\mathrm{watt}$ gefunden, was keiner (physikalisch) gültigen Lösung für diesen Anwendungsfall entspricht.

Darüber hinaus dürfen der Zielfunktion formal nur genau die Variablen übergeben werden, die zur Bildung neuer Parameter-Permutationen verwendet werden d.h. ausschließlich die Gene. Das sind für jeden möglichen Ventilator (V_1 bis V_{numFan}) der Ventilator-Typ tp, der Indikator x und die Drehzahl n. Um deren Abtastrate auf die der Enumeration (10 %) anpassen zu können,

wird statt der Drehzahl eine der 11 Drehzahleinstellungen $numn = 0, ..., 11$ als Gen genutzt (als Variable übergeben). Die Drehzahl wird im Anschluss in Abhängigkeit ihrer Einstellung berechnet. So wird innerhalb der Zielfunktion in die beschriebenen Gleichungen 4-1 bis 4-3, welche zur Berechnung des Leistungsbedarfs verwendet werden, statt n, $n(numn)$ eingesetzt. Werden weitere Variablen an die Zielfunktion übergeben, kann der GA nicht ausgeführt werden. Die Variablen der Referenzluftverteilung (Anzahl der Ventilatoren, Skalierfaktoren, Drehzahl, Strangabschnittsvolumenstrom und ähnliche Variablen) müssen daher in Form von globalen Variablen definiert werden.

Der GA stoppt, sobald eines der folgenden Standardabbruchkriterien eintritt:

- Es werden mehr Generationen als 100 * (Anzahl der Parameter) gebildet
- Der minimale Zielfunktionswert ändert sich im Durchschnitt innerhalb von 50 Generationen weniger als die gesetzte Toleranz von 10^{-6}

Das formale Modell gibt im Anschluss den kleinsten gefundenen Leistungsbedarf, welcher dem globalen aber auch dem lokalen Minimum entsprechen kann, und zugehörige Parameter-Permutation aus. So gibt die heuristische Optimierungsmethode, wie bereits zu Anfang des Kapitels 5.3 erwähnt, keine Garantie für die Güte der Lösung. Es besteht die Möglichkeit, nach jedem Start der Simulation ein anderes Ergebnis zu erhalten. An einem frei gewählten Zahlenbeispiel in Tabelle Tabelle 5-1 soll die Bedeutung des heuristischen Verhaltens dieser Methode verdeutlicht werden.

Tabelle Tabelle 5-1 Güte der heuristischen Lösung

Beschreibung	*Leistungsbedarf bei Parameter-Permutation [n_1 n_2 n_3, tp_1 tp_2 tp_3, x_1 x_2 x_3]*
Mögliches Ergebnis der Heuristik bei einer 1. Simulation	
Frei gewähltes Zahlenbeispiel	$P = 1000$ W bei [60 50 80, 3 2 4, 1 1 1]
Mögliche Ergebnisse der Heuristik bei einer 2. Simulation	
I) Gleicher minimaler Leistungsbedarf mit gleicher zugehörigen Parameter-Permutation. Lösung identisch.	$P = 1000$ W bei [60 50 80, 3 2 4, 1 1 1]
II) Gleicher minimaler Leistungsbedarf mit einer anderen zugehörigen Parameter-Permutation	$P = 1000$ W bei [30 50 90, 2 2 1, 1 1 1]
III) Anderer minimaler Leistungsbedarf mit zugehöriger Parameter-Permutation	$P = 980$ W bei [20 50 20, 4 2 1, 0 1 1]

Wie bereits in vorangehenden Kapiteln erläutert, gibt es oft mehrere Parameter-Permutationen, die zum gleichen Leistungsbedarf führen. Weiterhin ist zu berücksichtigen, dass die Anfangspopulation des GA zufällig gewählt wird und die Mechanismen zur Bildung neuer Populationen darauf aufbauen. So findet der Algorithmus das Ergebnis II der 2. Simulation in Tabelle Tabelle 5-1, weil beispielsweise die Parameter-Permutation der Lösung der 1. Simulation (anders als bei Ergebnis I) in keiner der Populationen der 2. Simulation auftritt. Gleichzeitig ist die

Parameter-Permutation, die in II gefunden wird, jedoch die mit dem niedrigsten Leistungsbedarf in den Populationen der 2. Simulation. Wobei der Leistungsbedarf zufällig in beiden Simulationen gleich ist. In Ergebnis III wird ersichtlich, dass sowohl das Ergebnis der 1. Simulation als auch die Ergebnisse I und II der 2. Simulation nicht dem globalen Minimum entsprechen. Es kann allerdings auch keine Aussage dazu getroffen werden, ob das Ergebnis III dem globalen Minimum entspricht.

Daraus kann geschlossen werden, dass sowohl die Lösung, als auch die Populationen dieser Optimierungsmethode für das gegebene (gleichbleibende) Problem variieren können. Um die Wahrscheinlichkeit zu erhöhen, das globale Minimum zu finden, wird die Anzahl der Generationen innerhalb welcher die Toleranz eingehalten werden muss, von 50 auf 100 erhöht.

5.3.3 Grenzen der heuristischen Wahl

Die untersuchte Heuristik kann Luftverteilungen mit:

- einer Kombination aus maximal ein bis drei Ventilatoren,
- vier verschiedenen zur Auswahl stehenden Ventilator-Typen entsprechend der Beschreibung in Kapitel 4.1.1,
- Drehzahleinstellungen mit einer maximalen Abtastrate von 10 %

abbilden.

Es ist zu beachten, dass mit der bestehenden - verständlich nachzuvollziehenden - Heuristik nicht jeder Strang der Luftverteilung einzeln betrachtet werden kann. Nur ein gesamter Luftvolumenstrom kann vorgegeben und zur Berechnung verwendet werden. Aus diesem Grund können hiermit noch keine hybriden oder dezentralen Luftverteilungen abgebildet werden. Weiterhin kann deshalb keine Prüfung der Druckgleichung im Strang erfolgen. Eine serielle Verschaltung der ein bis drei Ventilatoren ist jedoch möglich und die Komponenten Kanalstück und Klappe werden durch einen definierten maximalen Druckabfall im Netz berücksichtigt. Begrenzt ist die Simulation mit der bestehenden Heuristik also auf zentrale Luftverteilungen, deren maximaler Druckabfall bekannt ist.

Dabei wird jeder Ventilator bisher mit drei Parametern beschrieben. Insgesamt wird hier also ein Optimierungsproblem mit maximal neun Parametern gelöst. Der verwendete GA kann aber auch Optimierungsprobleme mit einer weitaus größeren Anzahl an Parametern lösen. Eine genaue Anzahl kann nicht genannt werden. Diese ist abhängig von den Gleichungs- und Ungleichungs-Nebenbedingungen, wie komplex es ist, diese einzuhalten und davon, ob die Parameter ganzzahlig bleiben oder beispielsweise für die Drehzahleinstellung auf eine kontinuierliche Betrachtung über gegangen wird.

Im Gegensatz zu Enumeration und Faktorisierung werden mit der Heuristik und der angegebenen Luftverteilung die Grenze des physikalischen Arbeitsspeichers nicht erreicht. Überdies werden hier nicht die Grenzen des GA dargestellt, vielmehr wird beschrieben, was mit dem bestehenden formalen Modell der Heuristik simuliert wird.

6 Evaluierung und Nutzen der Optimierungsmethoden

In diesem Kapitel werden die Optimierungsmethoden hinsichtlich ihrer Eignung zur energetischen Optimierung von Luftverteilungen evaluiert.

Dazu wird zunächst anhand eines Auslegungsszenarios mit Hilfe der Enumeration gezeigt, dass Luftverteilungen mit einem großen Entscheidungsspielraum eine höhere Energieeinsparung erzielen, als solche mit einem kleineren, wodurch die Relevanz der Optimierungsmethoden im Allgemeinen verdeutlicht wird.

Anschließend wird gezeigt, wie gut die Simulationsergebnisse der Methoden Faktorisierung und Heuristik mit denen der Enumeration übereinstimmen, was zeigt in welchem Maß die beiden Methoden für die Lösung des Optimierungsproblems geeignet sind. Ferner folgt eine Bewertung aller drei Methoden bezüglich ihrer Laufzeit, ihrer Güte und ihrer Erweiterbarkeit. Die zugrunde liegenden Definitionen werden beschrieben.

6.1 Nutzen der Optimierungsmethoden

Folgendes Auslegungsszenario wird betrachtet: Eine Lüftungsanlage wird für ein Bürogebäude mit zwei Zonen ausgelegt, wobei die Luftkonditionierung vernachlässigt wird. Für Luftverteilungen dieser geringen Netzgeometrie wird üblicherweise ein zentraler Ventilator vorgesehen, dessen Betrieb auf einen definierten Punkt (*maximaler Druckabfall im System* | *maximaler Luftvolumenstrom*) ausgelegt wird. In einer Standardplanung wählt ein Planer wenige Ventilator-Typen aus, bestimmt den Betriebspunkt in deren Kennfeld und entscheidet sich im Anschluss für den Ventilator-Typ, bei welchem der höchste Wirkungsgrad erzielt wird.

Wie bereits in Kapitel 2.2 dargelegt, ist der zentrale Ventilator nicht zwangsläufig die Variante mit dem geringsten Energiebedarf. Wird eingeräumt, dass eine variable Anzahl und Position der Ventilatoren möglich ist, sind gleichzeitig andere Betriebspunkte der einzelnen Ventilatoren möglich. Die Auswahl der Ventilator-Typen bleibt außerdem weiterhin bestehen. Der Entscheidungsspielraum des Planers erweitert sich damit enorm. Wird nicht jede der entstehenden Möglichkeiten berechnet, fehlt allerdings die Entscheidungsgrundlage. Auf der anderen Seite müsste für die Berechnung unverhältnismäßig viel Zeit aufgewendet werden.

Genau für solche Fälle dienen die Optimierungsmethoden dazu, mit überschaubarem zeitlichen Aufwand, die besten Entscheidungen innerhalb eines Entscheidungsspielraums zu treffen, sodass der minimale Energiebedarf einer Luftverteilung erreicht wird. Wie groß die potentielle Energieeinsparung im Auslegungsszenario ist, wird nachfolgend im Detail bestimmt.

Bei der deterministischen Optimierungsmethode Enumeration wird jede durch das Optimierungsproblem beschreibbare Parameter-Permutation einer Luftverteilung genau einmal verwendet, um jeden möglichen Leistungsbedarf zu berechnen (siehe Kapitel 5.1.2). Die Berechnung beruht auf analytischen Gleichungen, welche in Abhängigkeit der Parameter-Permutation deren Ausgabewert ändern, ansonsten jedoch für jede Leistungsbestimmung unverändert bleiben. Aus allen Leistungsbedarfen wird erst im Anschluss der minimale bestimmt. Für eine Validierung dieser Methode reicht es aufgrund des gleichbleibenden Lösungsverfahrens aus, wenige Leistungsbedarfe stichprobenartig nachzuvollziehen und analytisch zu prüfen.

Die Leistungsbedarfe dieser Überprüfung stimmen dabei zu 100 % mit den Leistungsbedarfen

der Simulation (Simulationsergebnisse) überein. So kann davon ausgegangen werden, dass kein Fehler in dem formalen Modell der Enumeration vorliegt. Sie wird hier verwendet, um den Entscheidungsspielraum im Auslegungsszenario abzubilden und das daraus entstehende Optimierungsproblem zu lösen.

6.1.1 Auslegungsbeispiel Luftverteilung

Der maximal erforderliche Außenluftvolumenstrom Q eines Bürogebäudes mit zwei Räumen - einem Großraumbüro (Maximalbesetzung 50 Personen, Q_I) und einem Konferenzraum (Maximalbesetzung 30 Personen, Q_{II}) - wird nach DIN EN 15251 Tabelle B.2 berechnet. Es wird davon ausgegangen, dass der gesamte Luftaustausch in den Räumen durch die Lüftungsanlage erfolgt. Der Luftwechsel durch Infiltration wird vernachlässigt. Außerdem wird angenommen, dass der maximal erforderliche Außenluftvolumenstrom über den ganzen Tag konstant gehalten wird (KVS). Gesucht wird eine Luftverteilung mit zwei Strängen (siehe Bild Bild 6-1).

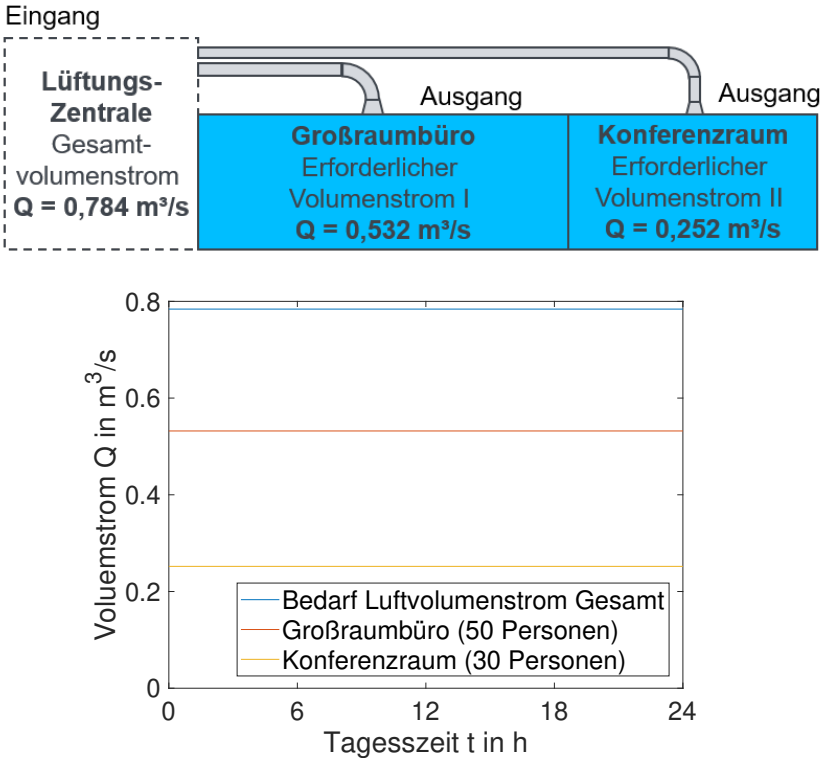


Bild: Bild 6-1 Auslegung einer Lüftungsanlage in einem Bürogebäude mit zwei Lüftungskanälen (Strängen). Schematische Darstellung der Aufgabenstellung des Auslegungsbeispiels.

Die Auslegung der Lüftungskomponenten erfolgt für drei Varianten. In der ersten Variante wird eine Luftverteilung mit genau einem Ventilator geplant. Die Kenndaten des Ventilators werden dabei Herstellerunterlagen entnommen, welche in die Modellierung der Enumeration implementiert werden. Durch die Limitierung auf maximal einen Ventilator ist dessen Position festgelegt (zentrale Lüftungsanlage). Es verbleiben zwei Freiheitsgrade, die Drehzahleinstellung und der Ventilator-Typ. Drei verschiedene Ventilator-Typen (Gebhardt Ventilatoren VZR200, VZR225, VZR250 [**GebhardtVentilatoren**]) stehen zur Auswahl. Sowohl der Laufraddurchmesser als auch der maximal förderbare Volumenstrom steigt von Typ VZR200 zu Typ250.

Die Kennfelder der Typen werden mit Regressions- und Interpolationsfunktionen der MATLAB *Curve Fitting Toolbox* beschrieben. So wird durch das Auslesen von Datenpunkten (Druckauf-

bau|Volumenstrom) bei Volllast ein Polynom zweiten Grades erstellt, mit welchem für einen gegebenen Volumenstrom der Druckaufbau bestimmt werden kann. Um die Vorgehensweise des formalen Modells der Enumeration beibehalten zu können, werden die Teillasten abgebildet, in dem das Polynom in Abhängigkeit der Drehzahleinstellung (von 40 bis 100 %) verändert wird. Die in Kapitel 4.1.2 beschriebenen analytischen Gleichung werden durch das Polynom ersetzt. Außerdem wird die 2D-Interpolation des Wirkungsgrades durch eine MATLAB Interpolationsfunktion (thin plate interp) der genannten Toolbox ersetzt, um Unregelmäßigkeiten des realen Kennfeldes genauer abbilden zu können. Bild Bild 6-2 stellt eine lineare Interpolation und eine *Thin Plate Interpolation* im Vergleich dar. Die entsprechende Anpassung des formalen Modells ist im Anhang zu finden.

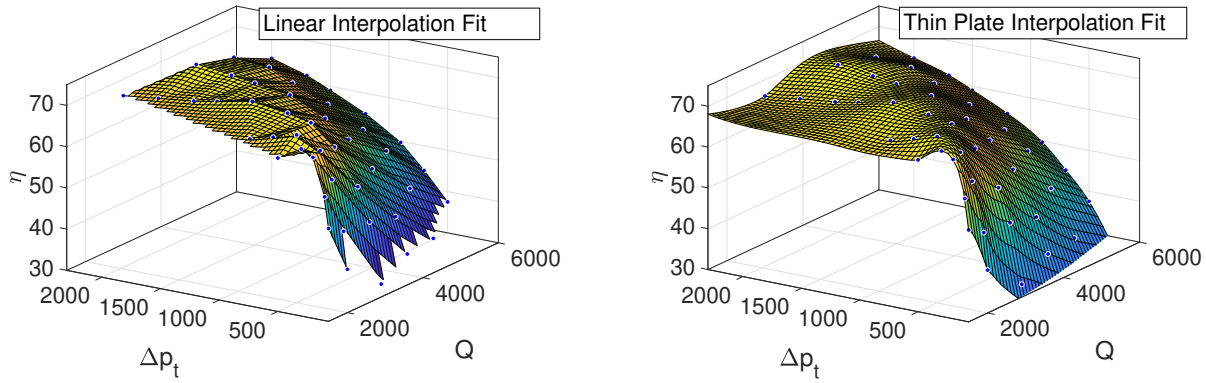


Bild: Bild 6-2 Abbildung eines realen Ventilator-Kennfeldes links mit einer linearen Interpolationsfunktion, rechts mit der Thin Plate Interpolation

Die Abbildung der Ventilator-Kennfelder wird genauso in der zweiten und dritten Variante des Auslegungsszenarios verwendet. Der Unterschied besteht darin, dass die maximale Anzahl der Ventilatoren auf zwei (Variante II) oder auf drei (Variante III) erhöht wird. Damit steigt die Anzahl der Parameter-Permutationen und zugehörigen Leistungsbedarfe. Im Anschluss werden die drei Fälle miteinander verglichen. Das Energieeinsparpotential, welches mit der erarbeiteten Optimierungsmethode Enumeration erzielt werden kann, wird in den folgenden Ergebnissen zum Ausdruck gebracht:

Variante I

Ausgangsdaten:

Volumenstrom Q_I = 0,532 m^3 s^{-1}, Q_{II} = 0,252 m^3 s^{-1}

Parameter-Permutation und Lösung:

Strang-Matrix = $\begin{bmatrix} 1 & | & 1 & 1 & 0 & | & 1 & 0 \\ 1 & | & 1 & 0 & 1 & | & 0 & 1 \end{bmatrix}$

Drehzahl-Variation [n_1 = 80%, n_2 = -, n_3 = -]

Typ-Variation [tp_1 = 1, tp_2 = -, tp_3 = -]

Zielfunktionswert der Lösung P = 1180,30 W

Wirkungsgrade [η_1 = 66,9%, η_2 = -, η_3 = -]

Gleichungssystem:

Strang I: $\Delta p_{t,I}(n, tp, Q)_1 + \sum_{m=1}^{numDuct} \Delta p_{v,R,I}(Q)_m + \Delta p_{v,K,I}(\alpha, Q)$ = 0

$1007 - 898 - 7642/\alpha_I^2$ = 0

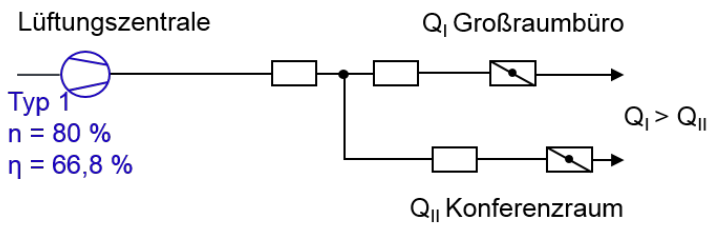
Strang II: $\Delta p_{t,II}(n, tp, Q)_1 + \sum_{m=1}^{numDuct} \Delta p_{v,R,II}(Q)_m + \Delta p_{v,K,II}(\alpha, Q)$ = 0

$1007 - 678 - 1715/\alpha_{II}^2$ = 0

Randbedingungen: $\alpha_I, \alpha_{II} \geq 0°$,$\alpha_I, \alpha_{II} \leq 90°$

Klappen-Öffnungswinkel: α_I = 8°, α_{II} = 2°, wobei 0° einer geschlossenen Klappe entspricht

Graphische Darstellung:



Steht nur ein Ventilator zur Verfügung, wird der Ventilator in Strangabschnitt I positioniert, um die Bedingung von mindestens einem Ventilator pro Strang zu erfüllen. So wird sicher gestellt, dass in jedem Raum der erforderliche Luftvolumenstrom vorliegt. Der Volumenstrom und der quadratisch davon abhängige Druckabfall in den Kanalstücken des Strangs Großraumbüro sind größer als der des Strangs Konferenzraum. Diese und nachfolgende Betrachtung erfolgt ohne die Berücksichtigung der Druckabfälle durch die Klappen. Die Auslegung des zentralen Ventilators erfolgt für den Strang mit dem größten Druckabfall, demnach für Strang I. Nachdem der Leistungsbedarf für alle Parameter-Permutationen dieser Konfiguration ermittelt ist, wird das Leistungsminimum bei 1180,30 W festgestellt, um gerade den Druckabfall von 898 Pa zu kompensieren. Diese Luftverteilung hat nach Gleichung 3-14 somit einen Energiebedarf W_I von 28,3 kW h d^{-1}.

Aufgrund der Abtastrate der Drehzahleinstellung kann der Druckaufbau des gewählten Ventilators in einem Schritt einen Wert von 885 Pa annehmen, im nächsten Schritt einen Wert von 1007 Pa. Der Druckabfall der Kanalstücke in Strang I kann jedoch ausschließlich mit dem Druckaufbau des zweiten Schritts kompensiert werden, da der des ersten nicht ausreicht. Es wird deutlich, dass mit einer feineren Abtastrate möglicherweise eine energieeffizientere Lösung gefunden werden kann. Mit dieser Abtastrate werden, trotz einer Anpassung der Druckabfälle der Klappen an die Realität, fast komplett geschlossene Klappen notwendig, um die Druckgleichungen des Gleichungssystems einhalten zu können.

In Bezug auf die Klappen-Öffnungswinkel ist weiterhin zu berücksichtigen, dass diese erst bestimmt werden, nachdem geprüft wird, ob der Druckabfall durch die Kanalstücke im Strang mit Hilfe des Druckaufbaus der Ventilatoren im Strang überwunden werden kann. Der Öffnungswinkel wird also an chronologisch letzter Stelle und ausschließlich durch die noch fehlende Druckdifferenz von Druckaufbau durch die Ventilatoren im Strang (im zu prüfenden Leistungsminimum) zu Druckabfall durch die Kanalstücke im Strang bestimmt. Der Faktor vor dem Öffnungswinkel (hier beispielsweise 7.642) ergibt sich ausschließlich durch den Volumenstrom im Strangabschnitt der Klappe. Gleiches gilt für nachfolgende Betrachtungen.

Variante II

Ausgangsdaten:

Volumenstrom Q_I = 0,532 m^3 s^{-1}, Q_{II} = 0,252 m^3 s^{-1}

Parameter-Permutation und Lösung:

Strang-Matrix = $\begin{bmatrix} 1\,0 \mid 1\,1\,0 \mid 1\,0 \\ 0\,1 \mid 1\,0\,1 \mid 0\,1 \end{bmatrix}$

Drehzahl-Variation [n_1 = 60%, n_2 = 50%, n_3 = -]

Typ-Variation [tp_1 = 3, tp_2 = 3, tp_3 = -]

Zielfunktionswert der Lösung P = 945,66 W

Wirkungsgrade [η_1 = 70,8%, η_2 = 70,7%, η_3 = -]

Gleichungssystem:

Strang I: $\sum_{m=1}^{2} \Delta p_{t,I}(n, tp, Q)_m + \sum_{m=1}^{numDuct} \Delta p_{v,R,I}(Q)_m + \Delta p_{v,K,I}(\alpha, Q)$ = 0

$930 - 898 - 7642/\alpha_I^2$ = 0

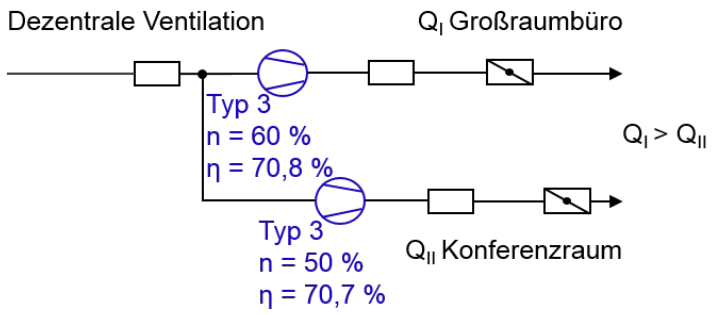
Strang II: $\sum_{m=1}^{2} \Delta p_{t,II}(n, tp, Q)_m + \sum_{m=1}^{numDuct} \Delta p_{v,R,II}(Q)_m + \Delta p_{v,K,II}(\alpha, Q)$ = 0

$692 - 678 - 1715/\alpha_{II}^2$ = 0

Randbedingungen: $\alpha_I, \alpha_{II} \geq 0°$,$\alpha_I, \alpha_{II} \leq 90°$

Klappen-Öffnungswinkel: α_I = 15°, α_{II} = 11°

Graphische Darstellung:



Stehen zwei Ventilatoren für diese zwei Stränge mit unterschiedlichem Volumenstrom zur Verfügung, liegt das Konzept der dargestellten zentralen Lüftung mit dezentralen Ventilatoren - hier mit *dezentraler Ventilation* bezeichnet - nahe. So kann jeder Ventilator für den im jeweiligen Strang erforderlichen Volumenstrom optimal ausgelegt und betrieben werden. Obwohl der Druckabfall durch die Kanalstücke in beiden Strängen keine Veränderung zu Variante I aufweist (Druckabfall durch Klappen unberücksichtigt), muss der Ventilator in Strang II einen wesentlich geringeren Druckabfall überwinden. Er kann für diesen optimal betrieben werden. Da der obere Ventilator in Variante II außerdem von einem geringeren Volumenstrom durchströmt wird als der zentrale in Variante I, kann hier eine andere Drehzahleinstellung gefunden werden, sodass der

Druckaufbau auch in Strang I geringer ausfällt. Beide Ventilatoren können aufgrund der Druck- und Volumenstromveränderungen außerdem in einem besseren Wirkungsgrad betrieben werden.

Durch den Einsatz eines zweiten optimal eingesetzten Ventilators reicht somit eine Leistung von 946 W aus, um die gewünschte Luftverteilung zu realisieren. Diese hat einen Energiebedarf W_{II} von 22,7 kW h d^{-1}.

Variante III

Ausgangsdaten:

Volumenstrom Q_I = 0,532 m^3 s^{-1}, Q_{II} = 0,252 m^3 s^{-1}

Parameter-Permutation und Lösung:

Strang-Matrix = $\begin{bmatrix} 1\,1\,0 \mid 1\,1\,0 \mid 1\,0 \\ 0\,0\,1 \mid 1\,0\,1 \mid 0\,1 \end{bmatrix}$

Drehzahl-Variation [n_1 = 50%, n_2 = 50%, n_3 = 50%]

Typ-Variation [tp_1 = 2, tp_2 = 2, tp_3 = 3]

Zielfunktionswert der Lösung P = 926,44 W

Wirkungsgrade [η_1 = 70,6%, η_2 = 70,6%, η_3 = 70,7%]

Gleichungssystem:

Strang I: $\sum_{m=1}^{3} \Delta p_{t,I}(n, tp, Q)_m + \sum_{m=1}^{numDuct} \Delta p_{v,R,I}(Q)_m + \Delta p_{v,K,I}(\alpha, Q) = 0$

$902 - 898 - 7642/\alpha_I^2 = 0$

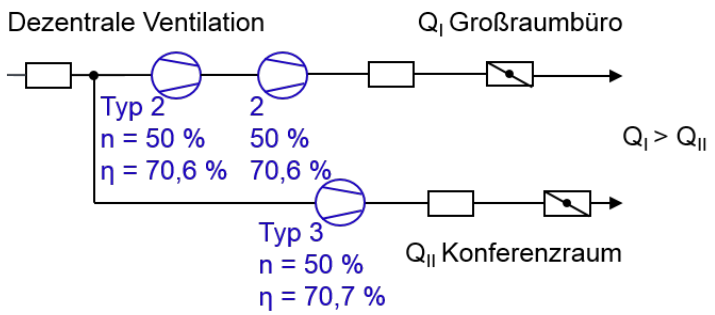
Strang II: $\sum_{m=1}^{3} \Delta p_{t,II}(n, tp, Q)_m + \sum_{m=1}^{numDuct} \Delta p_{v,R,II}(Q)_m + \Delta p_{v,K,II}(\alpha, Q) = 0$

$692 - 678 - 1715/\alpha_{II}^2 = 0$

Randbedingungen: $\alpha_I, \alpha_{II} \geq 0°$,$\alpha_I, \alpha_{II} \leq 90°$

Klappen-Öffnungswinkel: α_I = 40°, α_{II} = 11°

Graphische Darstellung:



Stehen drei Ventilatoren für zwei Stränge mit unterschiedlichem Volumenstrom zur Verfügung, wird die *dezentrale Ventilation* mit einer seriellen Verschaltung zweier Ventilatoren gekoppelt. Die Vorteile der Variante II werden beibehalten. Die serielle Verschaltung zweier Ventilatoren verursacht nach Kapitel 3.2, eine Aufteilung des erforderlichen Druckaufbaus auf beide Ventilatoren. Solch ein Ansatz ist sinnvoll für Strang I mit dem größeren Volumenstrom. Ein geringerer Druckaufbau erfordert eine geringere Drehzahl und verschiebt den Betriebspunkt des Ventilators möglicherweise hin zu einem höheren Wirkungsgrad.

In Variante III trifft nur Ersteres zu, die Drehzahl beider Ventilatoren ist geringer. In Summe kann damit ein Druckaufbau (902 Pa) berechnet werden, welcher dem maximalen Druckabfall des Stranges II (898 Pa) näher kommt als der in Variante II (930 Pa), sodass eine Energieeinsparung möglich ist. Der Vorteil der seriellen Verschaltung ist in dieser Betrachtung daher der Abtastrate

der Drehzahl geschuldet.

Durch den Einsatz dreier optimal eingesetzter Ventilatoren reicht somit das Leistungsminimum von 926 W aus, um die gewünschte Luftverteilung zu realisieren. Diese hat einen Energiebedarf W_{III} von 22,2 kW h d^{-1}.

6.1.2 Energieeinsparpotential

Das Energieeinsparpotential WP_s beschreibt die prozentuale Energieeinsparung, die nach Gleichung 6-1 berechnet wird. 100% entsprechen dem Energiebedarf von Variante I als Referenz. Sie ist dabei bereits für eine Auswahl von drei Ventilator-Typen und alle Drehzahleinstellungen optimiert. Variante I wird als Referenzszenario genutzt, da sie einer typischen Auslegung einer Lüftungsanlage diesen Maßstabs entspricht. So wird in der Praxis oft ein zentrales Lüftungsgerät nach dem maximalen Druckabfall der gesamten Luftverteilung ("worst case Szenario) ausgelegt.

$$WP_s = 100 - \frac{100}{W_I} * W(numFan, pos_{1,\ldots,numFan}, tp_{1,\ldots,numFan}, n_{1,\ldots,numFan}) \tag{6-1}$$

WP_s	Energieeinsparpotential der Luftverteilung in %
W_I	Energiebedarf der Variante I in J
W	Energiebedarf einer anderen Variante mit einem größeren Entscheidungsspielraum in J
$numFan$	Maximale Anzahl der Ventilatoren im System
pos	Indikator zur Position des Ventilators (Vorhandensein (1) oder nicht Vorhandensein (0)); Angabe als Positions-Tupel
tp	Einheitsloser Ventilator-Typ
n	Ventilator-Drehzahleinstellung in %

Daraus ergibt sich für Variante II ein Energieeinsparpotential von 19,9%. Für Variante III kann eine Energieeinsparung von 21,5% erzielt werden. Vor der Ausführung der Planung kann sie einer wirtschaftlichen Betrachtung sowie einer Lebenszyklusanalyse unterzogen werden. Hierbei sind den potentiellen 20% Energieeinsparung im Betrieb die Mehrkosten durch erhöhte Material- und Wartungskosten sowie der gesteigerte Primärenergieeinsatz des zusätzlichen Materials, gegenüberzustellen. Diese Gegenüberstellung ist jedoch nicht Teil der vorliegenden Arbeit.

Anhand des Auslegungsbeispiels kann nachgewiesen werden, dass für eine energetische Optimierung eine große Anzahl an zur Verfügung stehenden Freiheitsgraden genutzt wird, um nicht nur dezentrale Ventilatoren in zentralen Luftverteilungen, sondern gleichzeitig die serielle Verschaltung zweier Ventilatoren zu verwenden. Die Relevanz der Optimierung durch einen großen Entscheidungsspielraum kann aufgezeigt werden. Ferner kann nachgewiesen werden, dass die Enumeration sich eignet, um die energetische Optimierung der diskret-kombinatorisch abgebildeten Luftverteilung durchzuführen.

Aufgrund der deterministischen Arbeitsweise der Enumeration kann davon ausgegangen werden, dass Luftverteilungen größerer Netzgeometrie und größeren Entscheidungsspielraums auf gleiche Weise energetisch optimiert werden können. Ihr Energieeinsparpotential ist als wesentlich größer einzuschätzen. Limitierende Faktoren der aktuellen Modellierung sind - wie in Kapitel 5.1.2 erläutert - die Laufzeit und die Matrizen-Größe.

6.2 Evaluierung durch Vergleich

In vorangehendem Kapitel findet der Nachweis statt, dass die Optimierungsmethode Enumeration zur Lösung des beschriebenen Optimierungsproblems geeignet ist. Im Weiteren dient sie daher als Referenz für eine Evaluierung der beiden anderen Optimierungsmethoden. Stimmen die Simulationsergebnisse der Faktorisierung und der Heuristik mit denen der Enumeration überein, sind deren Modellierungen in gleichem Maße zur Lösung geeignet.

Ein Vergleich der Simulationsergebnisse kann nur erfolgen, wenn gleiche Ausgangsbedingungen bestehen. Um diese zu schaffen, werden zuerst die unterschiedlichen Ausgangsbedingungen der drei Methoden aufgefasst, anschließend werden diese aneinander angepasst.

6.2.1 Unterschiede der Optimierungsmethoden

Sowohl in Kapitel 5.2.3 der Faktorisierung, als auch in Kapitel 5.3.3 der Heuristik wird dargelegt, dass mit den beschriebenen Modellierungen beider Optimierungsmethoden nicht die gleichen Freiheitsgrade abgebildet werden können wie mit der Enumeration.

An dem konkreten Beispiel der Referenzluftverteilung, werden die Unterschiede der Methoden erläutert: Für maximal drei Ventilatoren und sechs verschiedene konstante Gesamtvolumenströme - $0{,}5\,\mathrm{m^3\,h^{-1}}$ bis $3\,\mathrm{m^3\,s^{-1}}$, in $0{,}5\,\mathrm{m^3\,s^{-1}}$-Schritten (Volumenströme in beiden Strängen gleich hoch) -

- wird mit der deterministischen **Enumeration**, für eine Abtastrate der Drehzahleinstellung von 10 % aus 1.107.392 - $num\pi_D$ x $num\pi_T$ x $num\pi_P$ - Parameter-Permutationen, die gleiche Anzahl an Leistungsbedarfen und eine weitaus höhere Anzahl an Zwischenergebnissen berechnet und gespeichert. Dabei wird eine Parameter-Permutation durch die Anordnung der Auswahlen - [Auswahl π_P, Auswahl π_T, Auswahl π_D] - beschrieben. Dieser Vorgang nimmt 91,4 % der Programm-Laufzeit ein.

  Danach findet die Prüfung des Klappen-Öffnungswinkels 445 mal statt, bis die für dieses Anwendungsbeispiel gültige Lösung gefunden wird. Die Prüfung nimmt 5 % der Programm-Laufzeit ein.

  Der komplette Vorgang wird sechs Mal (für jeden Volumenstrom) wiederholt. Insgesamt werden hier 6,64 Millionen Leistungsbedarfe in einer Programm-Laufzeit von 429 Minuten berechnet.

- wird mit der deterministischen **Faktorisierung** für eine Diskretisierung von (Δp_d = 10 Pa, Δq = $0{,}5\,\mathrm{m^3\,s^{-2}}$) aus 65,93 Millionen - $num\pi_{Druck}$ x $num\pi_T$ - Parameter-Permutationen, die gleiche Anzahl an Leistungsbedarfen und zugehörige Zwischenergebnisse berechnet und gespeichert. Hier wird eine Parameter-Permutation durch die Anordnung anderer Auswahlen - [Auswahl π_T, Auswahl π_{Druck}] - beschrieben. Statt der Drehzahl-Variation, wird die Druck-Variation berücksichtigt, wodurch nur drei der 13 Strang-Matrizen abgebildet werden. Die Abtastrate der Drehzahlen ist nicht mehr ganzzahlig, jedoch gleichzeitig

nicht kontinuierlich, da sie durch die Druck-Variation begrenzt ist. Dabei findet keine Prüfung der Klappen-Öffnungswinkel statt. Unter Berücksichtigung aller sechs Volumenströme werden 396 Millionen Leistungsbedarfe in einer Programm-Laufzeit von 58 Minuten berechnet.

- werden mit der **zufälligen Heuristik**, für eine Abtastrate der Drehzahleinstellung von 10 %, aus 681.472 - $num\pi_D$ x $num\pi_T$ x $num\pi_{Vorhanden}$ - Parameter-Permutationen, nur 9.091 Leistungsbedarfe und Zwischenergebnisse berechnet. Dies entspricht 100 Generationen von Parameter-Permutationen. Hier wird die Parameter-Permutation durch die Anordnung der Auswahlen - [Auswahl π_D, Auswahl π_T, Auswahl $\pi_{Vorhanden}$] - beschrieben und somit werden nur drei der 13 Strang-Matrizen berücksichtigt. Es findet keine Prüfung der Klappen-Öffnungswinkel statt. In jeder Generation wird außerdem nur der minimale Leistungsbedarf gespeichert, die restlichen Zwischenergebnisse werden gelöscht. Dieser Vorgang wird sechs Mal (für jeden Volumenstrom) wiederholt. Insgesamt werden hier 0,05 Millionen Leistungsbedarfe in einer Programm-Laufzeit von etwa 75 Sekunden berechnet. Im Unterschied zu den beiden anderen Methoden erhöhen sich die Programm-Laufzeit und die Anzahl berechneter Leistungsbedarfe jedoch nicht mit der Anzahl der Parameter-Permutationen, sondern ausschließlich mit der Anzahl der Volumenströme.

Weiterhin ist zu erläutern, dass mit einer feinen Diskretisierung der Faktorisierung oder einer erweiterten Heuristik (kontinuierliche Drehzahlbetrachtung) weitaus mehr Drücke als in Bild Bild 5-4 dargestellt sind, berechnet werden können. Damit wäre es möglich, eine Lösung zu finden, welche das Optimum der (begrenzten) Enumeration unterschreitet. Mit einer erweiterten Heuristik ist es theoretisch sogar möglich, den global minimalen Druckaufbau (mit zugehörigem Leistungsbedarf) zu finden, welcher ausschließlich den Druckabfall der Kanalstücke überwinden muss. Für einen Vergleich der derzeit bestehenden Methoden, wird jedoch der minimale Druckaufbau der strangweisen Prüfung aus der Enumeration herangezogen.

6.2.2 Vergleich der Lösungen

Nachfolgend wird erläutert, an welchen Stellen die Unterschiede der Methoden bestehen bleiben und an welchen sie aneinander angepasst werden: Nachdem eindeutig beschrieben wird, dass das Lösungsverfahren der Heuristik darauf beruht, nicht alle Parameter-Permutationen zu berechnen, findet an dieser Stelle keine Anpassung der Heuristik an die anderen Methoden statt, indem die Anzahl der Generationen, an die der zu berechnenden Leistungsbedarfe der anderen Methoden angepasst wird. Diese Diskrepanz der Ausgangsbedingungen bleibt bestehen. Sie wird später in der Bewertung der Methoden berücksichtigt, indem die Laufzeit auf die berechneten Leistungsbedarfe bezogen werden.

Auch für die Modellierung der Faktorisierung findet keine Anpassung statt. Die Diskretisierung der Faktorisierung wird nicht so angepasst, dass sie nur genauso viele Leistungsbedarfe wie die Enumeration berechnen muss. Sie wird so fein wie möglich gewählt, was die sehr große Anzahl an zu berechnenden Leistungsbedarfen mit sich bringt. So wird erreicht, dass der Druckaufbau des Knotenpunktes nur wenig vom minimalen Druckabfall der Enumeration abweicht. In diesem Punkt bleiben die Ausgangsbedingungen daher ebenfalls verschieden.

Die erste und einzige Anpassung der Ausgangsbedingungen findet daher für die Modellierung der Enumeration statt. Es wird die Einschränkung festgelegt, dass von 13 Strang-Matrizen nur die drei zentralen zulässig sind. Diese werden in Bild Bild 6-3 dargestellt. Freiheitsgrad 6, die

Position der Ventilatoren, wird damit eingeschränkt. Es können weder dezentrale noch hybride Luftverteilungen abgebildet werden. Damit reduziert sich hier die Anzahl an berechneten Leistungsbedarfen auf 1,53 Millionen, was eine Programm-Laufzeit von 144 Minuten in Anspruch nimmt. Für die Enumeration ist darüber hinaus zu beachten, dass die Simulation mit zwei identischen Strangvolumenströmen erfolgt, welche gleichmäßig erhöht werden und in Summe den oben genannten Gesamtvolumenströmen entsprechen. Gleiches Vorgehen ist möglich, wenn zwei voneinander abweichende Strangvolumenströme in Summe den Gesamtvolumenströmen entsprechen.

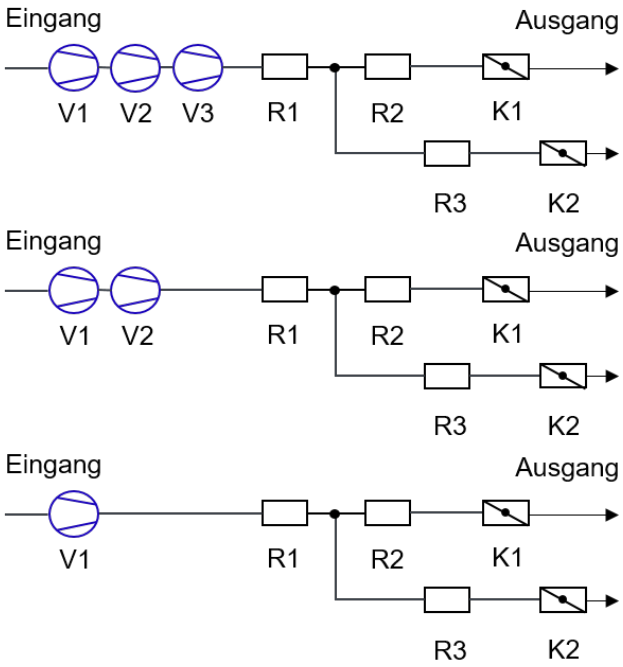


Bild: Bild 6-3 Schematische Darstellung der Evaluationsluftverteilung (Referenzluftverteilung mit eingeschränktem Freiheitsgrad 6 - Position der Ventilatoren) für einen Vergleich der Optimierungsmethoden

Bild Bild 6-4 stellt die Simulationsergebnisse der Optimierungsmethoden im Vergleich für oben beschriebene Luftverteilung und genannte Diskrepanzen der Ausgangsbedingungen dar. Auf der Y-Achse sind die Leistungsbedarfe der Enumeration als Referenz aufgetragen, auf der X-Achse die der Faktorisierung in blau und die der Heuristik (GA) in rot. Entspricht der Leistungsbedarf einer Optimierungsmethode genau dem der Enumeration, liegt er auf der diagonalen schwarzen Strichlinie. Weicht dieser um weniger als 10 % von dem der Enumeration ab, liegt der Punkt innerhalb der schwarzen Begrenzungslinien.

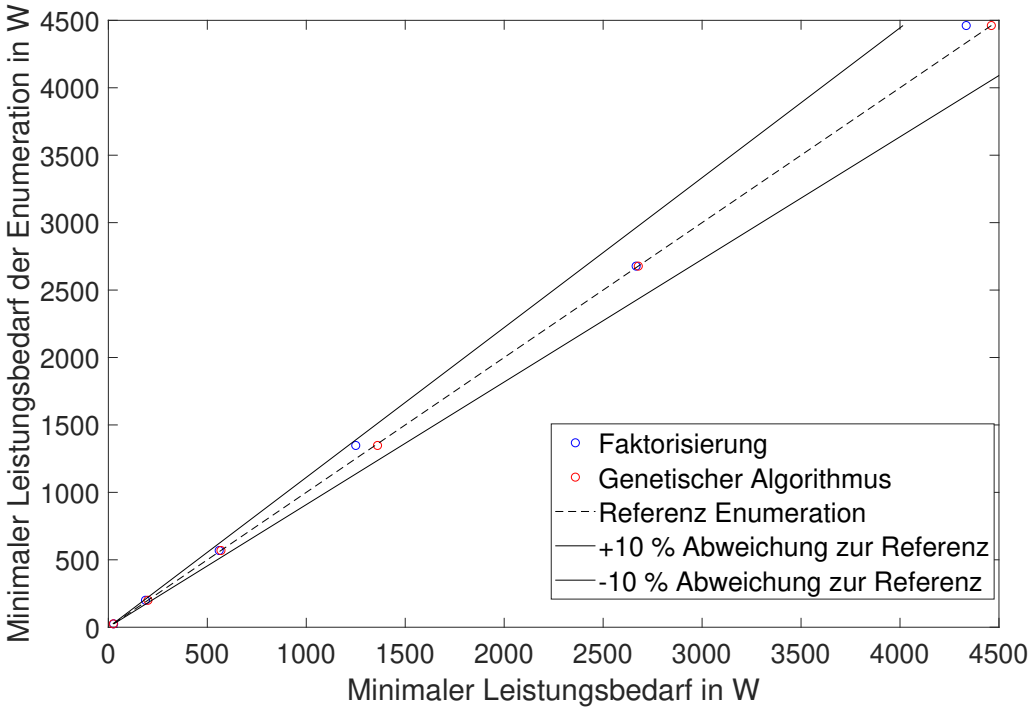


Bild: Bild 6-4 Ergebnisse der Simulation verschiedener Volumenströme mit den drei Optimierungsmethoden. Als Referenzoptimierungsmethode dient die Enumeration.

Es wird ersichtlich, dass für die Optimierungsmethoden Heuristik und Enumeration identische Leistungsbedarfe gefunden werden können. Obwohl mit der Heuristik keine Garantie bezüglich der Reduktion der Laufzeit und der Güte der Lösung gegeben werden kann, werden mit dieser Modellierung im vorliegenden Beispiel zu 100 % - in 10 von 10 Simulationen - genau die dargestellten Leistungsbedarfe gefunden. Hierbei lassen sich die minimalen Werte meist schon nach weniger als 10 Generationen (nach etwa 90 Parameter-Permutationen) finden. Die zum minimalen Leistungsbedarf zugehörige Parameter-Permutation kann dabei variieren, weil es oft mehrere Parameter-Permutationen gibt, die zum gleichen Leistungsbedarf führen. Gleiches gilt jedoch für die Enumeration; es wird mehr als eine Parameter-Permutation mit dem gleichen Leistungsbedarf gefunden. In der Modellierung der Enumeration ist allerdings formal festgelegt, dass immer die erste gefundene, die Lösung ist, welche angegeben wird. Weicht die Parameter-Permutation der Lösung der Heuristik also bei gleichem Leistungsbedarf von der der Enumeration ab, ist dies dem formalen Modell geschuldet und kein Fehler. Für zentrale Luftverteilungen kleiner Netzgeometrie, wie die der Evaluierungsluftverteilung, kann somit die Eignung dieser Methode für das Lösen des Optimierungsproblems festgestellt werden. Dies schafft die Grundlage für Methoden-Erweiterungen, auf welche in Kapitel 8 eingegangen wird.

Die Lösungen der Faktorisierung weichen in Bild Bild 6-4 leicht ab; dies ist den abweichenden Knotenpunkten geschuldet. Wie nah sie an den betrachteten Punkten ($\sum_{V1}^{V3} \Delta p|Q$) der Enumeration liegen, ist durch die Diskretisierung limitiert und für jeden Punkt verschieden. Insgesamt ist die Abweichung der Leistungsbedarfe für alle Punkte jedoch kleiner als 10 %. Für kleine Luftverteilungen kann daher auch für die Faktorisierung nachgewiesen werden, dass sie für die Lösung des Optimierungsproblems geeignet ist. Inwiefern die Methode jedoch beispielsweise für eine Betrachtung aller 13 Strang-Matrizen geeignet ist, wird in Kapitel 8 erläutert.

6.2.3 Bewertungsgrößen und Bewertung

Neben der Eignung der Methoden sollen sie anhand deren Laufzeit, Güte und Erweiterbarkeit bewertet werden. Die Begriffe werden dabei wie folgt definiert:

Laufzeit: Der Suchraum der Optimierungsmethoden ist begrenzt auf die Lösungsmenge beziehungsweise auf die Diskretisierung und die daraus zusammengesetzten Parameter-Permutationen. Wird die Anzahl der Parameter-Permutationen erhöht, wachsen der Suchraum und die korrelierenden Größen Rechenaufwand und Laufzeit (siehe Kapitel 3). Die in Kapitel 5.1.1 erwähnte rechnerspezifische Programm-Laufzeit, welche mit der MATLAB-Funktion $tic - toc$ gemessen wird, berücksichtigt dabei nicht nur das Wachstum des Suchraums, sondern den während dessen auftretenden Einfluss der Qualität der Programmierung. So wird mit dem Bewertungskriterium Laufzeit neben der Anzahl auszuführender elementarer Rechenoperatoren die Programmierung der Optimierungsmethode bewertet.

Um die drei Methoden besser miteinander vergleichen zu können, wird die Laufzeit in Tabelle Tabelle 6-1 in Zeit pro berechnetem Leistungsbedarf angegeben.

Güte: Die Güte wird durch die Zuverlässigkeit und die Konvergenzeffizienz abgebildet. Die Zuverlässigkeit sagt nach *Wünsch* [**AndreasWunsch.2016**] aus, dass eine Abweichung von der erwarteten Funktionsweise des Verfahrens als unwahrscheinlich angesehen wird und mit hoher Wahrscheinlichkeit eine Lösung erzielt wird. Im vorliegenden Fall soll mit hoher Wahrscheinlichkeit nicht nur eine beliebige Lösung erzielt werden, sondern die Lösung, welche das globale Minimum des Leistungsbedarfs beinhaltet. Zusammen mit der Konvergenzeffizienz wird bewertet, wie schnell diese Lösung mit der Optimierungsmethode erreicht wird.

Erweiterbarkeit: Anhand des Kriteriums der Erweiterbarkeit soll eine Aussage darüber gemacht werden, ob und wie gut die Freiheitsgrade der Modelle erweitert werden können.

Anknüpfend an die Definition dieser Größen, erfolgt die Bewertung: Tabelle Tabelle 6-1 stellt die Laufzeiten dar.

Tabelle Tabelle 6-1 Laufzeit der Optimierungsmethoden

Optimierungsmethode	Berechnung der Laufzeit	Laufzeit
Enumeration	1,53 Millionen/144 Minuten	10.625 Berechnungen pro Minute
Faktorisierung	395,64 Millionen/58 Minuten	6,8 Millionen Berechnungen pro Minute
Heuristik	0,05 Millionen/1,25 Minuten	40.000 Berechnungen pro Minute

Wird die Laufzeit der Enumeration mit der der Faktorisierung verglichen, kann festgestellt werden, dass obwohl die Rechenschritte der Methoden sehr ähnlich sind, die Faktorisierung wesentlich schneller ist. Ein Grund dafür liegt in der Berücksichtigung der Strangvolumenströme und in der strangweisen Prüfung der Druckgleichung. Dadurch hat sowohl die Strangnummer, die Komponentennummer und die ID einen Einfluss auf den Leistungsbedarf. Der strang- und komponentenweise Zugriff erfordert große Matrizen für die Zwischenergebnisse und eine hohe Anzahl eingebetteter Schleifen.

Insgesamt ist nicht, wie zu erwarten, die Laufzeit der Heuristik am geringsten, sondern die der Faktorisierung. Der Unterschied der beiden Methoden ist, dass die Anzahl der Berechnungen der Heuristik mit einer steigenden Modellgröße nicht zunimmt, sondern ausschließlich mit einer steigenden Anzahl an Volumenströmen (vgl. Kapitel 6.2.1). Ersteres gilt nicht für die Laufzeit der Faktorisierung.

Tabelle Tabelle 6-2 gibt einen Überblick über die Bewertung der einzelnen Kriterien, in Form eines positiven (+), eines negativen (-) und eines neutralen (o) Symbols.

Tabelle Tabelle 6-2 Bewertung der Optimierungsmethoden

Optimierungsmethode	Laufzeit	Güte	Erweiterbarkeit
Enumeration	-	+	-
Faktorisierung	+	+	o
Heuristik	+	+	+

In Bezug auf die Güte ist von den deterministischen Methoden zu erwarten, dass falls das Optimierungsproblem innerhalb der Grenzen der Methoden liegt, mit sehr hoher Wahrscheinlichkeit genau die eine Lösung erzielt wird, welche den minimalen Leistungsbedarf beinhaltet. Dieses Kriterium kann für Luftverteilungen kleiner Netzgeometrie ohne Probleme erfüllt werden. Wird jedoch die Laufzeit mitbetrachtet zeigt sich, dass dieses Ergebnis vor allem mit der Enumeration nicht schnell gefunden wird. Die Güte des heuristischen Algorithmus ist wider Erwarten hoch. Trotz zufälliger Wahl und Veränderung der Parameter-Permutationen wird in kurzer Zeit in zehn von zehn Simulationen das gleiche Ergebnis gefunden, welches dem globalen Minimum entspricht.

Ein wesentlicher Vorteil der Modellierung der Enumeration ist, dass bereits in dem jetzigen Entwicklungsstadium, im Vergleich zu dem der beiden anderen Modellierungen, sowohl die Kombination von dezentralen und zentralen Ventilatoren abgebildet werden kann, als auch die Verschaltung von seriellen Ventilatoren. So sind mit der betrachteten Evaluationsluftverteilung die Freiheitsgrade der bestehenden Enumeration nicht voll ausgeschöpft. Die Enumeration kann mindestens 10 Strang-Matrizen mehr abbilden als die beiden anderen Methoden. Auch die Anzahl der Stränge kann noch um mindestens einen erhöht werden. Für einen größeren wählbaren Zahlenraum pro Freiheitsgrad ist für die Erweiterbarkeit der Enumeration aber eindeutig die Programm-Laufzeit problematisch. Der enorme Anstieg der Laufzeit mit der Modellgröße wird in Kapitel 5.1.3 - Bild Bild 5-5 dargestellt. Das ist der Grund für die negative Bewertung der Enumeration im Punkt Erweiterbarkeit.

Die Erweiterbarkeit der beiden anderen Methoden ist eine Abschätzung auf Basis der in Kapitel 8 erläuterten Überlegungen zur Erweiterung. Mit diesen Symbolen wird vielmehr das Potential, als die bestehende Methode bewertet.

7 Zusammenfassung

In Bezug zu den Klimazielen Deutschlands steht auch in der Entwicklung von Lüftungsanlagen die Energieeffizienz zunehmend im Fokus. Derzeit verursachen die Luftkonditionierung und die Luftverteilung den Energiebedarf von Lüftungsanlagen. Verschiedene in der vorliegenden Arbeit vorgestellte Ansätze zeigen, dass sowohl die Kombination von dezentralen und zentralen Ventilatoren -*hybride Ventilatoranordnung*-, als auch die serielle, parallele oder gemischte Verschaltung von zentralen Ventilatoren zu einer Energieeinsparung in der Luftverteilung führen. Ferner führt die an den Volumenstrom- und Druckbedarf angepasste Auswahl des/der Ventilator-Typs(en) und dessen/deren Drehzahleinstellung/en zu einer Energieeinsparung in der Luftverteilung.

Die vorliegende Arbeit formuliert daraus ein kombinatorisches Optimierungsproblem, welches alle genannten Ansätze zur Energieeinsparung in Form von gültigen Lösungen zulässt. Eine Optimierung der Luftverteilung findet hierbei statt, indem Parameter zur Auslegung (Ventilator-Anzahl, -Positionierung (zentral/dezentral/hybrid, seriell/parallel/gemischt verschaltet), -Dimensionierung (Typ und Art)) sowie Parameter für den Betrieb (Arbeitsbereich (Drehzahleinstellung)) an den Bedarf der zu planenden Luftverteilung so angepasst werden, dass deren kombinatorische Zusammensetzung (Parameter-Permutation) zu einem minimalen Leistungsbedarf führt. Die Summe der gültigen Lösungen - aller Kombinationen der Ansätze zur Energieeinsparung - entspricht dabei dem Entscheidungsspielraum des Planers.

Die Anzahl der Parameter-Permutationen steigt mit dem zulässigen Zahlenraum ihrer Parameter. Der daraus entstehende Entscheidungsspielraum und die korrelierende Komplexität des Problems wachsen nach Gleichungen der Kombinatorik exponentiell. Das Ziel, sich während der Planung einer Luftverteilung, für die Parameter-Permutation zu entscheiden, welche zu dem theoretisch minimalen Leistungsbedarf führt, wird daher durch ein - auf automatischen Arbeitsanweisungen beruhendes - Lösungsverfahren erreicht. Dadurch wird der komplexe und fehleranfällige Entscheidungsprozess des Planers vereinfacht und gleichzeitig dessen zeitlicher Aufwand möglichst gering gehalten.

Um zunächst analytisch und numerisch eindeutig nachvollziehbare Lösungen des Problems zu generieren, wird der diskrete Zahlenraum der Parameter sowie die Parameteranzahl wesentlich limitiert. Am Beispiel einer Referenzluftverteilung mit kleiner Netzgeometrie (zwei Stränge, drei Strangabschnitte) kann für den Druckaufbau von ein bis drei Ventilatoren eine Auswahl aus elf ganzzahligen Drehzahleinstellungen, aus vier Typen unterschiedlichen maximalen Druckaufbaus und aus vier Positions-Tupeln getroffen werden. Ausgewählt wird dabei für jeden Ventilator je eine der genannten Optionen, allerdings ist die wiederholte Auswahl (Auswahl mit „zurücklegen") erlaubt. Für die Auswahl der Drehzahleinstellungen und die der Typen spielt die Reihenfolge eine Rolle, für die Auswahl der Positions-Tupel bleibt diese unberücksichtigt. Aus allen möglichen kombinatorischen Zusammensetzungen entstehen 1.107.392 Parameter-Permutationen.

Der Druckabfall der Referenzluftverteilung wird durch drei Kanalstücke (eines in jedem Strangabschnitt) und einer Klappe pro Strang abgebildet.

Das erste von drei Lösungsverfahren - das der Modellierung der **Enumeration** - berechnet unter Berücksichtigung eines vorgegebenen Volumenstroms pro Strang mit jeder zur Auswahl stehenden Parameter-Permutation einen analytisch beschriebenen Druckaufbau, einen interpolierten Wirkungsgrad sowie dem zugehörigen Leistungsbedarf. Durch den Vergleich aller Werte findet die Enumeration den minimalen Leistungsbedarf. Im Anschluss muss für jeden Strang der Luftverteilung die Bedingung eingehalten werden, dass der Druckaufbau gleich dem Druckabfall durch die Kanalstücke ist. Kann die Druckbilanzbedingung, unter Berücksichtigung eines zusätzlichen Druckabfalls durch eine Klappe pro Strang, für beide Stränge gleichzeitig eingehalten werden, ist die Lösung des Problems bestimmt. Das Optimierungsproblem der Referenzluftverteilung wird hiermit in etwa sieben Stunden gelöst.

Das Lösungsverfahren ist deterministisch und erfolgt daher bei jeder Simulation exakt gleich. Der Rechenaufwand wächst exponentiell mit dem Entscheidungsspielraum und der korrelierenden Modellgröße. Eine Lösung des Optimierungsproblems übersteigt mit heute verfügbaren Rechnerleistungen bereits für größere Entscheidungsspielräume als die der Referenzluftverteilung eine hinnehmbare Laufzeit.

Um Luftverteilungen größeren Entscheidungsspielraums, vor allem größerer Netzgeometrie abbilden zu können, wird im Rahmen dieser Arbeit auch die Eignung prädiktiver Verfahren, wie die des Einsatzes eines neuronalen Netzes geprüft. Verfahren dieser Art werden oft für große Datenmenge eingesetzt. Anders als bei dem hier betrachteten Optimierungsproblem haben die Eingangsdaten von neuronalen Netzen nicht zwangsläufig einen eindeutig beschreibbaren physikalischen Zusammenhang mit den Ausgangsdaten. Daher werden zwei Optimierungsmethoden betrachtet, welche zur Lösung solcher Probleme besser geeignet sind: die so genannte Faktorisierung, welche das kombinatorische Problem durch eine Aufteilung des Entscheidungsspielraums auf zwei Teilmodellierungen reduziert und eine heuristische Wahl der Kombinatorik, welche beruhend auf einem genetischen Algorithmus durch Mechanismen der natürlichen Evolutionstheorie nur die besten Lösungen „überleben"lässt.

In der hier beschriebenen Teilmodellierung der **Faktorisierung** wird dazu die Anzahl der Parameter-Permutationen für ein bis drei Ventilatoren zunächst auf die Auswahl der Drehzahleinstellung sowie die des Ventilator-Typs reduziert. Es wird statt der Auswahl der Drehzahleinstellung ein Druckaufbau für jeden Ventilator gewählt, welcher aus einer definierten Anzahl an Teildrücken gebildet wird. Darüber werden die diskreten, aber nicht mehr ganzzahligen Drehzahlen berechnet. Im Vergleich zur Enumeration erfolgt dieser Vorgang außerdem nicht nur für einen Volumenstrom, sondern für eine definierte Anzahl an Volumenströmen gleichzeitig. Dieses Optimierungsproblem wird gelöst. Die Auswahl der Positions-Tupel wird in eine im Rahmen dieser Arbeit nicht entstandene Teilmodellierung verschoben, welche in Kapitel 8 schematisch erläutert wird. Die Abbildung dezentraler und hybrider Ventiltoranordnungen sowie die Prüfung der Druckbilanz können ohne die Auswahl der Positions-Tupel nicht berücksichtigt werden.

Es wird ein Lösungs-Kennfeld bestimmt, welches für jeden Knotenpunkt (Summe des Druckaufbaus aller drei Ventilatoren|Volumenstrom) die Lösung (Parameter-Permutation, welche zum minimalen Leistungsbedarf führt) beinhaltet. Auf Basis des Kennfeldes soll in der weiteren Teilmodellierung die Lösung für einen gegebenen Volumenstrom und Druckabfall im Strangabschnitt einfach ausgelesen werden können.

Analog zu dem Lösungsverfahren der Enumeration ist das der Faktorisierung deterministisch und der Rechenaufwand wächst mit dem - hier anders definierten - Entscheidungsspielraum

und der korrelierenden Modellgröße. Der Vorteil dieser Methode ist, dass das Lösungs-Kennfeld nicht für einzelne Luftverteilungen berechnet werden muss. Ist das Kennfeld einmalig erstellt, sind die Lösungen für viele Luftverteilungen gleichen Entscheidungsspielraums bekannt und können in der weiteren Teilmodellierung abgerufen werden.

Mit dem dritten betrachteten Lösungsverfahren - das der **Heuristischen Wahl der Kombinatorik** - ist die Auswahl aus den vier Positions-Tupeln und die damit zusammenhängende Abbildung dezentraler und hybrider Ventilatoranordnungen sowie die Prüfung der Druckbilanz ebenfalls noch nicht möglich. Ansonsten werden die gleichen Parameter-Permutationen wie mit der Enumeration beschrieben.

Das Verfahren nutzt einen vordefinierten GA, welcher darauf beruht, aus zufällig gewählten Parameter-Permutationen und daraus berechneten Leistungsbedarfen Entscheidungskriterien zu entwickeln. Über Mechanismen der natürlichen Evolutionstheorie und den entwickelten Kriterien erzeugt der GA in jeder Generation entweder Parameter-Permutationen, die zu einem niedrigeren Leistungsbedarf führen als die der vorangehenden Generation oder übernimmt die Parameter-Permutation mit dem niedrigsten Leistungsbedarf der vorangehenden Generation. Damit führt die Berechnung von nur wenigen Leistungsbedarfen, unabhängig von der Modellgröße, zu dem minimalen Wert.

Hier wird ein stochastisch-heuristisches Lösungsverfahren beschrieben, welches nicht bei jeder Simulation exakt gleich erfolgt. Es ist möglich, dass die Lösungen nicht den global minimalen Leistungsbedarf enthalten. Gleichzeitig wächst die Rechenzeit jedoch nicht mit steigender Modellgröße, sodass das Potential dieses Verfahrens, Luftverteilungen eines großen Entscheidungsspielraumes abbilden zu können, sehr groß ist.

Für ein beschriebenes Auslegungsbeispiel mit realen Ventilator-Kennfeldern und ähnlichem Entscheidungsspielraum wie bei der Referenzluftverteilung kann mit der Optimierungsmethode Enumeration nachgewiesen werden, dass - wenn die Anzahl und die Position der Ventilatoren variabel wählbar sind - sowohl eine serielle Verschaltung als auch ein dezentraler Einsatz dieser zum minimalen Leistungsbedarf der Luftverteilung führen. Schon für Anlagen dieses Entscheidungsspielraumes kann im Vergleich zur zentralen Luftverteilung mit einem Ventilator eine Energieeinsparung von über 20 % erzielt werden. Mit der vorliegenden Arbeit wird daher gezeigt, dass die gleichzeitige Betrachtung aller oben genannter Ansätze notwendig ist, um den global minimalen Leistungsbedarf einer Luftverteilung zu erhalten. Es reicht hierfür nicht aus, diese unabhängig voneinander zu betrachten.

Weiterhin wird durch einen Vergleich der Faktorisierung und der Heuristischen Wahl der Kombinatorik mit der Enumeration belegt, dass sich alle Methoden zur Lösung von Optimierungsproblemen mit kleinem Entscheidungsspielraum eignen. Nimmt dieser zu, z.B. durch die Erweiterung der Netzgeometrie, ist die Enumeration zu zeitaufwendig und speicherintensiv. Für die beiden anderen Methoden besteht an dieser Stelle die Möglichkeit einer Erweiterung der Modellierung. Allerdings ist derzeit das Ausmaß des Entscheidungsspielraums für die bestehenden Methoden ohne Erweiterung noch wesentlicher limitiert, als das der Enumeration. So können dezentrale oder hybride Ventiltoranordnungen mit diesen beiden Methoden bisher noch nicht abgebildet werden.

Abschließend kann zusammengefasst werden, dass die vorliegende Arbeit für das Institut für Gebäudeenergetik, Thermotechnik und Energiespeicherung einen theoretischen Einstieg in die Optimierung der Fluidverteilung darstellt. Durch die entwickelten Methoden erfolgt der

Nachweis, dass Energie mit der diskret-kombinatortischen Optimierung eingespart werden kann.

8 Ausblick

Nachdem in der vorliegenden Arbeit die Energieeinsparung mit der diskret-kombinatorischen Optimierung nachgewiesen wird, kann zukünftig die Entwicklung verbesserter und erweiterter Optimierungsmethoden erfolgen. Mit den nachfolgenden Überlegungen sollen die dazu notwendigen Entwicklungskonzepte festgehalten werden.

Dazu wird zuerst beschrieben, wie die Modelle der betrachteten Optimierungsmethoden formal - etwa durch eine Programmcode-Optimierung - verbessert werden können. Danach wird erläutert, wie auch zeitlich variable Volumenströme mit der Methode Enumeration berücksichtigt werden können. Abschließend wird dargestellt, wie die Methoden Faktorisierung und Heuristische Wahl der Kombinatorik theoretisch Luftverteilungen mit einer sehr viel größeren Netzgeometrie und einem wesentlich erweiterten Entscheidungsspielraum abbilden können.

Verbesserung der formalen Form: Insgesamt können alle benutzerdefinierten MATLAB - Programme und Funktionen verbessert werden. Mit dem Button *Run and Time* des MATLAB Editors, kann eine zeitliche Analyse des formalen Modells stattfinden. Jeder Befehl und dessen Zeitbedarf wird aufgelistet. So sind neben den Interpolationsfunktionen *interp2 thin plate interp* auch Befehle wie *find* oder *eval* sehr zeitintensiv und können möglicherweise durch andere Arbeitsanweisungen ersetzt werden. Es besteht weiterhin die Möglichkeit, weniger Matrizen im Zwischenspeicher abzulegen. Leere beziehungsweise ungültige Matrixeinträge können, statt der Einträge *unzulässige Zahl* (NaN) oder *Null*, aus der Matrix gelöscht werden. Dazu müsste die Adressierung der Matrizen verändert werden. Die Größe der Matrizen und deren Speicherplatz kann so verringert werden, während die Grenzen der Methoden erweitert werden.

Zeitlich variable Volumenströme: Die Modellierung der Enumeration kann für eine vierte Variante des Auslegungsbeispiels um zeitlich variierende Volumenströme erweitert werden. Da diese die Raumluftqualität an den Bedarf anpassen, entstehen bei Abwesenheit der Nutzer reduzierte Volumenströme und weitere Energie kann eingespart werden. Hiermit kann der bisher vernachlässigte Freiheitsgrad 2 (die zeitliche Änderung der Volumenströme) berücksichtigt werden, welcher durch eine weitere Energieeinsparung für energieeffiziente Lüftungsanlagen stetig an Bedeutung gewinnt.

Bild Bild 8-1 zeigt die zeitliche Belegung des Bürogebäudes für das beschriebene Beispiel, anhand typischer Belegungsprofile und Tabelle B.2 aus DIN EN 15251. Es wird ersichtlich, dass beide Räume, einer der beiden oder keiner der beiden voll belegt sein können. Außerdem wird für jeden Raum ein minimaler Luftwechsel bei Nicht-Belegung vorgegeben. Daraus resultieren drei verschiedene Volumenstromzusammensetzungen, für welche, unter Berücksichtigung ihres Tagesstundenanteils, eine gesamte Lösung des Optimierungsproblems bestimmt werden soll.

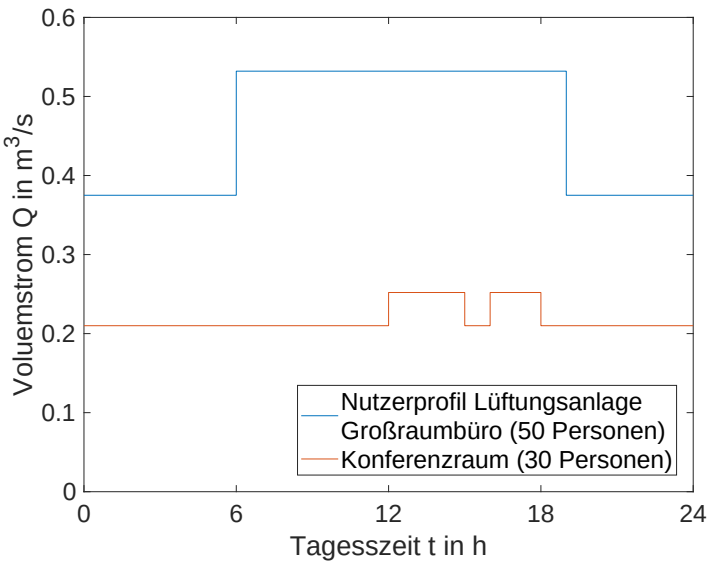


Bild: Bild 8-1 Schematische Darstellung von Tages-Belegungs-Profilen des Luftvolumenstrom-Bedarfes

Schematisch kann die Enumeration, wie in Bild Bild 8-2 dargestellt, angepasst werden. Im ersten Schritt findet die Dimensionierung der Luftverteilung statt. Dazu werden in Form der Auswahl der Positions-2-Tupel die Strang-Matrix festgelegt und eine Ventilator-Typ-Variation gewählt. Für jede dieser kombinierten Auswahlen findet die Auswahl der Drehzahl-Variation, für jede der Volumenstromzusammensetzungen statt. Die Leistungsbedarfe dieser drei werden entsprechend ihres täglichen Stundenanteils aufsummiert und damit der Tagesenergiebedarf bestimmt. Im letzten Schritt wird die Parameter-Permutation ausgewählt, welche unter Berücksichtigung aller Volumenstromzusammensetzungen den geringsten Energiebedarf pro Tag benötigt. So wird die Luftverteilung sowohl optimal dimensioniert, als auch optimal betrieben.

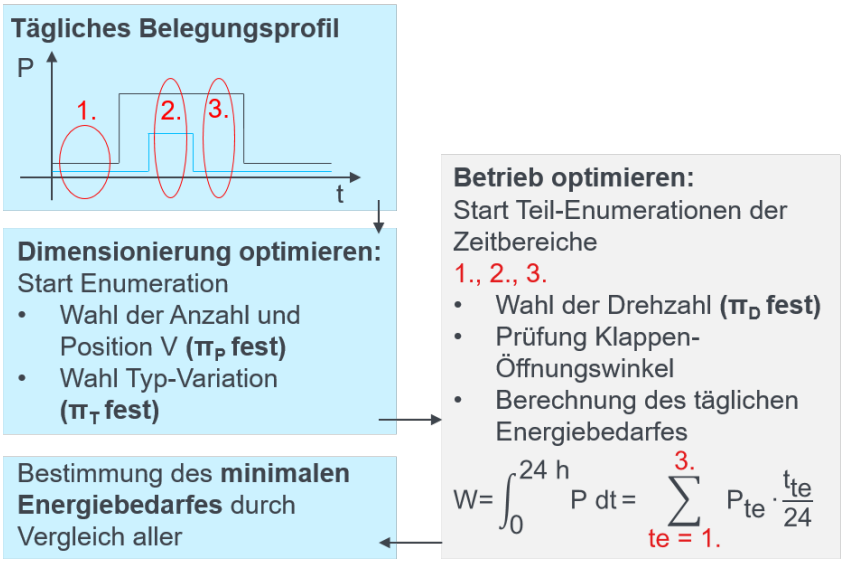


Bild: Bild 8-2 Schematischer Vorgang der Enumeration für zeitlich variable Volumenströme

Methoden-Erweiterungen zur Abbildung großer Netzgeometrien und Entscheidungsspielräume: Für eine Erweiterung der Faktorisierung wird nicht von maximal drei, sondern von

maximal neun Ventilatoren im Gesamtsystem der Referenzluftverteilung ausgegangen. Es sind nun in Bild Bild 4-1 alle dargestellten Ventilatoren möglich. Mit der Überlegung, das Lösungs-Kennfeld für eine Kombination von drei Ventilatoren, einmal für den Volumenstrom in Strangabschnitt I, einmal für den in Strangabschnitt II und einmal für den in Strangabschnitt III zu nutzen.

Im ersten Schritt einer zweiten Teilmodellierung müsste jeweils das Lösungs-Kennfeld in jedem der Strangabschnitte auf den entsprechenden Strangabschnittsvolumenstrom eingegrenzt werden. Ist also beispielsweise in Strangabschnitt I ein Volumenstrom von $4\,\mathrm{m}^3\,\mathrm{s}^{-1}$ festgelegt, können nur noch die Knotenpunkte der Form $(\sum_1^3 \Delta p|\ 4\,\mathrm{m}^3\,\mathrm{s}^{-1})$ ausgewählt werden. Es bleiben nun in jedem Lösungs-Kennfeld, in Abhängigkeit der Diskretisierung des Druckes, eine bestimmte Anzahl n an möglichen Knotenpunkten. Schematisch wird dies durch die blau markierten Bereiche im jeweiligen Lösungs-Kennfeld in Bild Bild 8-3 dargestellt.

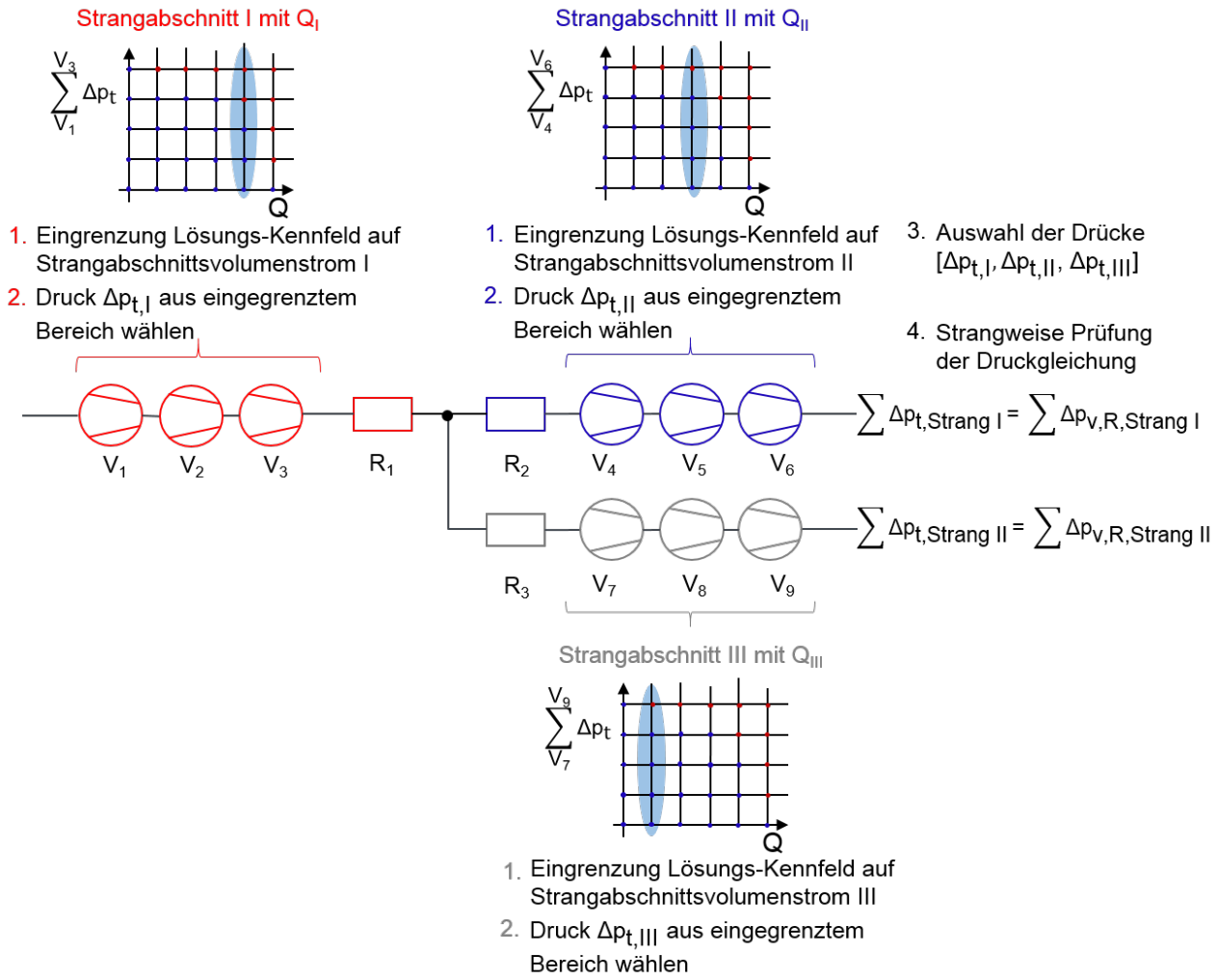


Bild: Bild 8-3 Schematische Darstellung einer Erweiterung der Modellierung der Faktorisierung

Im Anschluss (Schritt 2 in Bild Bild 8-3) wird in jedem Strangabschnitt einer der verbleibenden Knotenpunkte ausgewählt. Genauer ausgedrückt, wird ein Druckaufbau in jedem Strangabschnitt ausgewählt. Die Kombination der ausgewählten Drücke kann wiederum als Variation mit Wiederholung bezeichnet werden. Angenommen es stehen an jedem der drei Strangabschnitte k, $n = 5$ Drücke zur Auswahl, gibt es $n^k = 125$ Variationen, die Drücke der Strangabschnit-

te in der Form [$\Delta p_{t,I}$, $\Delta p_{t,II}$, $\Delta p_{t,III}$] miteinander zu kombinieren. Dabei muss immer die Summe des Druckaufbaus in Strangabschnitt I und II mindestens den Druckabfall in Strang I kompensieren und die Summe des Druckaufbaus in Strangabschnitt I und III mindestens den Druckabfall in Strang II. Ziel ist es also mit der entsprechenden Variation die Druckbilanz in beiden Strängen gleichzeitig zu erfüllen. Anschließend müsste eine Teilenumeration aller Variationen stattfinden, welche dieses Ziel erreichen.

Da an jedem Knotenpunkt der Lösungs-Kennfelder bereits der minimale Leistungsbedarf und die zugehörige Typ-Variation sowie die Drehzahleinstellung und die Anzahl der Ventilatoren hinterlegt ist, kann für jede in dieser Teilmodellierung vorliegenden Variation der Leistungsbedarf ausgelesen werden, ohne diesen neu berechnen zu müssen. Die Summe aller Leistungsbedarfe der gewählten Knotenpunkte ist der Leistungsbedarf der gesamten Luftverteilung. So ist die Variation mit dem minimalen gesamten Leistungsbedarf die Lösung des Optimierungsproblems. Auf diese Art und Weise kann die Netzgeometrie beliebig erweitert und das globale Leistungsminimum der gesamten Luftverteilung gefunden werden. Um diesen Vorgang formal abzubilden, sind einige weitere Arbeitsanweisungen zu formulieren.

Weiterhin ist zu beachten, dass die erforderlichen Strangabschnittsvolumenströme in der Mitte eines Kontrollkennfeldes liegen können. In diesem Fall kann nur näherungsweise eine Aussage zur Lösung gemacht werden, weshalb eine weitaus feinere Diskretisierung des Lösungs-Kennfeldes angestrebt wird.

Heuristische Wahl der Kombinatorik: Die Simulationsergebnisse der Heuristik stimmen für die Evaluationsluftverteilungen sehr gut mit denen der Enumeration überein. Außerdem stößt die Modellierung der Heuristik, wie in Kapitel 5.3.3 erläutert, erst für wesentlich größere Luftverteilsysteme an die Grenzen des physikalischen Arbeitsspeichers oder der Rechner-Laufzeit, als bei den beiden anderen Methoden. Aus diesem Grund ist dies die Optimierungsmethode mit dem größten Potential. Sie sollte daher um die strangweise Prüfung der Druckgleichung und um die Erzeugung der Strang-Matrix-Kombinationen erweitert werden.

Die Netzgeometrie (Strangabschnitt und Strang) kann dazu beispielsweise durch einen weiteren Parameter x pro Ventilator variieren. Insgesamt stehen so zwei Parameter pro Ventilator für die Positionierung zur Verfügung. Die Betrachtung der Position kann analog der Positions-n-Tupel der Enumeration erfolgen. Die Zielfunktion muss dann entsprechend angepasst werden. So wird beispielsweise mit dem ersten x auf den Volumenstrom in Strang I der Referenzluftverteilung zugegriffen, mit dem zweiten auf den in Strang II. Die Berechnung des Druckaufbaus, des Wirkungsgrades sowie des Leistungsbedarfes kann analog der in Kapitel 5.1.2 beschriebenen erfolgen. Neben der Berechnung von nur wenigen Parameter-Permutationen ist ein wesentlicher Vorteil gegenüber der Enumeration, dass nicht eine große Anzahl an Zwischenergebnissen, sondern nur die beste gefundene Lösung gespeichert wird. Es kann vermutet werden, dass hiermit eine sehr große Anzahl an Strängen betrachtet werden kann. Anstelle der Prüfung des minimalen Druckaufbaus der Enumeration, kann die Modellierung an der gleichen Stelle des formalen Modells um die Prüfung des Klappen-Öffnungswinkels nach Kapitel 5.1.2 erweitert werden.

Da der hier verwendete Algorithmus in der Lage ist, gemischt-ganzzahlige Probleme zu lösen, kann zukünftig außerdem die Drehzahl in einem kontinuierlichen Zahlenraum betrachtet werden. So ist es möglich, den global minimalen Druckaufbau (mit zugehörigem Leistungsbe-

darf) zu finden, welcher ausschließlich den Druckabfall der Kanalstücke überwinden muss. Der Einsatz einer zusätzlichen Klappe kann entfallen. Formal muss dazu der Zielfunktion statt der Anzahl der Drehzahleinstellungen $numn$ die kontinuierliche definierte Drehzahl n übergeben werden.

Ferner kann eine Optimierung der Heuristik untersucht werden. Das globale Minimum von Problemen kleiner Modellgrößen wird mit dem GA unter Verwendung des Problemwissens sehr schnell gefunden. Für wachsende Modellgrößen ist es jedoch möglich, dass die Lösung wesentlich vom globalen Minimum abweicht. Daher kann es möglicherweise sinnvoll sein, den GA mit einer lokalen Suche zu verknüpfen, sodass ein hybrider Algorithmus vorliegt. Die Suche verändert dabei die Parameter des Lösungsvektors nur geringfügig. Statt zeitaufwendig die Lösung der Zielfunktion für jede Veränderung zu berechnen, wird nur die Differenz zum vorherigen Zielfunktionswert berechnet. Dieser Vorgang wird so oft wiederholt, bis das globale Minimum gefunden ist [**UniversitaetTuebingen**].

Es bleibt festzuhalten, dass die Optimierung der Fluidverteilung Gegenstand weiterer Betrachtungen sein sollte. Ein hohes Entwicklungspotential bietet vor allem die Modellierung von großen Kanalnetzgeometrien und Entscheidungsspielräumen.

9 Literaturverzeichnis

10 Anhang

10.1 Schema Parameter-Permutationen

Mit Hilfe der Variations- und Kombinationsmatrizen aus Kapitel 4.1.2 können die 1107392 Parameter-Permutationen erzeugt werden. Die Reihenfolge der Anordnung wird immer eingehalten ([Auswahl π_P, Auswahl π_T, Auswahl π_D]). Im ersten Schritt wird die Auswahl π_P iteriert, bis alle möglichen Kombinationen der Positions-Tupel genau einmal vorkommen. Die beiden anderen Auswahlen bleiben in diesem Schritt konstant. Im zweiten Schritt wird für jede Iteration des ersten Schritts die Auswahl π_T iteriert. Die Drehzahl-Variation bleibt dabei konstant. Im dritten Schritt wird diese für jede der bisher aufgetretenen Anordnung von Auswahl π_P und Auswahl π_T, bis jede Anordnung genau einmal vorkommt.

Parameter-Permutation	**Auswahl-Anordnung** 1. Iteration in π_P 2. Iteration in π_T 3. Iteration in π_D
π_1	$\pi_{P\,1}, \pi_{T\,1}, \pi_{D\,1}$
π_2	$\pi_{P\,2}, \pi_{T\,1}, \pi_{D\,1}$
…	$\pi_{P\,...}, \pi_{T\,1}, \pi_{D\,1}$
	$\pi_{P\,numπP}, \pi_{T\,1}, \pi_{D\,1}$
	$\pi_{P\,1}, \pi_{T\,2}, \pi_{D\,1}$
	$\pi_{P\,2}, \pi_{T\,2}, \pi_{D\,1}$
	$\pi_{P\,...}, \pi_{T\,2}, \pi_{D\,1}$
	$\pi_{P\,numπP}, \pi_{T\,2}, \pi_{D\,1}$
	$\pi_{P\,1}, \pi_{T\,...}, \pi_{D\,1}$
	$\pi_{P\,2}, \pi_{T\,...}, \pi_{D\,1}$
	$\pi_{P\,...}, \pi_{T\,...}, \pi_{D\,1}$
	$\pi_{P\,numπP}, \pi_{T\,...}, \pi_{D\,1}$
	$\pi_{P\,1}, \pi_{T\,numπT}, \pi_{D\,1}$
	$\pi_{P\,2}, \pi_{T\,numπT}, \pi_{D\,1}$
	$\pi_{P\,...}, \pi_{T\,numπT}, \pi_{D\,1}$
	$\pi_{P\,numπP}, \pi_{T\,numπT}, \pi_{D\,1}$
	$\pi_{P\,1}, \pi_{T\,1}, \pi_{D\,2}$
	$\pi_{P\,2}, \pi_{T\,1}, \pi_{D\,2}$
	$\pi_{P\,...}, \pi_{T\,1}, \pi_{D\,2}$
	$\pi_{P\,numπP}, \pi_{T\,1}, \pi_{D\,2}$
	$\pi_{P\,1}, \pi_{T\,2}, \pi_{D\,...}$
	$\pi_{P\,2}, \pi_{T\,2}, \pi_{D\,...}$
	$\pi_{P\,...}, \pi_{T\,2}, \pi_{D\,...}$
	$\pi_{P\,numπP}, \pi_{T\,2}, \pi_{D\,...}$
	$\pi_{P\,1}, \pi_{T\,...}, \pi_{D\,numπD}$
	$\pi_{P\,2}, \pi_{T\,...}, \pi_{D\,numπD}$
	$\pi_{P\,...}, \pi_{T\,...}, \pi_{D\,numπD}$
$\pi_{numParPerm}$	$\pi_{P\,numπP}, \pi_{T\,...}, \pi_{D\,numπD}$

Bild: Bild 10-1 Schema zur Erzeugung aller Parameter-Permutationen. In drei Schritten wird zuerst die Positions-Tupel-Kombination iteriert, dann die Typ-Variation und zuletzt die Drehzahl-Variation.

10.2 Alle 13 Strang-Matrizen

Nachfolgend werden in Bild Bild 10-2 alle in Kapitel 5.1.2 erwähnten 13 Strang-Matrizen dargestellt, die für eine Referenzluftverteilung mit maximal drei Ventilatoren und zwei Strängen, unter Berücksichtigung der genannten Bedingungen möglich sind.

Die oberen fünf Strang-Matrizen sind besonders geeignet für gleiche Strangvolumenströme. Ist der benötigte Luftvolumenstrom in einer der beiden Stränge größer, ist in diesem Strang oft der Einsatz von einer höheren Anzahl an Ventilatoren sinnvoll (Vgl. unterste Strang-Matrix). Bisher können die Volumenströme beliebig gewählt werden. Sowohl der obere, als auch der untere Strang müssen daher den höheren Luftvolumenstrom transportieren können, weshalb immer zwei ähnliche Anordnungen möglich sind, deren Stränge gespiegelt genau der anderen Anordnung entsprechen (Vgl. letzte und vorletzte Strang-Matrix in Bild Bild 10-2). Wird im Vorfeld festgelegt, dass die Luftvolumenströme der Luftverteilung entsprechend ihrer Größe sortiert eingegeben werden, kann an dieser Stelle die Anzahl der Strang-Matrizen möglicherweise weiter reduziert werden. Diese Betrachtung bedarf jedoch einer weiteren Prüfung, welche im Rahmen dieser Arbeit nicht mehr stattgefunden hat.

ID-Matrix

$$\left[\begin{array}{ccc|cc|cc} 1 & 1 & 1 & 2 & 2 & 3 & 3 \end{array}\right]$$

Strang-Matrix

$$\left[\begin{array}{ccc|ccc|cc} 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{array}\right]$$

$$\left[\begin{array}{ccc|ccc|cc} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{array}\right]$$

$$\left[\begin{array}{ccc|ccc|cc} 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{array}\right]$$

$$\left[\begin{array}{ccc|ccc|cc} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{array}\right]$$

$$\left[\begin{array}{ccc|ccc|cc} 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{array}\right]$$

$$\left[\begin{array}{ccc|ccc|cc} 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{array}\right]$$

$$\left[\begin{array}{ccc|ccc|cc} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{array}\right]$$

$$\left[\begin{array}{ccc|ccc|cc} 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{array}\right]$$

$$\left[\begin{array}{ccc|ccc|cc} 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{array}\right]$$

$$\left[\begin{array}{ccc|ccc|cc} 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{array}\right]$$

$$\left[\begin{array}{ccc|ccc|cc} 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{array}\right]$$

$$\left[\begin{array}{ccc|ccc|cc} 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{array}\right]$$

$$\left[\begin{array}{ccc|ccc|cc} 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{array}\right]$$

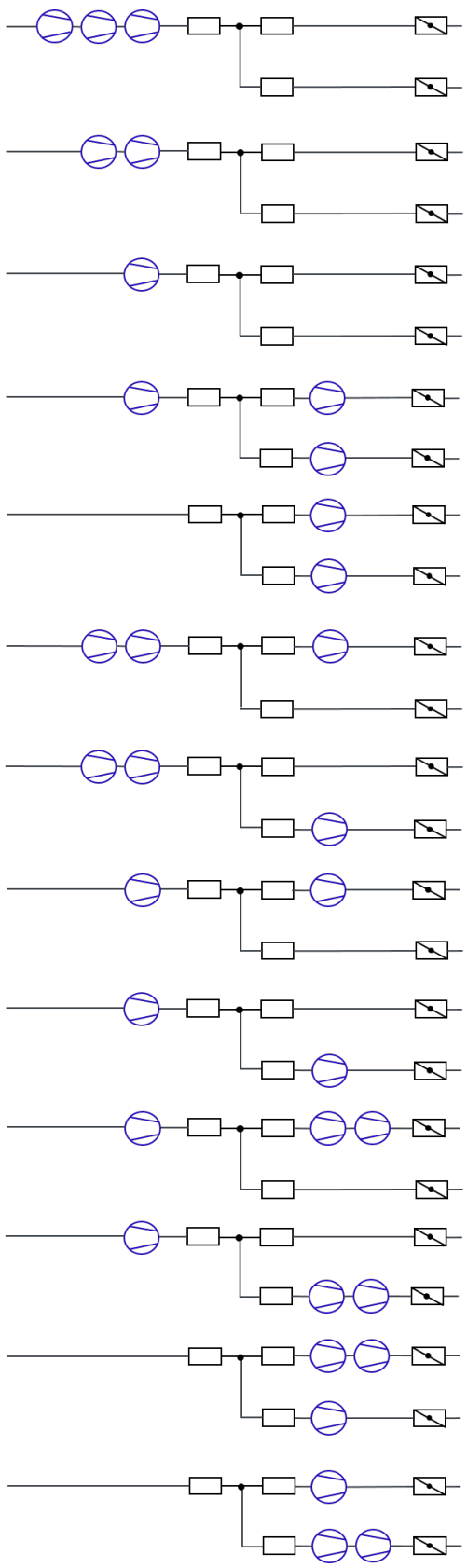


Bild: Bild 10-2 Darstellung aller 13 Strang-Matrizen, welche für die beschriebene Referenzluftverteilung zur Auswahl stehen.

10.3 Formale Modelle der Optimierungsmethoden

Die folgenden Kapitel enthalten die, mit der in MATLAB integrierten Programmiersprache formulierte, formalen Formen der Optimierungsmethoden. Die verwendeten Programme und Funktionen sind den Optimierungsmethoden zugeordnet. In den jeweiligen Hauptprogrammen wird kurz auf alle verwendeten Funktionen verwiesen.

Wird der Name der Funktion, einer zu übergebenden Variablen oder einer zurückzugebenden Variablen verändert, muss dieser sowohl an der Stelle des Funktionsaufrufs, als auch in der Funktion entsprechend angepasst werden.

10.3.1 Skripte Enumeration

Hauptprogramm

Enumeration_main.m

In diesem MATLAB-file erfolgt die Initialisierung der Luftverteilung. Außerdem werden alle Funktionen aufgerufen und miteinander kombiniert, entsprechend der schematischen Darstellung Bild 5-1.

```
1  %% Enumeration - Optimizationmethod 1 for reference air distribution
2  % Minimum of electrical power demand for a fluid distribution with a
3  % given number of elements, meshes and their volume flow.
4  % More description in ReadMe.txt
5  %
6  % Used functions:
7  % eta_all.m
8  % solveAlpha_numMeshEqualNumDamp.m
9  % permutation.m
10 % meshMatrix_test.m (uses nmultichoosek.m)
11 % saveData.m
12 %
13 % Parameters:
14 % number of fans (numFan)
15 % number of duct elements (numDuct)
16 % number of dampers (numDamp)
17 % number of meshes (numMesh)
18 % target flows for every mesh (targetflow(i))
19 % characteristics of Elements
20 % scale factor for fan characteristics (sacles pressure, changes curve
21 % shape) (scaleFan)
22 %
23 % Comments:
24 % i is in all loops equal to numMesh
25 % j is in all loops equal to numEl (or numFan)
26 %
27 % History:
28 % 2019 - 05 - 15        S.Lott
29 %                       Script created
30 % 2019 - 05 - 24        functions solveAlpha and elPower added
31 % 2019 - 06 - 26        function elPower not used anymore, funktions
32 %                       permutation.m and meshMatrix_test.m
33 %                       (uses nmultichoosek.m)added
34 %                       Skript is tested for a special number of elements
```

```
%                        and meshes (2 meshes, 2 dampers, 2-3 fan, 3 ducts,
%                        target flows until )
%%
clc
clear

%% Current Param - change here target flow, number of elements/meshes,
%  initialize mesh matrix

% set number of fan, duct and damper resistance
nF = 'Maximum number of fan (2-3) = ';

% nF = 9 out of memory problem!
% nF = 6 array exceeds maximum array size preference problem!
%        limit may take a long time and cause MATLAB to become unresponsive

numFan = input(nF);              % maximum number of fan
numDuct = 3;                     % temporary const, as a constant duct
                                 % structure is assumed
numDamp = 2;                     % total number of damper, in this
                                 % simulation constant

% initialize rotational speed and increment
N = 11;                          % increment rotational speed
n = linspace(0,100,N);           % rotational speed [%]

% initialize number of meshes and total elements
numMesh = 2;
numEl = numFan + numDuct + numDamp;

% initialize and set target flow in dependency of mesh
tf1 = 'Target flow mesh 1 [m^3/s] = ';
tf2 = 'Target flow mesh 2 [m^3/s] = ';

    % initialize targetflow with zeros and define the output flows
targetflow = zeros(1,numMesh)';

    % targetflow(1) = volume flow in mesh one and two [m^3/s]
targetflow(1) = input(tf1);
targetflow(2) = input(tf2);

tic
% initialize fan characteristic scale factors
scaleFan = [1,0.75,0.5,0.25];

% Initialize ID matrix (used in solveAlpha)
    % ID Matrix, where fan = 1, duct = 2 and damper = 3 are located in
    % mesh matrix
cID_fan = zeros(numMesh, numFan);
cID_fan(:,:) = 1;
cID_duct = zeros(numMesh, numDuct);
cID_duct(:,:) = 2;
cID_damp = zeros(numMesh, numDamp);
cID_damp(:,:) = 3;

cID = [cID_fan,cID_duct,cID_damp];
```

```
% initialize function handle electrical power fan
P  = @(eta,dp,q) dp*q*100/eta;

% get mesh matrix
[M, fanComb] = meshMatrix_test(numMesh, numEl, numDamp, numFan);

% fill volume flow struct
V = targetflow.*M;              % volume flow matrix (there is only a
                                % volume flow if the element is existing;
                                % for calculation of volume flow through
                                % element)
q = struct([]);                 % initialize volume flow struct (for better
                                % elementwise access)

% volume flow for each element
    % q(1,1).flow = total flow through element 1
    % and mesh matrix combination 1
for numComb = 1:length(M(1,1,:))
    for j = 1:numEl
        q(j, numComb).flow = sum(V(:,j,numComb));
    end
end
toc
%% Initialize pressure struct of characteristics
dp = struct([]);                % initialize pressure struct

% fill pressure struct with characteristic function handles (analytical
% description) for each element in each mesh matrix combination
for numComb = 1:length(M(1,1,:))      % loop for mesh matrix permutation
    for i = 1:numMesh                 % loop for numMesh
        for j = 1:numEl               % loop for numElement
            if M(i, j, numComb) == 1  % element is existing
               switch  cID(i, j)      % element is fan, duct or damper
                  case 1
                  % Fan
                    % characteristic fan, analytical description
                    % --> adjust eta_all to fan characteristic!
                   dp(i,j,numComb).syst = @(j,n,scaleFan) scaleFan*10*n-...
                       (scaleFan*10*n/((n/20)^2)*q(j,numComb).flow.^2);

                   case 2
                  % Duct
                    % characteristic duct, analytical description
                    % factor adjustment: 100000 = very steep, 100 = enough
                    % pressure drop for varying fan characteristics
                    dp(i,j,numComb).syst = @(j) -(100)*q(j,numComb).flow^2;

                   case 3
                  % Damper
                    % characteristic damper, analytical description
                    % factor adjustment: 300000 = steep can only find alpha
                    % for low target flows; 300 = less steep, find alpha
                    % (pressure drops) for almost all target flows and
                    % pressure build ups
                    dp(i,j,numComb).syst = @(alpha, j) -300/(alpha^2/90)*...
```

```
                    q(j,numComb).flow.^2;
                end
            end
        end
    end
end
toc
%% Calculation of electical power demand for every permutation of
%  rotational speed and characteristic, for every fan combination.
%  Find minimum with an physically possible damper angle alpha.

    %% Get electical power demand matrix
    % get power demand P_allFan for all possible parameter-permutations
    [P_allFan,P_fan,dpAllFan,nPerm,charPerm,eta] = permutation(P, ...
    numFan,numMesh, n, M, q,scaleFan, dp);
toc
    %% Solve equation system to get alphas (function: solveAlpha), delete
    % power demand and parameter-permutation, if no physical alpha can
    % be found
        % initialize start arrays
    alphas  = struct([]);
    P_allFan_orig = P_allFan;

    % loop until power demand minimum with physically possible damper
    % angles is find
    while isempty(alphas) || any(structfun(@isempty, alphas))

        % get power minimum of all permutations and combinations and
        % related pressure values
        Opt_PallFan = min(P_allFan_orig(:));

        % get position of power minimum
            % nVar = rotaional speed permutation, charVar =
            % characteristic permuation, fanCom = fan combination, with
            % these values it is possible to get a postion in the power
            % demand matrix
        [nVar,charVar,fanCom] = ind2sub(size(P_allFan_orig),find(...
        P_allFan_orig==Opt_PallFan));
            if isempty(nVar) || isempty(charVar) || isempty(fanCom)
            % for no solution found (all power demands are equal or nan)
            dp_fanOpt = 0;
            eta_fanOpt = 0;
            else
            % for more than one minimum, take the first minimum
            dp_fanOpt = dpAllFan(:,:,nVar(1),charVar(1),fanCom(1));
            eta_fanOpt = eta(:,nVar(1),charVar(1),fanCom(1));
            end

        % get damper angles from function - solveAlpha
        try
        [alphas, eqn, fullSym] = solveAlpha_numMeshEqualNumDamp(numMesh,...
        numEl, numFan, numDuct, numDamp, M, dp, cID, dp_fanOpt, fanCom(1));

        % display error message if no alpha was found for several reasons
        catch ME
            if (strcmp(ME.identifier,'MATLAB:badsubscript'))
```

```
                msg = ME.message;
                disp(msg);
                % bad subscript exception handling goes here
                disp(['Target flows are in total to high.'...
                'Chose lower target flow or other fan characteristic.']);
            end
        end

        % set power minimum to NaN to get next higher minimum
        % if no alpha was found
        if isempty(alphas)== 1 || any(structfun(@isempty, alphas)) == 1
        P_allFan_orig(nVar,charVar,fanCom) = nan;
        end

    end
    toc

    %% Get minimum of power matrix and plot/display
    % display used mesh matrix and damper angles
    dispLine = ['|';'|'];

    disp('Maschen-Matrix (optimale Ventilator Kombination)')
    disp([num2str(M(:,1:numFan,fanCom(1))), num2str(dispLine(:,1)),...
    num2str(M(:,(numFan+1):(numEl-numDamp),fanCom(1))),...
    num2str(dispLine(:,1)),num2str(M(:,(numEl-numDamp+1):numEl),fanCom(1))])
    disp('Ventilator = 1, Kanalstueck = 2, Klappe = 3')
    disp([num2str(cID(1,1:numFan)), num2str(dispLine(1,1)),...
    num2str(cID(1,(numFan+1):(numEl-numDamp))),num2str(dispLine(1,1)),...
    num2str(cID(1,(numEl-numDamp+1):numEl))])

    P_opt_text = {'Drehzahl Variation' 'Kennlinien Variation' ...
    'Optimum P_el'};
    P_opt_value = {num2str(nPerm(nVar(1),:)),...
    num2str(charPerm(charVar(1),:)), Opt_PallFan};
    P_opt = [P_opt_text; P_opt_value];
    % structfun returns type double of each field of the struct alphas
    Alphas = structfun(@double, alphas, 'uniformoutput', 0);

    disp(P_opt)
    disp(Alphas)

    %% Save data as property structs (function)
    saveData(numEl,numFan,numDamp,M,cID,alphas,Alphas,fanCom(1),...
    nPerm,charPerm,nVar(1),charVar(1),targetflow,Opt_PallFan,dp_fanOpt,...
    eta_fanOpt, dpAllFan, P_allFan,fullSym)
    toc
    %% Plot graph
    switch numFan
        % plot for a possible number of fan = 2
        case 2
        % reshape power demand matrix to N*N matrix
        P_plot = reshape(P_allFan_orig,[N,N,length(charPerm),...
        length(M(1,1,:))]);

        % plot (the first) optimal parameter-permuatation and power demand
        set(figure, 'Units', 'normalized', 'Position', ...
```

```
        [0.3, 0.35, 0.35, 0.4]);
        surf(n,n, P_plot(:,:,charVar(1),fanCom(1)))
            xlim([0 100])
            ylim([0 100])
            %zlim([0 1600])

            title(['Kennlinienvariation:     ',...
            num2str(charPerm(charVar(1),:)), ...
            '     Ventialtorkombination:     ',...
            num2str(scaleFan(fanComb(fanCom(1),1))),'    ',...
            num2str(scaleFan(fanComb(fanCom(1),2)))])
            xlabel('Drehzahl Ventilator 1 [1/s]')
            ylabel('Drehzahl Ventilator 2 [1/s]')
            zlabel(sprintf...
            ('Elektrischer Leistungsbedarf\nFluidverteilung [W]'))
            view(30,30)
            grid on

            xh = get(gca,'XLabel');                 % Handle of the x label
            set(xh, 'Units', 'Normalized')
            pos = get(xh, 'Position');
            set(xh, 'Position',pos.*[0.8,-0.1,1],'Rotation',-10)
            yh = get(gca,'YLabel');                 % Handle of the y label
            set(yh, 'Units', 'Normalized')
            pos = get(yh, 'Position');
            set(yh, 'Position',pos.*[1.05,-0.4,1],'Rotation',30)
        % plot for a possible number of fan = 3
        case 3
            % reshape power demand matrix to N*N*N matrix
            P_plot = zeros(N,N,N,length(charPerm),length(M(1,1,:)));

            for fComb = 1:length(M(1,1,:))
                for cPerm = 1:length(charPerm)
                P_plot(:,:,:,cPerm,fComb) = reshape(...
                P_allFan_orig(:,cPerm,fComb),[N,N,N]);
                end
            end

            % plot optimal characteristic permuatation and fan combination
            set(figure, 'Units', 'normalized', 'Position', ...
            [0.3, 0.35, 0.35, 0.4]);
            nz = reshape(nPerm(:,3),[N N N]);
            get_nzIndex = find(nz(1,1,:) == nPerm(nVar,3));
            if size(get_nzIndex,1)> 1
                get_nzIndex = get_nzIndex(1);
            end

            surf(n,n,nz(:,:,get_nzIndex),P_plot(:,:,get_nzIndex,...
            charVar(1),fanCom(1)));
            colorbar
            xlim([0 100])
            ylim([0 100])
            zlim([0 100])

title(sprintf(...
['Elektrischer Leistungsbedarf Fluidverteilung [W]\nKennlinienvariation:  '...
```

```
,num2str(charPerm(charVar(1),:)), '     Ventialtorkombination:     ',...
num2str(scaleFan(fanComb(fanCom(1),1))),'    ',...
num2str(scaleFan(fanComb(fanCom(1),2))),'    ',...
num2str(scaleFan(fanComb(fanCom(1),1)))]))
xlabel('Drehzahl Ventilator 1 [1/s]')
ylabel('Drehzahl Ventilator 2 [1/s]')
zlabel('Drehzahl Ventilator 3 [1/s]')
view(30,30)
grid on

            xh = get(gca,'XLabel');                 % Handle of the x label
            set(xh, 'Units', 'Normalized')
            pos = get(xh, 'Position');
            set(xh, 'Position',pos.*[0.8,-0.1,1],'Rotation',-10)
            yh = get(gca,'YLabel');                 % Handle of the y label
            set(yh, 'Units', 'Normalized')
            pos = get(yh, 'Position');
            set(yh, 'Position',pos.*[1.05,-0.4,1],'Rotation',30)
    end
```

Genutzte Funktionen

numltichoosek.m

Erzeugen der Ventilator-Positions-Tupel-Kombinationen

```
%% Function: nmultichoosek - returns number and values of mesh matrix
%  combinations in depedency of total number of fan
        % used in meshMatrix.m
        % online source: 'https://stackoverflow.com/questions/28284671/
        % generating-all-combinations-with-repetition-using-matlab'

function combs = nmultichoosek(values, k)
if numel(values)==1
    n = values;
    combs = nchoosek(n+k-1,k);
else
    n = numel(values);
    combs = bsxfun(@minus, nchoosek(1:n+k-1,k), 0:k-1);
    combs = reshape(values(combs),[],k);
end
```

meshMatrix_test.m

Erzeugen der Strang-Matrix unter Berücksichtigung der Bedingung - ein Ventilator pro Strang

```
function[M, fanComb] = meshMatrix_test(numMesh, numEl, numDamp, numFan)
%% Initialize mesh matrix
    % the first entries (until numFan) of mesh matrix are describing the
    % fan positions, following the duct positions and the last entries are
    % descrining the damper positions (see cID).

    %% Initialize fan 2-tupel  -  Only valid for 2 meshes
    %  tupel means one couple of boolean values for fans in mesh matrix
```

```
9
10      vv = ones(numMesh,1);        % [1 1]
11      nn = zeros(numMesh, 1);      % [0 0]
12      vn = nn;
13      vn(1,1) = true;              % [1 0]
14      nv = nn;
15      nv(numMesh,1) = true;        % [0 1]
16
17      tupel = [vv, nn, vn, nv];    % all possible fan tupel for 2 meshes
18
19      % get all fan position combinations with repetition
20      fanComb = nmultichoosek(1:length(tupel),numFan);
21
22      % initialize mesh Matrix for fan, duct and damper positions
23      fanPos = zeros(numMesh, numFan, length(fanComb)); % variable fan pos.
24      ductPos = [1 1 0;1 0 1];                      % constant duct position
25      dampPos = eye(numMesh, numDamp);            % constant damper position
26
27      M = zeros(numMesh, numEl, length(fanComb));
28
29      % get mesh matrix for all possible fan position combinations
30      for numComb = 1:length(fanComb)
31          for j = 1:numFan
32              % get fan position combinations
33              fanPos(:,j,numComb) = tupel(:, fanComb(numComb,j));
34              % combine all element positions to mesh matrix
35              M(:,:,numComb) = [fanPos(:,:,numComb),ductPos,dampPos];
36          end
37      end
38
39      % get all mesh matrix with at least one fan per mesh (condition)
40          % initialize matrix to delete not valid mesh matrices
41      deleteM =  zeros(1,length(fanComb));
42      deleteM_0damp = zeros(1,length(fanComb));
43      sumM = zeros(1,numFan,numComb);
44          % loop for fan position combinations to find not valid mesh
45          % matrices and delete them
46      for numComb = 1:length(fanComb)
47          sumM(:,:,numComb) = sum(M(:,1:numFan,numComb));
48
49          for nM = 1:numMesh
50              % get fan combinations where condition is not fulfilled
51              if sum(M(nM,1:numFan,numComb)) == 0
52                  deleteM(numComb) = numComb;
53              end
54
55              % for the possibility of zero dampers in the system this
56              % if-condition is needed
57                  % get fan combinations where all fan in both meshes
58              if numDamp == 0
59                  if sum(sum(M(:,1:numFan,numComb))) == numMesh*numFan
60                  % for different target flow in both meshes this mesh matrix
61                  % configuration will never be a solution
62                  deleteM_0damp(numComb) = numComb;
63                  elseif sum(sum(M(:,1:numFan,numComb))) == numMesh && ...
64                  sum(M(:,1,numComb))== numMesh
```

```
                deleteM_0damp(numComb) = numComb;
                elseif sumM(:,1,numComb) == 2 && sumM(:,2,numComb) == 2 &&...
                sumM(:,3,numComb) == 0
                deleteM_0damp(numComb) = numComb;
                end
            end
        end
    end

    % get fan position combinations where condition is not fulfilled
    % without zeros
    endDeleteM = deleteM(deleteM ~= 0);

    % delete mot valid fan position combinations
    M(:,:,endDeleteM) = [];

    % delete fan position combination which is not possible for
    % zero dampers in the system and different target flow in both meshes
    if numDamp == 0
       endDeleteM_0damp = deleteM_0damp(deleteM_0damp ~= 0);
       M(:,:,endDeleteM_0damp) = [];
    end

end
```

eta_all.m

Definition des Wirkungsgrad-Gitters zur Nutzung einer MATLAB 2D Interpolation. Diese Funktion wird für alle Optimierungsmethoden verwendet.

```
%% Function: eta_all - get efficiency eta via 2 D interpolation
%  for different fan types (scaleFan), pressures and volume flows

function [eta_out] = eta_all(dp, q, scaleFan)
x = scaleFan;
eta_out = interp2([x*0,x*200,x*400,x*600,x*800,x*1000],[0,1,2,3,4,5],...
    [80,70,60,50,40,30;70,80,70,60,50,40;60,70,80,70,60,50;...
    50,60,70,80,70,60;40,50,60,70,80,70;30,40,50,60,70,80], dp, q);

    % set efficiency to NaN for upper and lower boundary of valid pressures
    %(only valid in the first quadrant and depending on scaleFan)
    % and for pressures of value NaN
    if x == 1
        if dp > 1000 || dp <= 0 || isnan(dp) == 1
            eta_out = nan;
        end
    elseif x == 0.75
        if dp > 750 || dp <= 0 || isnan(dp) == 1
            eta_out = nan;
        end
    elseif x == 0.5
        if dp > 500 || dp <= 0 || isnan(dp) == 1
            eta_out = nan;
        end
```

```
    elseif x == 0.25
        if dp > 250 || dp <= 0 || isnan(dp) == 1
            eta_out = nan;
        end
    end
end
```

permutation.m

Erzeugen der Drehzahl- und Typ-Variationen des Ventilators und Berechnung von Druckaufbau, Wirkungsgrad und Leistungsbedarf aller Parameter-Permutationen

```
%% Function:permutation - get all parameter-permutations,
%  corresponding pressures, efficienies and power demands

function [P_allFan,P_fan,dpAllFan,nPerm,charPerm,eta] =...
    permutation(P,numFan, numMesh, n, M, q,scaleFan, dp)

% initialize length of each permutation (number of fan which shell be
% combined)
K = numFan;

% create all possible permutations (with repetition)
% of rotational speed n for a fan combination
cellSpeed = cell(K, 1);                          % preallocate a cell array
[cellSpeed{:}] = ndgrid(n);                      % create K grids of values
nPerm = cellfun(@(n){n(:)}, cellSpeed);   % convert grids to column vectors
nPerm = [nPerm{:}];            % obtain all permutations of rotaional speed

    % numFan = 9 is responding a out of memory problem at this point,
    % because the number of rotational speed permutations are rising to
    % 11^9 = 2357947691

% create all possible permutations (with repetition)
% of fan types for a fan combinatorics
cellChar = cell(K, 1);                           % preallocate a cell array
[cellChar{:}] = ndgrid(scaleFan);                % create K grids of values
charPerm = cellfun(@(scaleFan){scaleFan(:)}, ...
cellChar);                                  % convert grids to column vectors
charPerm = [charPerm{:}];            % obtain all permutations of fan types

    % numFan = 9 is responding the number of fan type permutations of
    % 4^9 = 262144

% initialize arrays
dpAllFan = zeros(numMesh,numFan,length(nPerm),length(charPerm),...
           length(M(1,1,:)));
eta = zeros(numFan,length(nPerm),length(charPerm),length(M(1,1,:)));
P_fan = zeros(numFan,length(nPerm),length(charPerm),length(M(1,1,:)));
P_allFan = zeros(length(nPerm),length(charPerm),length(M(1,1,:)));

tic
% loop for all elements in all meshes, in all rotational speed ...
% permutations, in all characteristic permutations and all mesh matrix
```

```
% permutations
for numComb = 1:length(M(1,1,:))
    for chn = 1:length(charPerm)
        for vn = 1:length(nPerm)
            for i = 1:numMesh
                for j = 1:numFan
                    if M(i,j,numComb) ==1
                    % get pressure build up for every fan
                    dpAllFan(i,j,vn,chn,numComb) = ...
                    dp(i,j,numComb).syst(j,nPerm(vn,j),charPerm(chn,j));
                    dpAllFan(dpAllFan(:)<0)=0;

                    % get efficiency
                        % calculation of efficiency for one fan in relation
                        % to number of meshes in which fan is used (without
                        % relation, pressure for every mesh would sum up)
                    eta(j,vn,chn,numComb) = eta_all(sum(dpAllFan(:,j,...
                    vn,chn,numComb))/sum(M(:,j,numComb)),...
                    q(j,numComb).flow,charPerm(chn,j));

                    % get electrical power demand
                    P_fan(j,vn,chn,numComb) = P(eta(j,vn,chn,numComb),...
                    sum(dpAllFan(:,j,vn,chn,numComb))/...
                    sum(M(:,j,numComb)),q(j, numComb).flow);

                    end
                end
            end
            % sum up power demand for one rotational speed, one
            % characteristic permutation and one mesh matrix permutation
            P_allFan(vn,chn,numComb) = sum(P_fan(:,vn,chn,numComb));
        end
    end
end

% set all negativ values and zeros to NaN
    % only positive (greater than zero) power demands are considered
P_allFan(P_allFan(:)==0)=nan;
P_allFan(P_allFan(:)<0)=nan;
end
```

saveData.m

Speichern des minimalen Leistungsbedarfs mit zugehöriger Parameter-Permutation für simulierten Volumenstrom

```
%% Function: save Data - for different tagret flows
function[meshMatrix, IDMatrix] = ...
    saveData(numEl,numFan,numDamp,M,cID,alphas,Alphas,...
    fanCom,nPerm,charPerm,nVar,charVar,targetflow,Opt_PallFan,dp_fanOpt,...
    eta_fanOpt, dpAllFan, P_allFan, fullSym)
% initialize arrays
meshMatrix = [M(:,1:numFan,fanCom(1)),M(:,(numFan+1):(numEl-numDamp),...
              fanCom(1)),M(:,(numEl-numDamp+1):numEl)];
```

```
IDMatrix = [cID(1,1:numFan),cID(1,(numFan+1):(numEl-numDamp)),...
            cID(1,(numEl-numDamp+1):numEl)];

%% get tables with simulation results
% mesh matrix and ID
T1 = table([{'Masche I'};{'Masche II'};...
    {'MatrixID, Ventilator = 1, Kanalstueck = 2, Klappe = 3'}],...
    [meshMatrix(1,:);meshMatrix(2,:);IDMatrix],'VariableNames',...
    {'ParameteterName' 'AufbauMaschenmatrix'});

% rotational speed and characteristic permutations, target flows
T2 = table([{'V_Drehzahlvariation'};{'V_Kennlinienvariation'}],...
    [nPerm(nVar(1),:); charPerm(charVar(1),:)], 'VariableNames',...
    {'ParameteterName' 'LoesungsparameterVentilator'});
T3 = table({'M_Volumenstrom Masche I und II'},[targetflow(1), ...
    targetflow(2)],'VariableNames',...
    {'ParameteterName' 'LoesungsparameterMaschen'});

% damper angles
if isempty(alphas)== 0 || any(structfun(@isempty, alphas)) == 0
T4 = table({'M_Klappenstellwinkel Masche I und II'},...
    [Alphas.alpha1, Alphas.alpha2], 'VariableNames',...
    {'ParameteterName' 'LoesungsparameterMaschen'});
elseif isempty(alphas)== 0 || isempty(Alphas.alpha1) == 0
T4 = table({'M_Klappenstellwinkel'},[Alphas.alpha1, NaN], ...
    'VariableNames',{'ParameteterName' 'LoesungsparameterMaschen'});
else
T4 = table({'M_Klappenstellwinkel'},[NaN, NaN], 'VariableNames',...
    {'ParameteterName' 'LoesungsparameterMaschen'});
end

% optimal (energy efficient) power demand
T5 = table({'Leistungsbedarf in energetischem Optimum'},...
    Opt_PallFan, 'VariableNames',{'ParameteterName' 'Loesungswert'});

% get pressure and  efficiency for all fan
T6 = table([{'V_anliegenderDruck Masche I'};...
    {'V_anliegenderDruck Masche II'};{'V_Wirkungsgrad'}],...
    [dp_fanOpt(1,:);dp_fanOpt(2,:);eta_fanOpt'], 'VariableNames',...
    {'ParameteterName' 'LoesungsparameterVentilator'});

%% Get one table out of all results
T_1 = outerjoin(T2,T3,'MergeKeys', true);
T_2 = outerjoin(T_1,T4,'MergeKeys', true);
T_3 = outerjoin(T_2,T5,'MergeKeys', true);
T_4 = outerjoin(T_3,T5,'MergeKeys', true);
T_5 = outerjoin(T_4,T6,'MergeKeys', true);
T = outerjoin(T_5,T1,'MergeKeys', true);

%% save workspace variables and text file with results
filename_txt = sprintf('PressureAndPower%03g_%03g_%01gFan.txt',...
    round(targetflow(1),1,'significant'),...
    round(targetflow(2),1,'significant'),numFan);
filename_mat = sprintf('PressureAndPower%03g_%03g_%01gFan.mat',...
    round(targetflow(1),1,'significant'),...
    round(targetflow(2),1,'significant'),numFan);
```

```
save(filename_mat, 'dpAllFan','P_allFan', 'Opt_PallFan','fullSym');
writetable(T,filename_txt);
end
```

10.3.2 Skripte Faktorisierung

Hauptprogramm

factorization_main.m

In diesem MATLAB-file erfolgt die Initialisierung der Luftverteilung. Die Berechnung der Leistung, des Wirkungsgrades, sowie der Drehzahlen erfolgt für aller Punkte (Druckaufbau|Volumenstrom) eines Ventilators. Dies erfolgt für alle Ventilator-Typen. Außerdem werden alle Funktionen aufgerufen und miteinander kombiniert, entsprechend der schematischen Darstellung Bild 5-6.

```
%% Get solution-characeristics for three fan in a system
% This script generates arrays with all parameter-permutations and
% corresponding power demand for a combination of three fan at each point
% (total pressure build-up|volume flow). Function
% getPropertyStruct.m gets only the minimum power demand and
% parameter-permutation for each point.
%
% The properties are saved as csv-file with the name
% 'prop_volumeflow_pressure build-up.csv'. The name of the saved variable
% is for each row declared on the left side of each csv-file.
%
% More description in ReadMe.txt
%
% Used functions:
% eta_all.m
% getFanCombination.m
% getPropertyStruct.m
% saveTables.m
%
% Parameters:
% fan characteristic scale factor (scale) - at the moment there are
% four different fan characteristics
% fan characteristic scale factor index (iscale) - increase with increased
% number of scale
% discretisation pressure (dp) - possible discretization 10 or greater
% discretisation volume flow (dq) - possible discretization 0.5 or greater
% number of fans (numFan) - only 3 is valid
%
% Comments:
%
% History:
% 2019 - 08 - 02        S.Lott
%                       Script created
% 2019 - 08 - 23        Functions created
% 2019 - 08 - 26        Added 'onlyOneValuePerPoint.m'
%%
clear

%% Initialization
% initialize number of fan in a sub-mesh
```

```
numFan = 3;

% initialize vectors
scale = [0.25 0.5 0.75 1.0];  % fan characteristic scale factor [-]
iscale = [1 2 3 4];           % fan characteristic scale factor index [-]
dp = 0:20:1000;               % discretisation pressure [Pa]
dq = 0.5:0.5:5;                   % discretisation volume flow [m^3/s]

%%                  Caluclations for one Fan                            %%
%% Get discrete mesh with information of dp,q,P,n,eta for each fan
 % characteristic (scale factor)
% prealocate loop arrays
    % pressure [Pa]
    dp_init = zeros(length(scale), length(dp), length(dq));
    % fan efficiency [%]
    eta_init = zeros(length(scale), length(dp), length(dq));
     % fan power demand [W]
    P_init = zeros(length(scale), length(dp), length(dq));
    % relative rotational speed [%]
    n_init = zeros(length(scale), length(dp), length(dq));

% loop for every fan characteristic pressure and volume flow and calculate
% the corresponding efficiency, power demand and rotational speed
tic
for nscale = 1:length(scale)
    for ndp = 1:length(dp)
        for ndq = 1:length(dq)
        % matrix sizes: length(scale)*length(dp_init)*length(q_init)

            % save pressure in this matrix size
            dp_init(nscale,ndp,ndq) = dp(ndp);

            % get rotational speed out of analytical description
                %(Enumeration script dp_fan(n,scale,q) solved for n)
            n_init(nscale,ndp,ndq) =(dp(ndp) + (dp(ndp)^2 + 160000*...
                dq(ndq)^2*scale(nscale)^2).^(1/2))/(20*scale(nscale));

            % if rotational speed > 100% or < 0% there is no corresponding
            % physical power. Therefore it is assumed that the related
            % pressure can't be generated!
            if n_init(nscale,ndp,ndq)>100 || n_init(nscale,ndp,ndq)<0
                n_init(nscale,ndp,ndq)= nan;
                eta_init(nscale,ndp,ndq) = nan;
                P_init(nscale,ndp,ndq)= nan;
                dp_init(nscale,ndp,ndq)= nan;
            else
                eta_init(nscale,ndp,ndq) = eta_all(dp_init...
                    (nscale,ndp,ndq),dq(ndq),scale(nscale));
                P_init(nscale,ndp,ndq) = dp(ndp)*dq(ndq)*100/...
                    (eta_init(nscale,ndp,ndq));
            end
        end
    end
end
toc

```

```
%% Get all properties of all combination of three fan
[dp_main,dp_all,P_all,n_all1,n_all2,n_all3,eta_all1,eta_all2,...
    eta_all3,scale_all3,scale_all1,scale_all2,dp_all1,dp_all2,...
    dp_all3,charPerm] = getFanCombination(dp, dp_init, P_init, n_init,...
    eta_init, dq, iscale, numFan);
toc
%% Get solution (property struct with minimum power demand
 % at each point (dp|Q)
[P,n1,n2,n3,eta1,eta2,eta3,scale1,scale2,scale3,dp1,dp2,dp3,dpGes] = ...
    getPropertyStruct(dq, dp_main,dp_all,P_all,n_all1,n_all2,n_all3,...
    eta_all1,eta_all2,eta_all3,scale_all3,scale_all1,scale_all2,dp_all1,...
    dp_all2,dp_all3);
toc
%% Save fan properties (dp, q, P, n, eta)
[prop] = saveTables(dq,P,n1,n2,n3,eta1,eta2,eta3,scale1,...
    scale2,scale3,dp1,dp2,dp3,dpGes, dp,P_all,dp_main,eta_init,P_init);
toc
```

Genutzte Funktionen

getFanCombinations.m

Kombination der einzelnen Ventilator-Eigenschaften für alle Ventilator-Typ-Variationen und Druckaufbau-Variationen für jeden Punkt (Totaler Druckaufbau|Volumenstrom).

```
%% Function: getFanCombination - get all power demands for permutations
 % with repetition of three fan out of four fan types and three pressures
 % out of length(dp)^3 at all points (dp_total|Q)

function [dp_main,dp_all,P_all,n_all1,n_all2,n_all3,eta_all1,eta_all2,...
    eta_all3,scale_all3,scale_all1,scale_all2,dp_all1,dp_all2,dp_all3,...
    charPerm] = getFanCombination(dp, dp_init, P_init, n_init, eta_init,...
    dq, iscale, numFan)
    %% Get properties of combinations of fan characteristics
    % initialize length of each permutation
    K = numFan;                      % number of fan which shall be combined

    % create all possible permutations (with repetition) of fan types
    cellChar = cell(K, 1);            % preallocate a cell array
    [cellChar{:}] = ndgrid(iscale);   % create K grids of values
    % convert grids to column vectors
    charPerm = cellfun(@(iscale){iscale(:)}, cellChar);
    charPerm = [charPerm{:}];         % obtain all permutations of fan types

    % prealocate loop arrays (for a faster calculation)
        % pressure of fan combination [Pa]
        % (the pressure of three fan is added up)
    dp_all = zeros(length(dq),length(charPerm),length(dp_init),...
        length(dp_init), length(dp_init));
        % power demand of fan combination [W]
        % (the pressure of three fan is added up)
    P_all = zeros(length(dq),length(charPerm),length(dp_init),...
        length(dp_init), length(dp_init));
        % rotational speed of fan combination [%]
        % (the rotational speed of the first fan is collected)
```

```
    n_all1 = zeros(length(dq),length(charPerm),length(dp_init),...
        length(dp_init), length(dp_init));
        % (the rotational speed of the second fan is collected)
    n_all2 = zeros(length(dq),length(charPerm),length(dp_init), ...
        length(dp_init), length(dp_init));
        % (the rotational speed of the third fan is collected)
    n_all3 = zeros(length(dq),length(charPerm),length(dp_init), ...
        length(dp_init), length(dp_init));
        % efficiency of fan combination [%]
        %(the efficiency of the first fan is collected)
    eta_all1 = zeros(length(dq),length(charPerm),length(dp_init),...
        length(dp_init), length(dp_init));
        % (the efficiency of the second fan is collected)
    eta_all2 = zeros(length(dq),length(charPerm),length(dp_init),...
        length(dp_init), length(dp_init));
        % (the efficiency of the third fan is collected)
    eta_all3 = zeros(length(dq),length(charPerm),length(dp_init), ...
        length(dp_init), length(dp_init));
        % scale factor of fan combination [-]
        %(the scale factor of the first fan is collected)
    scale_all1 = zeros(length(dq),length(charPerm),length(dp_init),...
        length(dp_init), length(dp_init));
    % (the scale factor of the second fan is collected)
    scale_all2 = zeros(length(dq),length(charPerm),length(dp_init), ...
        length(dp_init), length(dp_init));
    % (the scale factor of the third fan is collected)
    scale_all3 = zeros(length(dq),length(charPerm),length(dp_init),...
        length(dp_init), length(dp_init));
    % pressure of a single fan in the fan combination [Pa]
    % (the pressure of the first fan is collected)
    dp_all1 = zeros(length(dq),length(charPerm),length(dp_init), ...
        length(dp_init), length(dp_init));
    % (the pressure of the second fan is collected)
    dp_all2 = zeros(length(dq),length(charPerm),length(dp_init), ...
        length(dp_init), length(dp_init));
    % (the pressure of the third fan is collected)
    dp_all3 = zeros(length(dq),length(charPerm),length(dp_init), ...
        length(dp_init), length(dp_init));


    %% Loop for volume flow and fan combination and for each
     % pressure step of each of the three fan

       % add up the pressures of three fan to get all permutations of total
       % pressure build up through three fan

    % loop volume flow discretisation steps
    for ndq = 1:length(dq)
        % loop different fan combinations
        for ncPerm = 1:length(charPerm)

            % loop for pressures of every fan in the occuring combination
            % loop pressure fan 1
            for ndp1 = 1:length(dp)
                % loop pressure fan 2
                for ndp2 = 1:length(dp)
```

```
                    % loop pressure fan 3
                    for ndp3 = 1:length(dp)
                        % due to add up the pressures segments of every
                        % fan in the occuring combination, more than one
                        % fan combination at one volume flow generates
                        % the same total pressure build up - all of them
                        % are collected
                        dp_all(ndq,ncPerm,ndp1,ndp2,ndp3) = dp_init...
                            (charPerm(ncPerm,1),ndp1,ndq)+dp_init...
                            (charPerm(ncPerm,2),ndp2,ndq)+dp_init...
                            (charPerm(ncPerm,3),ndp3,ndq);
                        P_all(ndq,ncPerm,ndp1,ndp2,ndp3) = P_init...
                            (charPerm(ncPerm,1),ndp1,ndq)+P_init...
                            (charPerm(ncPerm,2),ndp2,ndq)+P_init...
                            (charPerm(ncPerm,3),ndp3,ndq);
                        n_all1(ndq,ncPerm,ndp1,ndp2,ndp3) = n_init...
                            (charPerm(ncPerm,1),ndp1,ndq);
                        n_all2(ndq,ncPerm,ndp1,ndp2,ndp3) = n_init...
                            (charPerm(ncPerm,2),ndp2,ndq);
                        n_all3(ndq,ncPerm,ndp1,ndp2,ndp3) = n_init...
                            (charPerm(ncPerm,3),ndp3,ndq);
                        eta_all1(ndq,ncPerm,ndp1,ndp2,ndp3) = eta_init...
                            (charPerm(ncPerm,1),ndp1,ndq);
                        eta_all2(ndq,ncPerm,ndp1,ndp2,ndp3) = eta_init...
                            (charPerm(ncPerm,2),ndp2,ndq);
                        eta_all3(ndq,ncPerm,ndp1,ndp2,ndp3) = eta_init...
                            (charPerm(ncPerm,3),ndp3,ndq);
                        scale_all1(ndq,ncPerm,ndp1,ndp2,ndp3) = ...
                            charPerm(ncPerm,1);
                        scale_all2(ndq,ncPerm,ndp1,ndp2,ndp3) = ...
                            charPerm(ncPerm,2);
                        scale_all3(ndq,ncPerm,ndp1,ndp2,ndp3) = ...
                            charPerm(ncPerm,3);
                        % to proof pressures
                        dp_all1(ndq,ncPerm,ndp1,ndp2,ndp3) = ...
                            dp_init(charPerm(ncPerm,1),ndp1,ndq);
                        dp_all2(ndq,ncPerm,ndp1,ndp2,ndp3) = ...
                            dp_init(charPerm(ncPerm,2),ndp2,ndq);
                        dp_all3(ndq,ncPerm,ndp1,ndp2,ndp3) = ...
                            dp_init(charPerm(ncPerm,3),ndp3,ndq);
                    end
                end
            end
        end
    end
    toc

    % initialize max pressure array
    dp_main = 0:dp(2):max(dp_all(:));
end
```

getPropertyStruct.m

Für jeden Punkt (Totaler Druckaufbau|Volumenstrom) finden des minimalen Leistungsbedarfs und dessen Parameter-Permutation.

```
%% Function: getPropertyStruct - get only minimum power demand
 %(not all power demands) at each point (dp|Q) and corresponding
 % parameter-permutation

function [P,n1,n2,n3,eta1,eta2,eta3,scale1,scale2,scale3,dp1,dp2,...
    dp3,dpGes] = getPropertyStruct(dq, dp_main,dp_all,P_all,n_all1,...
    n_all2,n_all3,eta_all1,eta_all2,eta_all3,scale_all3,scale_all1,...
    scale_all2,dp_all1,dp_all2,dp_all3)
    %% Get property struct
    % preallocate structs for loop
    P = struct(); n1 = struct(); n2 = struct(); n3 = struct(); ...
        eta1 = struct(); eta2 = struct(); eta3 = struct(); ...
        scale1 = struct(); scale2 = struct(); scale3 = struct(); ...
        dp1 = struct();dp2 = struct();dp3 = struct(); dpGes= struct();

    %% sort dp_all to get all values (related index) at the
     % same point (pressure|volume flow)
    for nq = 1:length(dq)
        for ndp=1:length(dp_main)
          % get all index of same pressure value
          % (e.g. all index of dp = 50 Pa); dp_res is already sorted
          [idx] = find(dp_all(nq,:,:,:,:) ==dp_main(ndp));

          %% get all power demands at current pressure and volume
           % flow(e.g. q_init = 1 m^3/s | dp = 50Pa)

          % if current pressure was found switch is true
          switch true
              % the current pressure can not be found
              case isempty(idx)
                  dpGes(nq,ndp).end = nan;
                  P(nq,ndp).end = nan;
                  n1(nq,ndp).end = nan;
                  n2(nq,ndp).end = nan;
                  n3(nq,ndp).end = nan;
                  eta1(nq,ndp).end = nan;
                  eta2(nq,ndp).end = nan;
                  eta3(nq,ndp).end = nan;
                  scale1(nq,ndp).end = nan;
                  scale2(nq,ndp).end = nan;
                  scale3(nq,ndp).end = nan;
                  dp1(nq,ndp).end = nan;
                  dp2(nq,ndp).end = nan;
                  dp3(nq,ndp).end = nan;

              % the current pressure can be found
              case ~isempty(idx)
                  %% get all existing properties for current point (dp|Q)
                  for nidx = 1:length(idx)
                      dp_current(nidx) = dp_all(nq,idx(nidx));
                      P_current(nidx) = P_all(nq,idx(nidx));
                      n1_current(nidx) = n_all1(nq,idx(nidx));
                      n2_current(nidx) = n_all2(nq,idx(nidx));
                      n3_current(nidx) = n_all3(nq,idx(nidx));
                      eta1_current(nidx) = eta_all1(nq,idx(nidx));
```

```
            eta2_current(nidx) = eta_all2(nq,idx(nidx));
            eta3_current(nidx) = eta_all3(nq,idx(nidx));
            scale1_current(nidx) = scale_all1(nq,idx(nidx));
            scale2_current(nidx) = scale_all2(nq,idx(nidx));
            scale3_current(nidx) = scale_all3(nq,idx(nidx));
            dp1_current(nidx) = dp_all1(nq,idx(nidx));
            dp2_current(nidx) = dp_all2(nq,idx(nidx));
            dp3_current(nidx) = dp_all3(nq,idx(nidx));
        end

        %% get minimum power and corresponding properties at
         % current point (dp|Q)

        % get minimum power demand > zero
        P(nq,ndp).end = min(P_current(P_current > 0));

        % switch for minimum power demand is NaN, zero or other
        switch true
            % minimum power demand is zero or NaN
            case isempty(P(nq,ndp).end)
                % minimum power demand is NaN
                if isnan(min(P_current(:)))
                    P(nq,ndp).end = min(P_current(:));
                    % get the corresponding index
                    % (for the properties) to the first NaN
                    % power demand configuration
                    [fidx] = find(isnan(P_current));
                    dpGes(nq,ndp).end = dp_current(fidx(1));
                    n1(nq,ndp).end = n1_current(fidx(1));
                    n2(nq,ndp).end = n2_current(fidx(1));
                    n3(nq,ndp).end = n3_current(fidx(1));
                    eta1(nq,ndp).end = eta1_current(fidx(1));
                    eta2(nq,ndp).end = eta2_current(fidx(1));
                    eta3(nq,ndp).end = eta3_current(fidx(1));
                    scale1(nq,ndp).end = scale1_current(fidx(1));
                    scale2(nq,ndp).end = scale2_current(fidx(1));
                    scale3(nq,ndp).end = scale3_current(fidx(1));
                    dp1(nq,ndp).end = dp1_current(fidx(1));
                    dp2(nq,ndp).end = dp2_current(fidx(1));
                    dp3(nq,ndp).end = dp3_current(fidx(1));
                else
                    % minimum power demand is zero
                    P(nq,ndp).end = min(P_current(:));
                    % get the corresponding index
                    % (for the properties) to the first zero
                    % power demand configuration
                   [fidx] = find(P_current == min(P_current(:)));
                    dpGes(nq,ndp).end = dp_current(fidx(1));
                    n1(nq,ndp).end = n1_current(fidx(1));
                    n2(nq,ndp).end = n2_current(fidx(1));
                    n3(nq,ndp).end = n3_current(fidx(1));
                    eta1(nq,ndp).end = eta1_current(fidx(1));
                    eta2(nq,ndp).end = eta2_current(fidx(1));
                    eta3(nq,ndp).end = eta3_current(fidx(1));
                    scale1(nq,ndp).end = scale1_current(fidx(1));
                    scale2(nq,ndp).end = scale2_current(fidx(1));
```

```
                                scale3(nq,ndp).end = scale3_current(fidx(1));
                                dp1(nq,ndp).end = dp1_current(fidx(1));
                                dp2(nq,ndp).end = dp2_current(fidx(1));
                                dp3(nq,ndp).end = dp3_current(fidx(1));
                            end

                        % minimum power demand is not zero
                        case ~isempty(P(nq,ndp).end)
                            % get the corresponding index
                            % (for the properties) to the first non zero
                            % power demand configuration
                            [fidx] = find(P_current ==...
                                min(P_current(P_current > 0)));
                            dpGes(nq,ndp).end = dp_current(fidx(1));
                            n1(nq,ndp).end = n1_current(fidx(1));
                            n2(nq,ndp).end = n2_current(fidx(1));
                            n3(nq,ndp).end = n3_current(fidx(1));
                            eta1(nq,ndp).end = eta1_current(fidx(1));
                            eta2(nq,ndp).end = eta2_current(fidx(1));
                            eta3(nq,ndp).end = eta3_current(fidx(1));
                            scale1(nq,ndp).end = scale1_current(fidx(1));
                            scale2(nq,ndp).end = scale2_current(fidx(1));
                            scale3(nq,ndp).end = scale3_current(fidx(1));
                            dp1(nq,ndp).end = dp1_current(fidx(1));
                            dp2(nq,ndp).end = dp2_current(fidx(1));
                            dp3(nq,ndp).end = dp3_current(fidx(1));
                    end
            end

            % clear current variables to get in next iteration only new
            % variables (because of different length of variables in
            % each iteration)
            clear idx dp_current P_current n1_current n2_current ...
                n3_current eta1_current eta2_current eta3_current...
                scale1_current scale2_current scale3_current ...
                dp1_current dp2_current dp3_current
        end
    end
end
```

saveTables.m

Speichern des Leistungsbedarfes, Wirkungsgrades und Parameter-Permutation für jeden Punkt (Totaler Druckaufbau|Volumenstrom).

```
function[prop] = saveTables(dq,P,n1,n2,n3,eta1,eta2,eta3,scale1,...
    scale2,scale3,dp1,dp2,dp3,dpGes, dp,P_all,dp_main,eta_init,P_init)
    %% save fan properties (dp, q, P, n, eta)
    % get variable names as table
    strVarName = string({'P';'n1';'n2';'n3';'eta1';'eta2';'eta3';...
        'scale1';'scale2';'scale3';'dp1';'dp2';'dp3';'dp'});
    varName = table(strVarName);

    % get directory to save results (csv-files)
```

```
    my_directory = ...
    'S:\ssl\_Speicher_Festplatte\LaTex_Lott\MatlabSkripte\Faktorisierung\propertyTables';

    % get all possible values of all properties for every point(dq|dp)
    for nq = 1:length(dq)
        for ndp=1:length(dpGes)
            % get all properties as one  table
            prop{nq,ndp} = array2table([P(nq,ndp).end;...
                n1(nq,ndp).end;n2(nq,ndp).end;n3(nq,ndp).end;...
            eta1(nq,ndp).end;eta2(nq,ndp).end;eta3(nq,ndp).end;...
            scale1(nq,ndp).end;scale2(nq,ndp).end;...
            scale3(nq,ndp).end;dp1(nq,ndp).end;dp2(nq,ndp).end;...
            dp3(nq,ndp).end;dpGes(nq,ndp).end]);
            % get variable file name
            fname = sprintf('prop_%.1f_%.1f_oneValue.csv',...
                dq(nq),dpGes(nq,ndp).end);

            % check if variable file name already exists
            if exist([my_directory filesep fname]) == 0
                fullfname = [my_directory filesep fname];
                % save tables for each point (dq|dp)
                writetable([varName,prop{nq,ndp}],fullfname)
            else
                fname = sprintf('prop_%.1f_%.1f_oneValue%.0f.csv', ...
                    dq(nq),dpGes(nq,ndp).end,dp_main(ndp));
                fullfname = [my_directory filesep fname];
                writetable([varName,prop{nq,ndp}],fullfname)
            end
        end
    end
    %%
    % save workspace variables
    save('propertyTables\factorization.mat','P','P_all','dp1',...
        'dp2','dp3','eta1','eta2','eta3','scale1','scale2','scale3',...
        'P_init','n1','n2','n3','dp','dp_main','eta_init','dq')
end
```

10.3.3 Skripte Genetischer Algorithmus

Hauptprogramm

LoopTenTimes.m

Zehnfaches Aufrufen des Hauptprogramms, um eine Aussage über die Reproduzierbarkeit der Lösung treffen zu können.

```
%% loop GA 10times
global sim

elapsedTimePersixVolumeflow = zeros(1,10);

for sim = 1:10
   [elapsedTime] = mainGA();

   elapsedTimePersixVolumeflow(sim) = elapsedTime;
```

```
end
```

mainGA.m

In diesem MATLAB-file erfolgt die Initialisierung des Optimierungsproblems mit Parameter und deren Zahlenraum, Zielfunktion und Konvergenzkriterium. Außerdem wird die MATLAB Funktion ga für verschiedene Volumenströme aufgerufen.

```
function[elapsedTime] = mainGA()
%% main function - minimization
clc
clear

tic
% get global variables
global q
global numFan
global scale
global numVar
global N
global n
global dp_balanceMin

% initialize variables
scale = [0.25 0.5 0.75 1.0];
numFan = 3;
numVar = 3;

% get minimum pressure from enumartion
q_init = 0.5:0.5:3;
dp_balance = nan(1,length(q_init));
dp_enum = [37.5,150,283.75,506.7,798.6,1140];

for idx = 1:length(q_init)
    dp_balance(idx) = dp_enum(idx);
end

% define problem
fitnessfcn = @myFitness;    % fitnessfcn: <Fitness function>
nVars = numVar*numFan;      % nvars: <Number of design variables>
Aineq = [];                 % Aineq: <A matrix for inequality constraints>
bineq = [];                 % bineq: <b vector for inequality constraints>
Aeq = [];                   % Aeq: <Aeq matrix for equality constraints>
beq = [];                   % beq: <beq vector for equality constraints>
lb = [1;1;1;1;1;1;0;0;0];   % lb: <Lower bound on X>
ub = [N;N;N;4;4;4;1;1;1];   % ub: <Upper bound on X>
nonlcon = [];               % nonlcon: <Nonlinear constraint function>
IntCon = 1:nVars;           % intcon: <Index vector for integer variables>

    % assumed mesh matrix = 1 1 1 (ausgesetzt, wenn hintere 3 lb = 0)
    %                        1 1 1
    % --> for x1,x2,x3 = 1 all fan are included
    % [n1;n2;n3;iscale1;iscale2;iscale3;x1;x2;x3]

%% generate visulatization and stopping criterion
```

```
rng default % for reproducability
opts = ...
    optimoptions(@ga,'PlotFcn',{@gaplotbestf,@gaplotdistance},'Display','iter', ...
    ...
    'MaxStallGenerations',100, 'MaxGenerations',100);

%% loop for target flows
for q = 0.5:0.5:3
    % get minimum pressure from enumartion
    [qidx] = find(q_init == q);          % index
    dp_balanceMin = dp_balance(qidx);      % value
    %% calculate minimum power demand by genetic algorithm with
     % (constraint) positive integer variables
    [var_min, P_min, exitFlag, Output] = ga(fitnessfcn, nVars, Aineq,...
        bineq, Aeq, beq, lb, ub, nonlcon, IntCon, opts);

    %% Display output information
    % get information out of ga % not necessary
        fprintf('The variable configuration (combnatorics) for minimal power ...
            demand was: n(%d,%d,%d), scale(%3g,%3g,%3g), x(%d,%d,%d)\n', ...
            n(var_min(1)), n(var_min(2)), n(var_min(3)), scale(var_min(4)), ...
            scale(var_min(5)), scale(var_min(6)), var_min(7), var_min(8), ...
            var_min(9));
        fprintf('The number of generations was : %d\n', ...
            Output.generations);
        fprintf('The number of function evaluations was : %d\n',...
            Output.funccount);
        fprintf('The best function value found was : %g\n', P_min);

    % generate information important for each problem to save
    [dp, dp_ges, eta] = getPressureAndEfficiency_andSave(var_min, P_min, ...
        Output);
        fprintf('The pressure and effiency to corresponding variable ...
            configuration was: dp(%.1f,%.1f,%.1f), dp_total = %.1f, ...
            eta(%.0f,%.0f,%.0f) at q=%.1f\n',...
            dp(1), dp(2), dp(3), dp_ges, eta(1), eta(2),eta(3), q);
end
toc
elapsedTime = toc;
end
```

Genutzte Funktionen

myFitness.m

Berechnung von Druckaufbau, Wirkungsgrad und Leistungsbedarf für eine Parameter-Permutation unter Berücksichtigung der Bedingungen an Leistungsbedarf und Druck (positiver Leistungsbedarf, minimaler Gesamtdruckaufbau wie in Enumeration).

```
%% Fitness function
% it is important, that this function only gets 'var', no more variables!!!
% This is the reason for the global variables

function P_el_ges = myFitness(var)
    %% General information to this function
```

```
% var =[n1;n2;n3;iscale1;iscale2;iscale3;x1;x2;x3];
% n = [n1 n2 n3] % depending on numFan
% scale = [scale1 scale2 scale3] % depending on numFan
% x = [x1 x2 x3] % depending on numFan x1,2,3= [0 1] wheter fan is
% existing or not at this position

% initialize numFan and tagret flow and fan characteristics
global numFan
global q                          % target flow in all fan
global scale
global dp_balanceMin
global N
global n

% get function handle for pressure
    % analytical description fan [Pa]
    dp_handle = @(q,n_init,iscale) ...
        scale(iscale)*10*n_init-(scale(iscale)*10*n_init/...
        ((n_init/20)^2)*q^2);
    % power demand fan [W]
    P_handle = @(q,dp,eta) dp*q*100/eta;

% initialize rotational speed and increment
N = 11;                            % increment rotational speed
n = linspace(0,100,N);             % rotational speed [%]

% preallocate power array for loop
P = zeros(numFan,1);
dp = zeros(numFan,1);
eta = zeros(numFan,1);

%% get power demand for every single fan in dependency of variables
for nFan = 1:numFan
    dp(nFan) = dp_handle(q,n(var(nFan)),var(nFan+numFan))*...
        var(length(var)-numFan+nFan);
    eta(nFan) = eta_all(dp_handle(q,n(var(nFan)),var(nFan+numFan)),...
        q, scale(var(nFan+numFan)));
    P(nFan) = P_handle(q,dp(nFan),eta(nFan));

    % define range of pressure and lower bound of power demand
    if dp(nFan)>(1000*var(nFan+numFan))|| dp(nFan)<0 || P(nFan)<=0
        dp(nFan) = nan;
        eta(nFan) = nan;
        P(nFan) = nan;
    end
end

%% get total pressure and power demand
dp_ges = sum(dp);

% get only not nan values for power demand
getPvalue = find(~isnan(P(:)) ==1);
P_fan = P(getPvalue);
x_fan = var(6+getPvalue);

if sum(~isnan(P(:))) >= 1 && sum(x_fan)>=1
```

```
    P_el_ges = sum(P_fan.*x_fan');
    else
    P_el_ges = nan;
    end

    %% check if negative pressures are used
    if ~isnan(dp_ges)
        % check pressure balance like in enumeration
        if ~isnan(dp_balanceMin)
            if dp_ges<dp_balanceMin
                P_el_ges = nan;
            end
        end
    % if negative pressures are used set power demand to nan
    else
        P_el_ges = nan;
    end
end
```

getPressureAndEfficiency_andSave.m

Speichern der Lösung (minimaler Leistungsbedarf und zugehörige Parameter-Permutation) und Anzahl der Generationen, die berechnet wurden.

```
%% Get minimal pressure and efficiency and save
function [dp, dp_ges, eta] = getPressureAndEfficiency_andSave...
    (var_min, P_min, Output)
%% General information to this function
% this function is equal to the fitness function. The output of
% this function are
% pressure and efficiency of the minimal variable configuartion
% var_min =[n1;n2;n3;iscale1;iscale2;iscale3;x1;x2;x3].


% initialize tagret flow
global q
global numFan
global scale
global n
global sim

% calculate pressure again with solution (get function handle for pressure)
dp_handle = @(q,n_init,iscale) ...
    scale(iscale)*10*n_init-(scale(iscale)*10*n_init/...
    ((n_init/20)^2)*q^2);

% preallocate power array for loop
dp = zeros(numFan,1);
eta = zeros(numFan,1);

    % get power demand for every single fan
    for nFan = 1:numFan
    dp(nFan) = dp_handle(q,n(var_min(nFan)),var_min(nFan+numFan));
    eta(nFan) = eta_all(dp_handle(q,n(var_min(nFan)),...
```

```
30          var_min(nFan+numFan)), q, scale(var_min(nFan+numFan)));
31      end
32
33  % get total pressure
34  dp_ges = dp(1)*var_min(7)+dp(2)*var_min(8)+dp(3)*var_min(9);
35
36  %% Save
37  % get variable names as table
38  strVarName = string({'P';'n1';'n2';'n3';'eta1';'eta2';'eta3';'scale1';...
39      'scale2';'scale3';'dp1';'dp2';'dp3';'dp_ges';'x1';'x2';'x3';...
40      'number of generations';'number of function evaluations'});
41  varName = table(strVarName);
42
43  % define property table
44  prop = array2table([P_min;n(var_min(1));n(var_min(2));n(var_min(3));...
45      eta(1);eta(2);eta(3);var_min(4);...
46  var_min(5);var_min(6);dp(1);...
47  dp(2);dp(3);dp_ges;var_min(7);var_min(8);var_min(9);...
48  Output.generations;Output.funccount]);
49
50  % get file name
51  fname = sprintf('prop_%.2f_%.1f_ga.csv', q,dp_ges);
52
53  % get directory to save results (csv-files)
54  my_directory = ...
55   'D:\Studium\Masterarbeit_IGTE\LaTex_Lott\MatlabSkripte\GA\propertyTables';
56      if exist([my_directory filesep fname]) == 0
57          fullfname = [my_directory filesep fname];
58          writetable([varName,prop],fullfname)
59      else
60          fname = sprintf('prop_%.1f_%.1f_%.0f.csv', q,dp_ges,sim);
61          fullfname = [my_directory filesep fname];
62          writetable([varName,prop],fullfname)
63      end
64
65  end
```

10.3.4 Skripte für den Vergleich der Optimierungsmethoden

Skripte an Auslegungsbeispiel

Enumeration_loop.m

In diesem MATLAB-file erfolgt die Initialisierung der Luftverteilung. Außerdem werden alle Funktionen aufgerufen und miteinander kombiniert, entsprechend der schematischen Darstellung Bild 5-1. Der Unterschied zum Hauptprogramm in 10.3.1 besteht in der Nutzung von Interpolationsfunktionen mit Fallunterscheidungen für die analytische Beschreibung der Ventilatoren und für die Interpolation des Wirkungsgrades.

```
1  % function[meshMatrix, IDMatrix, alphas,Alphas,...
2  %      nPerm,charPerm,nVar,charVar,targetflow,Opt_PallFan,dp_fanOpt,...
3  %      eta_fanOpt,fullSym,eta,fullSym_zeroDamp,eqSyst,eqSystOneDamp,...
4  %              eqSystZeroDamp, elapsedTime] = Enumeration_loop(x)
5
```

```
%% Enumeration - Optimization method 1 for reference air distribution
% Minimum of electrical power demand for a fluid distribution with a
% given number of elements, meshes and their volume flow.
% More description in ReadMe.txt
%
% Used functions:
% eta_all.m
% solveAlpha_numMeshEqualNumDamp.m
% permutation.m
% meshMatrix_test.m (uses nmultichoosek.m)
% saveData.m
%
% Parameters:
% number of fans (numFan)
% number of duct elements (numDuct)
% number of dampers (numDamp)
% number of meshes (numMesh)
% target flows for every mesh (targetflow(i))
% characteristics of Elements
% scale factor for fan characteristics (sacles pressure, changes curve
% shape) (scaleFan)
%
% Comments:
% i is in all loops equal to numMesh
% j is in all loops equal to numEl (or numFan)
%
% History:
% 2019 - 05 - 15        S.Lott
%                       Script created
% 2019 - 05 - 24        functions solveAlpha and elPower added
% 2019 - 06 - 26        function elPower not used anymore, funktions
%                       permutation.m and meshMatrix_test.m
%                       (uses nmultichoosek.m)added
%                       Skript is tested for a special number of elements
%                       and meshes (2 meshes, 2 dampers, 2-3 fan, 3 ducts,
%                       target flows until )
%%
% clc
% clear
clear functions
clearvars('-except', 'x');

tic
%% Get global variables
global numFan
global tf1
global tf2
global f1
global f2
global f3
global getEta_vzr200
global getEta_vzr225
global getEta_vzr250

% get data and fit curves from data sheets for fan characteristics
    % get rotational speed fit function
```

```
    [f1,f2,f3] = getRotSpeedFit();

    % get eta fit function
    [getEta_vzr200,getEta_vzr225,getEta_vzr250] = getEtaFit();

%% Current Param - change here target flow, number of elements/meshes,
%  initialize mesh matrix

% get number of fan (nF = as function input), duct and damper resistance
numDuct = 3;                    % temporary const, as a constant duct
                                % structure is assumed
numDamp = 2;                    % total number of damper, in this
                                % simulation constant

% initialize rotational speed and increment
N = 7;                          % increment rotational speed (test 61)
n = linspace(40,100,N);         % rotational speed [%]

% initialize number of meshes and total elements
numMesh = 2;
numEl = numFan + numDuct + numDamp;

% initialize and set target flow (tf1 and tf2 as function input) in ...
    dependency of mesh
    % initialize targetflow with zeros and define the output flows
targetflow = zeros(1,numMesh)';

    % targetflow(1) = volume flow in mesh one and two [m^3/h]
targetflow(1) = tf1(x)*3600;
targetflow(2) = tf2(x)*3600;

% initialize fan characteristic scale factors
scaleFan = [1,2,3];

% Initialize ID matrix (used in solveAlpha)
    % ID Matrix, where fan = 1, duct = 2 and damper = 3 are located in
    % mesh matrix
cID_fan = zeros(numMesh, numFan);
cID_fan(:,:) = 1;
cID_duct = zeros(numMesh, numDuct);
cID_duct(:,:) = 2;
cID_damp = zeros(numMesh, numDamp);
cID_damp(:,:) = 3;

cID = [cID_fan,cID_duct,cID_damp];

% initialize function handle electrical power fan
P  = @(eta,dp,q) dp*q*100/(eta*3600);

% get mesh matrix
[M, fanComb] = meshMatrix_test(numMesh, numEl, numDamp, numFan);

% fill volume flow struct
V = targetflow.*M;              % volume flow matrix (there is only a
                                % volume flow if the element is existing;
                                % for calculation of volume flow through
```

```
                                % element)
q = struct([]);                 % initialize volume flow struct (for better
                                % elementwise access)

% volume flow for each element
    % q(1,1).flow = total flow through element 1
    % and mesh matrix combination 1
for numComb = 1:length(M(1,1,:))
    for j = 1:numEl
        q(j, numComb).flow = sum(V(:,j,numComb));
    end
end
toc
%% Initialize pressure struct of characteristics
dp = struct([]);                % initialize pressure struct

% fill pressure struct with characteristic function handles (analytical
% description) for each element in each mesh matrix combination
for numComb = 1:length(M(1,1,:))      % loop for mesh matrix permutation
    for i = 1:numMesh                 % loop for numMesh
        for j = 1:numEl               % loop for numElement
            if M(i, j, numComb) == 1  % element is existing
               switch  cID(i, j)      % element is fan, duct or damper
                  case 1
                  % Fan
                    % characteristic fan, analytical description
                    % --> adjust eta_all to fan characteristic!
                    dp(i,j,numComb).syst = ...
                        @(j,n,scaleFan,numComb,q) ...
                        getFanCharacteristics(j,n,scaleFan,numComb,q);

                   case 2
                  % Duct
                    % characteristic duct, analytical description
                    % factor adjustment: 100000 = very steep, 100 = enough
                    % pressure drop for varying fan characteristics
                    dp(i,j,numComb).syst = @(j) ...
                        -(1000/3600^2)*q(j,numComb).flow^2;

                   case 3
                  % Damper
                    % characteristic damper, analytical description
                    % factor adjustment: 300000 = steep can only find alpha
                    % for low target flows; 300 = less steep, find alpha
                    % (pressure drops) for almost all target flows and
                    % pressure build ups
                    dp(i,j,numComb).syst = @(alpha, j) -300/3600^2/...
                        (alpha^2/90)*q(j,numComb).flow.^2;
               end
            end
        end
    end
end
toc
%% Calculation of electical power demand for every permutation of
%  rotational speed and characteristic, for every fan combination.
```

```
173 %  Find minimum with an physically possible damper angle alpha.
174
175     %% Get electical power demand matrix
176     % get power demand P_allFan for all possible parameter-permutations
177     [P_allFan,P_fan,dpAllFan,nPerm,charPerm,eta] = permutation(P, ...
178     numFan,numMesh, n, M, q,scaleFan, dp);
179 toc
180     %% Solve equation system to get alphas (function: solveAlpha), delete
181     % power demand and parameter-permutation, if no physical alpha can
182     % be found
183         % initialize start arrays
184     alphas  = struct([]);
185     P_allFan_orig = P_allFan;
186
187     % loop until power demand minimum with physically possible damper
188     % angles is find
189     while isempty(alphas) || any(structfun(@isempty, alphas))
190
191         % get power minimum of all permutations and combinations and
192         % related pressure values
193         Opt_PallFan = min(P_allFan_orig(:));
194
195         % get position of power minimum
196             % nVar = rotaional speed permutation, charVar =
197             % characteristic permuation, fanCom = fan combination, with
198             % these values it is possible to get a postion in the power
199             % demand matrix
200         [nVar,charVar,fanCom] = ind2sub(size(P_allFan_orig),find(...
201         P_allFan_orig==Opt_PallFan));
202             if isempty(nVar) || isempty(charVar) || isempty(fanCom)
203             % for no solution found (all power demands are equal or nan)
204             dp_fanOpt = 0;
205             eta_fanOpt = 0;
206             else
207             % for more than one minimum, take the first minimum
208             dp_fanOpt = dpAllFan(:,:,nVar(1),charVar(1),fanCom(1));
209             eta_fanOpt = eta(:,nVar(1),charVar(1),fanCom(1));
210             end
211
212         % get damper angles from function - solveAlpha
213         try
214         [alphas,fullSym,fullSym_zeroDamp,eqSyst,eqSystOneDamp,...
215             eqSystZeroDamp] = solveAlpha_test(numMesh,...
216         numEl, numFan, numDuct, numDamp, M, dp, cID, dp_fanOpt, fanCom(1));
217 %         [alphas, eqn, fullSym] = solveAlpha_numMeshEqualNumDamp(numMesh,...
218 %         numEl, numFan, numDuct, numDamp, M, dp, cID, dp_fanOpt, fanCom(1));
219
220         % display error message if no alpha was found for several reasons
221         catch ME
222             if (strcmp(ME.identifier,'MATLAB:badsubscript'))
223                 msg = ME.message;
224                 disp(msg);
225                 % bad subscript exception handling goes here
226                 disp(['Target flows are in total to high.'...
227                 'Chose lower target flow or other fan characteristic.']);
228             end
```

```
            continue
        end

        % set power minimum to NaN to get next higher minimum
        % if no alpha was found
        if isempty(alphas)== 1 || any(structfun(@isempty, alphas)) == 1
        P_allFan_orig(nVar,charVar,fanCom) = nan;
        end

    end
    toc
      Alphas = structfun(@double, alphas, 'uniformoutput', 0);

    %% Save data as property structs (function)
    [meshMatrix, IDMatrix] = ...
        saveData(numEl,numFan,numDamp,M,cID,alphas,Alphas,fanCom(1),...
    nPerm,charPerm,nVar(1),charVar(1),targetflow,Opt_PallFan,dp_fanOpt,...
    eta_fanOpt, dpAllFan, P_allFan,fullSym);

    toc

    elapsedTime = toc;
% end
```

Genutzte Funktionen

nmultichoosek.m

saveData.m

meshMatrix_test.m Bleiben entsprechend Kapitel 10.3.1. **eta_allDatasheet.m**

Berechnen des Wirkungsgrades entsprechend Ventilator-Typ und Interpolationsfunktion.

```
%% Function: eta_all - get efficiency eta via fit function out of getEtaFit
%  for different fan types (scaleFan), pressures and volume flows
function [eta_out] = eta_allDatasheet(dp, q, scaleFan)
% get eta fit functions (interpolation spline)
global getEta_vzr200; global getEta_vzr225; global getEta_vzr250

% get value of eta for specific dp, q and fan type
    switch scaleFan
        case 1
            eta_out = getEta_vzr200(q,dp);

            if dp<=0 || q<0
            eta_out = nan;
            end
        case 2
            eta_out = getEta_vzr225(q,dp);

            if dp<=0 || q<0
            eta_out = nan;
            end
        case 3
            eta_out = getEta_vzr250(q,dp);
```

```
23
24              if dp<=0 || q<0
25              eta_out = nan;
26              end
27      end
28  end
```

getEtaFit.m

Erzeugen der Wirkungsgrad-Interpolationsfunktionen für drei reale Kennfelder anhand abgelesener Datenpunkte (x|y|z = Druck|Volumenstrom|Wirkungsgrad).

```
1  %% getEtaFit Function - get interpolation function from Datasheet data
2  function[getEta_vzr200,getEta_vzr225,getEta_vzr250] = getEtaFit()
3  %% get data from data sheet
4  [dp_vzr200_eta,q_vzr200_eta,eta_vzr200_eta,dp_vzr225_eta,...
5      q_vzr225_eta,eta_vzr225_eta,dp_vzr250_eta,q_vzr250_eta,...
6      eta_vzr250_eta] = getDatasheet_eta();
7
8  %% cubic interpolation to fit eta data
9  % Set up fittype and options.
10 ft = 'thinplateinterp';
11
12 %% get eta in dependency of q and dp
13 % vzr200
14 [getEta_vzr200] = fit([q_vzr200_eta,dp_vzr200_eta],eta_vzr200_eta,ft,...
15                      'Normalize', 'on');
16 % vzr225
17 [getEta_vzr225] = fit([q_vzr225_eta,dp_vzr225_eta],eta_vzr225_eta,ft,...
18                      'Normalize', 'on');
19 % vzr250
20 [getEta_vzr250] = fit([q_vzr250_eta,dp_vzr250_eta],eta_vzr250_eta,ft,...
21                      'Normalize', 'on');
22 end
```

getRotSpeedFit.m

Erzeugen der analytischen Beschreibung des Druckes für drei reale Kennlinien bei einer Drehzahl von 100% anhand abgelesener Datenpunkte (x|y = Druck|Volumenstrom).

```
1  %% get data from data sheets
2  function[f1,f2,f3] = getRotSpeedFit()
3
4      % VZR 71-0200 data n = 100%
5      dp_vzr200 = [2350;1950;1625;1200;470];
6      q_vzr200 = [2300;3000;3500;4000;5000];
7
8      % VZR 71-0225 data n = 100%
9      dp_vzr225 = [2300;1900;1450;850;460];
10     q_vzr225 = [3000;4000;5000;6000;6800];
11
12     % VZR 71-0250 data n = 100%
```

```
    dp_vzr250 = [2400;2000;1600;1075;610;450];  % y
    q_vzr250 = [3900;5000;6000;7000;8000;8400];  % x

    % get fit functions
    % VZR 71-0200 data n = 100%
    f1 = fit(q_vzr200,dp_vzr200,'poly2');
    % VZR 71-0225 data n = 100%
    f2 = fit(q_vzr225,dp_vzr225,'poly2');
    % VZR 71-0250 data n = 100%
    f3 = fit(q_vzr250,dp_vzr250,'poly2');
end
```

getFanCharacteristics.m

Analytischen Beschreibungen des Druckes für drei reale Kennfelder in Abhängigkeit der Drehzahl (Teillast möglich).

```
%% Function getFanCharacteristics
function[dp_fanChar] = getFanCharacteristics(j,n,scaleFan,numComb,q)

    % get fit functons for rotational speed = 100% from data sheet
    global f1; global f2; global f3

    % get fit function depending on fan type and rotational speed
    switch scaleFan
        case 1
        dp_fanChar = (n./100).^...
        (-0.0615).*f1.p1.*q(j,numComb).flow.^2 + ...
        (n./100).^(1.1206).*f1.p2.*q(j,numComb).flow...
        + (n./100).^(2.0430).*f1.p3;
        case 2
        dp_fanChar = (n./100).^...
        (0.4152).*f2.p1.*q(j,numComb).flow.^2 + ...
        (n./100).^(0.6586).*f2.p2.*q(j,numComb).flow...
        +(n./100).^(2.0175).*f2.p3;
        case 3
        dp_fanChar = (n./100).^...
        (-0.6581).*f3.p1.*q(j,numComb).flow.^2 ...
        + (n./100).^(2.7386).*f3.p2.*...
        q(j,numComb).flow + (n./100).^(2.2648).*f3.p3;
    end
end
```

permutation.m

Entspricht der Funktion Permuation.m des Kapitels 10.3.1. Dieses Programm nutzt jedoch die beschriebenen Interpolationsfunktionen.

```
%% Function:permutation - get all parameter-permutations,
%  corresponding pressures, efficienies and power demands

function [P_allFan,P_fan,dpAllFan,nPerm,charPerm,eta] =...
```

```
    permutation(P,numFan, numMesh, n, M, q,scaleFan, dp)

% initialize length of each permutation (number of fan which shell be
% combined)
K = numFan;

% create all possible permutations (with repetition)
% of rotational speed n for a fan combination
cellSpeed = cell(K, 1);                          % preallocate a cell array
[cellSpeed{:}] = ndgrid(n);                      % create K grids of values
nPerm = cellfun(@(n){n(:)}, cellSpeed);   % convert grids to column vectors
nPerm = [nPerm{:}];            % obtain all permutations of rotaional speed

    % numFan = 9 is responding a out of memory problem at this point,
    % because the number of rotational speed permutations are rising to
    % 11^9 = 2357947691

% create all possible permutations (with repetition)
% of fan types for a fan combinatorics
cellChar = cell(K, 1);                           % preallocate a cell array
[cellChar{:}] = ndgrid(scaleFan);                % create K grids of values
charPerm = cellfun(@(scaleFan){scaleFan(:)}, ...
cellChar);                                  % convert grids to column vectors
charPerm = [charPerm{:}];            % obtain all permutations of fan types

    % numFan = 9 is responding the number of fan type permutations of
    % 4^9 = 262144

% initialize arrays
dpAllFan = zeros(numMesh,numFan,length(nPerm),length(charPerm),...
           length(M(1,1,:)));
eta = zeros(numFan,length(nPerm),length(charPerm),length(M(1,1,:)));
P_fan = zeros(numFan,length(nPerm),length(charPerm),length(M(1,1,:)));
P_allFan = zeros(length(nPerm),length(charPerm),length(M(1,1,:)));

% loop for all elements in all meshes, in all rotational speed ...
% permutations, in all characteristic permutations and all mesh matrix
% permutations
for numComb = 1:length(M(1,1,:))
    for chn = 1:length(charPerm)
        for vn = 1:length(nPerm)
            for i = 1:numMesh
                for j = 1:numFan
                    if M(i,j,numComb) ==1
                    % get pressure build up
                    dpAllFan(i,j,vn,chn,numComb) = ...
                        dp(i,j,numComb).syst(j,nPerm(vn,j),...
                        charPerm(chn,j),numComb,q);
                    dpAllFan(dpAllFan(:)<0)=0;

                    % get efficiency
                        % calculation of efficiency for one fan in relation
                        % to number of meshes in which fan is used (without
                        % relation, pressure for every mesh would sum up)
                    eta(j,vn,chn,numComb) = eta_allDatasheet...
                        (sum(dpAllFan(:,j,...
```

```
                        vn,chn,numComb))/sum(M(:,j,numComb)),...
                        q(j,numComb).flow,charPerm(chn,j));

                    % get electrical power demand
                    P_fan(j,vn,chn,numComb) = P(eta(j,vn,chn,numComb),...
                        sum(dpAllFan(:,j,vn,chn,numComb))/...
                        sum(M(:,j,numComb)),q(j, numComb).flow);

                    end
                end
            end
            % sum up power demand for one rotational speed, one
            % characteristic permutation and one mesh matrix permutation
            P_allFan(vn,chn,numComb) = sum(P_fan(:,vn,chn,numComb));
        end
    end
end

% set all negativ values and zeros to NaN
    % only positive (greater than zero) power demands are considered
P_allFan(P_allFan(:)==0)=nan;
P_allFan(P_allFan(:)<0)=nan;
end
```

Anpassung Skripte an Evaluations-Luftverteilung

Da für die Evaluation, wie in Kapitel 6 beschrieben, nur drei Strang-Matrizen zur Auswahl stehen, wird die Funktion **meshMatrix_test.m** der Enumeration entsprechend angepasst.

```
function[M, fanComb] = meshMatrix_test(numMesh, numEl, numDamp, numFan)
%% Initialize mesh matrix
    % the first entries (until numFan) of mesh matrix are describing the
    % fan positions, following the duct positions and the last entries are
    % descrining the damper positions (see cID).

    %% Initialize fan 2-tupel  -  Only valid for 2 meshes
    %  tupel means one couple of boolean values for fans in mesh matrix

    vv = ones(numMesh,1);        % [1 1]
    nn = zeros(numMesh, 1);      % [0 0]
    vn = nn;
    vn(1,1) = true;              % [1 0]
    nv = nn;
    nv(numMesh,1) = true;        % [0 1]

    tupel = [vv, nn, vn, nv];    % all possible fan tupel for 2 meshes

    % get all fan position combinations with repetition
        %fanComb = nmultichoosek(1:length(tupel),numFan);
        fanComb = [1 1 1; 1 1 2; 1 2 2]; % to compare with GA and factorization

    % initialize mesh Matrix for fan, duct and damper positions
    fanPos = zeros(numMesh, numFan, length(fanComb)); % variable fan pos.
    ductPos = [1 1 0;1 0 1];                      % constant duct position
```

```
26      dampPos = eye(numMesh, numDamp);            % constant damper position
27
28      M = zeros(numMesh, numEl, length(fanComb));
29
30      % get mesh matrix for all possible fan position combinations
31      for numComb = 1:length(fanComb)
32          for j = 1:numFan
33              % get fan position combinations
34              fanPos(:,j,numComb) = tupel(:, fanComb(numComb,j));
35              % combine all element positions to mesh matrix
36              M(:,:,numComb) = [fanPos(:,:,numComb),ductPos,dampPos];
37          end
38      end
39
40      % get all mesh matrix with at least one fan per mesh (condition)
41          % initialize matrix to delete not valid mesh matrices
42      deleteM =  zeros(1,length(fanComb));
43      deleteM_0damp = zeros(1,length(fanComb));
44      sumM = zeros(1,numFan,numComb);
45          % loop for fan position combinations to find not valid mesh
46          % matrices and delete them
47      for numComb = 1:length(fanComb)
48          sumM(:,:,numComb) = sum(M(:,1:numFan,numComb));
49
50          for nM = 1:numMesh
51              % get fan combinations where condition is not fulfilled
52              if sum(M(nM,1:numFan,numComb)) == 0
53                  deleteM(numComb) = numComb;
54              end
55
56              % for the possibility of zero dampers in the system this
57              % if-condition is needed
58                  % get fan combinations where all fan in both meshes
59              if numDamp == 0
60                  if sum(sum(M(:,1:numFan,numComb))) == numMesh*numFan
61                  % for different target flow in both meshes this mesh matrix
62                  % configuration will never be a solution
63                  deleteM_0damp(numComb) = numComb;
64                  elseif sum(sum(M(:,1:numFan,numComb))) == numMesh && ...
65                  sum(M(:,1,numComb))== numMesh
66                  deleteM_0damp(numComb) = numComb;
67                  elseif sumM(:,1,numComb) == 2 && sumM(:,2,numComb) == 2 &&...
68                  sumM(:,3,numComb) == 0
69                  deleteM_0damp(numComb) = numComb;
70                  end
71              end
72          end
73      end
74
75      % get fan position combinations where condition is not fulfilled
76      % without zeros
77      endDeleteM = deleteM(deleteM ~= 0);
78
79      % delete mot valid fan position combinations
80      M(:,:,endDeleteM) = [];
81
```

```
    % delete fan position combination which is not possible for
    % zero dampers in the system and different target flow in both meshes
    if numDamp == 0
       endDeleteM_0damp = deleteM_0damp(deleteM_0damp ~= 0);
       M(:,:,endDeleteM_0damp) = [];
    end

end
```

Des weiteren erfolgt die Simulation der Enumeration mit **LoopEvaluation.m** für verschiedene Volumenströme. Ansonsten wird keine Veränderung vorgenommen.

```
%% Loop Enumeration for different target flows
clc
clear

global numFan
global tf1
global tf2

% initialize variables
numFan = 3;                                          % maximum number of fan
    % target flow mesh 1 [m^3/s]
tf1 = 0.1:0.1:0.6;
    % target flow mesh 2 [m^3/s]
tf2 = 0.4:0.4:2.4;

% preallocte variables for loop
elapsedTime_oneMeshMatrix = cell(1, length(tf1));

% loop for different target flow
for x = 1:length(tf1)
    % run enumeration with specific target flows
    try
    [elapsedTime] = Enumeration_loop(x);
    % get elapsed time for calculation and save of each solution
    elapsedTime_oneMeshMatrix{x} = elapsedTime;

    catch ME
    % continue to next loop interation if no alpha was found for any reason
        if (strcmp(ME.identifier,'MATLAB:badsubscript'))
            msg = ME.message;
            disp(msg);
            continue
        end
        disp(['no result was found with targetflows ' 'tf1= ' ...
        num2str(tf1(x)) 'tf2= ' num2str(tf1(x))]);
    end

end
    % elapsed time for all volume flows [s]
    getElapsedTime = cell2mat(elapsedTime_oneMeshMatrix(:));
    totalElapsedTime = sum(getElapsedTime);
    getElapsedTime_total = [getElapsedTime;totalElapsedTime];
```

```

    %% save table with elapsed time
    T = table(...
        {'Laufzeit pro Volumenstrom und fuer alle Volumenstroeme in s'},...
        getElapsedTime);

    writetable(T,'totalElapsedTime');
```