In-Grid

Innovative Grid-Entwicklungen für ingenieurtechnische Anwendungen

AP 2.3: Gridbasierte Simulation und Optimierung mehrstufiger Umformprozesse

Bericht 2.3.2: Prototyp eines verteilten Verfahrens zur Lösung des analysierten Problems und erste Ergebnisse zur Skalierbarkeit und Lösungsqualität

Projektpartner:

Universität Siegen

Institut für Wirtschaftsinformatik

Hölderlinstr. 3

57068 Siegen

Bearbeiter:

Dipl.-Ing. Frank Thilo

1. Inhalt

Nachdem im ersten Jahr des Forschungsprojekts „Innovative Grid-Entwicklungen für ingenieurtechnische Anwendungen“ (In-Grid) im Arbeitspaket „2.3 - Gridbasierte Simulation und Optimierung mehrstufiger Umformprozesse“ typische Probleme der Produkt- und Prozessoptimierung aus der Praxis der Umformtechnik in Form einer mathematisch-abstrakten Beschreibung formalisiert wurden, wurden darauf aufbauend in dem Monaten 13-18 prinzipielle Lösungsansätze zur gridbasieten Optimierung dieser Probleme entwickelt und analysiert.

Im Folgenden werden zunächst verschiedene Use-Cases aus der Umformtechnik identifiziert und in die Systematik aus Bericht 2.3.1 [1] bzw. 3.3.1 [2] (verteilte Optimierung) eingeordnet. Dabei ergibt sich für den Fall der sogenannten Ziehfolgeneinsparung der Bedarf an Verfahren, die über die in AP 3.3 Entwickelten hinausgehen.

Drei verschiedene Ansätze zur Lösung des Problems werden daraufhin beschrieben. Da eine Anbindung an die Umformsimulation Indeed und insbesondere an das CAD-Programm CATIA zur automatischen Netzgenerierung bei Geometrieänderungen zu diesem Zeitpunkt noch nicht umgesetzt ist, wird zunächst eine Ersatzfunktion verwendet. Für diese Ersatzfunktion werden erste Analyseergebnisse für die verschiedenen Ansätze präsentiert und Verteilungsaspekte bei der Lösung im Grid diskutiert. Den Abschluss bildet eine Diskussion noch ausstehender Arbeiten.

2. Problembeschreibung

In der Automobilindustrie werden Bauteile oft durch einen mehrstufigen Umformprozess gefertigt, bei dem eine plane Ausgangsplatine in mehreren nacheinander ablaufenden Schritten in die gewünschte Form gebracht wird.

2.1. Beispiel eines Tiefziehvorgangs

Als Beispiel eines solchen mehrstufigen Umformprozesses ist in Abbildung 1 der vierstufige Tiefziehprozess zur Herstellung eines Außenstützlagers dargestellt. Die initiale Platine wird dabei in der ersten Stufe zu einem Topf gezogen und in der zweiten Stufe weiter geformt. Dazu wird die Platine zwischen dem Niederhalter und der Matrize fixiert und unter Druck durch einen Stempel gemäß der Matrizenvorlage umgeformt. Darauf erfolgt in der dritten Stufe ein Lochen und in der vierten Stufe ein sogenanntes Aufstellen der Ränder. Der Umformprozess wird durch eine Beschneideoperation beendet. Zusätzlich ist das zu fertigende Realteil des Außenstützlagers abgebildet.

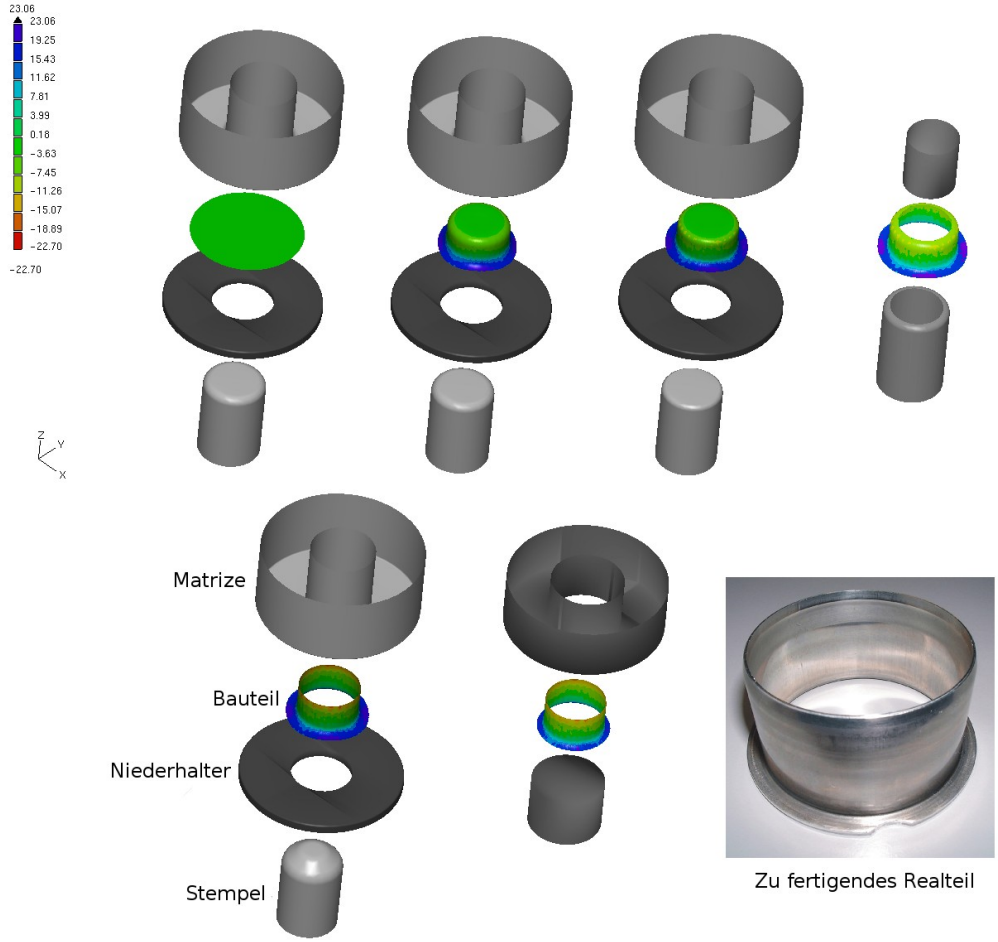

Abbildung 1: Darstellung eines vierstufigen Tiefziehprozesses zur Herstellung eines Außenstützlagers

Ein typisches Optimierungsproblem besteht nun darin, bei einer solchen mehrstufigen Zugdruckumformung Dicke und Durchmesser der Ausgangsplatine zu minimieren. Dabei sind simultan die Geometrien der Werkzeuge (Stempel, Niederhalter und Matrize) und die Prozessparameter aller Stufen (z. B. Verlauf der Niederhaltekräfte und der Ziehgeschwindigkeiten) derart zu bestimmen, dass die Endgeometrie bei minimalen Materialkosten unter Einhaltung einer minimalen Wandstärke ohne Risse und Falten gefertigt wird. Entscheidend ist, dass die Anzahl der Umformstufen dabei nicht in jedem Fall fest gegeben ist, sondern ebenfalls einen Parameter im Sinne der Optimierung darstellt, d.h. es sind nicht nur die Geometrie- und Prozessparameter der einzelnen Stufen zu bestimmen, sondern auch die optimale Anzahl der verwendeten Stufen selbst.

Bei den zu optimierenden Parametern ist zu beachten, dass diese sich gliedern in solche, die unabhängig voneinander für jede Stufe gesetzt werden können (Beispiele hierfür sind Niederhaltekräfte und Ziehgeschwindigkeiten sowie Parameter der abschließenden Beschneideoperation im Anschluss an die letzte Umformstufe), und solche, die bei aufeinanderfolgenden Stufen abhängig voneinander sind (z. B. Einlauf- und Ziehradien).

2.2. Beispiel: Auswirkung der Niederhaltekraft

Beispielhaft für die zu optimierenden Parameter sei anhand des vorstehend beschriebenen Tiefziehszenarios der Einfluss der Niederhaltekraft veranschaulicht. Dazu wurde die Niederhaltekraft zwischen 1000 N und 200.000 N variiert. Die Abbildungen 2 bis 4 zeigen den Effekt der unterschiedlichen Kräfte, indem die Änderung der Blechdicke im Vergleich zur Dicke der Ausgangsplatine am Ende des Umformvorgangs farblich dargestellt wird.

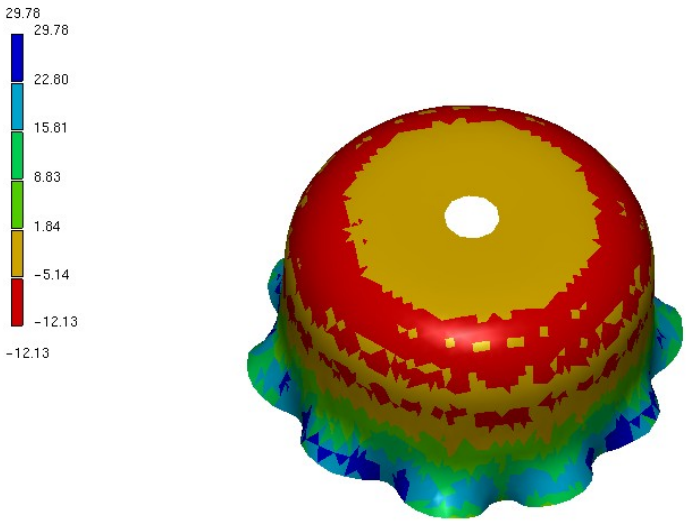

Abbildung 2: Änderung der Blechdicke nach Umformung in %, Niederhaltekraft = 1000 N

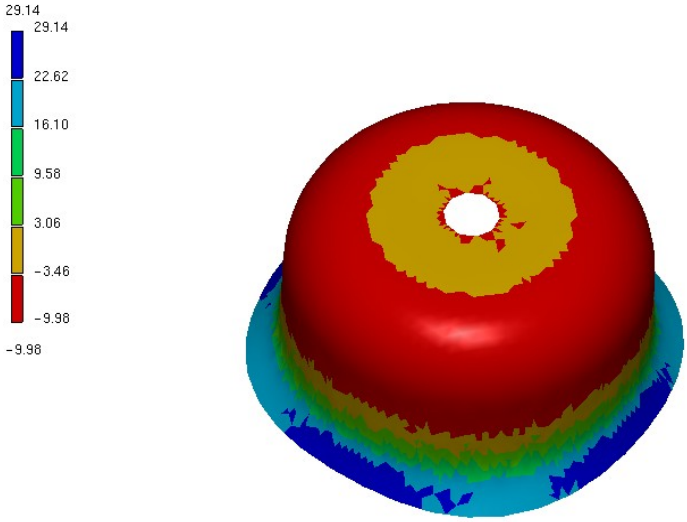

Abbildung 3: Änderung der Blechdicke nach Umformung in %, Niederhaltekraft = 4000 N

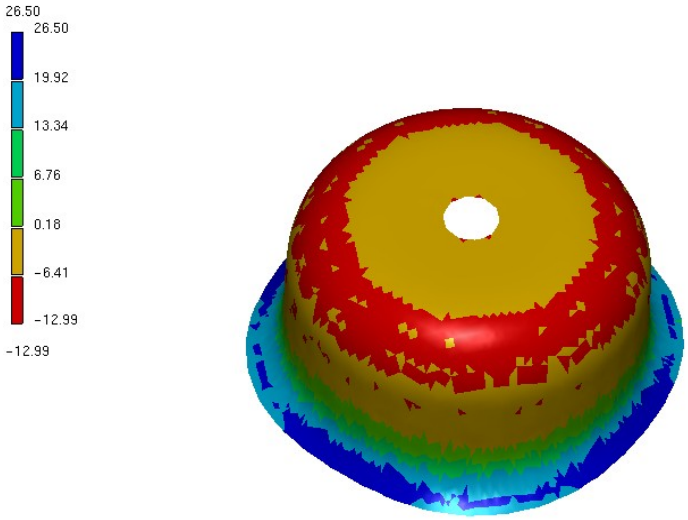

Abbildung 4: Änderung der Blechdicke nach Umformung in %, Niederhaltekraft = 20000 N

Blaue Bereiche bedeuten Verdickungen während rote Bereiche Ausdünnungen bedeuten. Das Szenario mit 4000 N weist die geringste maximale Ausdünnung von etwa 10 % auf. Werden lediglich 1000 N verwendet, so treten deutlich sichtbare, unerwünschte Wellenverformungen auf.

2.3. Typische Problemstellungen

Bei der Analyse typischer Designfragestellungen aus der Umformtechnik konnten aus der Praxis drei verschiedene Klassen von Problemstellungen identifiziert werden:

a) Einhaltung gegebener Blechdicke nach Umformung in n Ziehstufen

- Vorgabe Mindestblechdicke
  - Zielfunktion: Minimales Einsatzgewicht (80% Kosten) durch minimale Platinendicke und -geometrie (=Fläche)
  - Nebenbedingungen:
    - Mindestblechdicke des Endprodukts wird durch Kunden vorgegeben
    - Die ersten n-1 Ziehstufen, die Platinendicke sowie die Platinengeometrie sind variabel, die letzte Ziehstufe durch Geometrie des Endprodukts vorgegeben
- Entscheidungsvariablen:
  - Niederhaltekraft
  - Ziehdurchmesser, Ziehradien, Einlaufradien der Ziehstufen 1 bis n-1

- Platinendicke und -geometrie

- Vorgabe maximale Ausdünnung
  - Zielfunktion: Minimales Einsatzgewicht (80% Kosten) durch minimale Platinengeometrie (=Fläche)
  - Nebenbedingungen:
    - Platinendicke und maximal zulässige Ausdünnung (in Prozent der Platinendicke) vorgegeben
    - Die ersten n-1 Ziehstufen sowie die Platinengeometrie sind variabel, die letzte Ziehstufe durch Geometrie des Endprodukts vorgegeben
  - Entscheidungsvariablen:
    - Niederhaltekraft
    - Ziehdurchmesser, Ziehradien, Einlaufradien der Ziehstufen 1 bis n-1
    - Platinengeometrie

b) Ziehfolgeneinsparung zur inimierung der Betriebsmittelkosten

- Zielfunktion: Blechdicke nach Umformung in optimaler Anzahl k Ziehstufen
  - Vorgabe Mindestblechdicke und maximale Anzahl Ziehstufen n
- Nebenbedingungen:
  - Analog zu a)
  - Anzahl Ziehstufen k<n
- Entscheidungsvariablen:
  - Analog 1zu a)
  - Anzahl Ziehstufen k

c) Verschleißminimierung

- Zielfunktion: Minimaler Anstieg der Kräfte zu Beginn jeder Stufe i (mit 1≤i<n)
- Nebenbedingungen:
  - Analog 1zu a)
  - Steigerung der Kräfte < mmax
- Entscheidungsvariablen:
  - Analog zu a)

2.4. Einordnung in Problemklassen

Im InGrid-Bericht 3.3.1 [2] wurden aus den Anforderungen der verschiedenen Anwendungspakete heraus eine Reihe von Problemklassen definiert. Die grundlegendste dieser Problemklassen ist das Problem des optimalen Entwurfs (engl. *Problem of Optimal Design*, POD). Da sich die anderen in AP 3.3 betrachteten Problemstellungen auf diese Klasse zurückführen lassen, wurden die entwickelten Optimierungsverfahren auf die Klasse POD ausgerichtet. In Abbildung 5 ist eine schematische Darstellung eines POD unter Einsatz einer Simulationssoftware zu sehen.

Eine einzelne Umformstufe kann isoliert für sich betrachtet als ein solches Problem gesehen werden. Der Simulator implementiert dabei eine Blackbox-Funktion, die abhängig von den Größen *z* und *u* ist. *z* sind dabei die nichtveränderlichen Eingangsparameter wie z. B. die generelle durch einen CAD-Entwurf festgelegte Form eines Werkstücks, während *u* die vom Optimierungsalgorithmus zu setzenden Entscheidungsvariablen repräsentiert, beispielsweise Geometrieparameter wie die Dicke oder der Durchmesser einer Platine. Das Ergebnis der Simulation ist *y*, aus dem die zu optimierende Zielfunktion und eventuelle Nebenbedingungen errechnet werden.

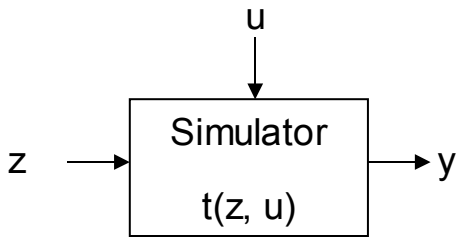

Abbildung 5: Schematische Darstellung eines Problems des optimalen Entwurfs (POD)

Im Falle eines mehrstufigen Tiefziehprozesses kann jeder einzelne Umformvorgang für sich isoliert als Problem des optimalen Entwurfs gesehen werden. Da sich der Gesamtvorgang aus mehreren hintereinander ausgeführten Einzelumformungen zusammensetzt, kann der gesamte Umformprozess als eine Kette aus einzelnen Prozessen gesehen werden, die jeweils den Anforderungen eines POD entsprechen (s. Abbildung 6).

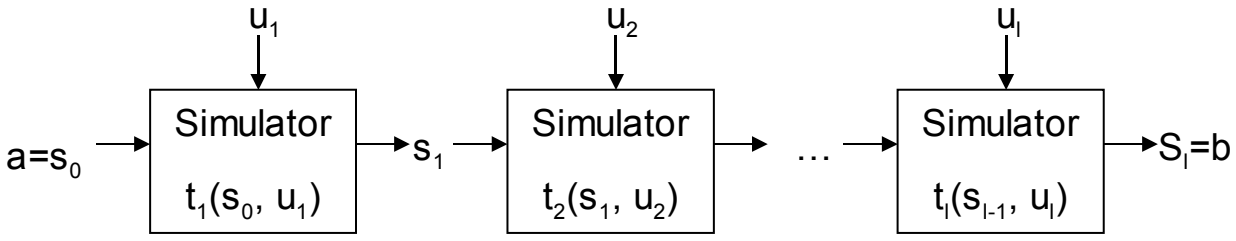

Abbildung 6: Schematische Darstellung eines Problems des Entwurfs diskreter mehrstufiger Prozesse (POM)

Ausgehend von einem Startzustand **a** (der initialen Platine) gilt es, einen vorgegebenen Zielzustand **b** (das fertige Bauteil) in einer vorgegebenen oder optimalen Anzahl von Stufen

$l \in \mathbb{N}$ zu erreichen. Die l diskreten Stufen sind dabei derart gekoppelt, dass der Ausgabezustand der Stufe k dem Eingangszustand in Stufe $k+1$ entspricht. Der Zustandsübergang ist durch den Umformvorgang selbst gegeben, der im Rahmen des virtuellen Prototyping durch einen Simulationslauf möglichst realitätsnah zu ersetzen ist. Als Simulationssoftware wird die vom assoziierten Partner GNS bereitgestellte Software INDEED bzw. in der ebenfalls in diesem Arbeitspaket weiterentwickelten parallelen Variante FETI-INDEED eingesetzt.

Wie in Bericht 2.3.1 [1] im Detail erläutert, lässt sich das mehrstufige Gesamtproblem als Problem des optimalen Entwurfs diskreter mehrstufiger Prozesse (POM) formal wie folgt beschreiben:

$$min\{f(s,u,l) \mid l \in \mathbb{N}, s \in S \subseteq \mathbb{R}^{n \cdot l}, u \in U_{POM} \subseteq \mathbb{R}^{n \cdot l}\}$$

mit $U_{POM} = \Big\{ u = (u_1, u_2, \ldots, u_l) \in \mathbb{R}^{n \cdot l} \mid x$

$$\begin{aligned} & g(s_{k-1}, u_k) \leqslant 0 && \forall\, k = 1, \ldots, l, \\ & h(s_{k-1}, u_k) = 0 && \forall\, k = 1, \ldots, l, \\ & s_k = t_k(s_{k-1}, u_k), && \forall\, k = 1,2, \ldots, l, \\ & s_0 = a,\ s_l = b,\ s_k \in S_k && \forall\, k = 1,2, \ldots, l-1 \Big\}. \end{aligned}$$

Die Variable $l \in \mathbb{N}$ entspricht dabei der Anzahl der Stufen, die neben den Entscheidungsvariablen $u_k, k = 1, \ldots, l$ zu optimieren ist. Die Entscheidungen in Stufe k hängen dabei von den Entscheidungen der vorherigen Stufe ab. Vereinfachend wird hierbei angenommen werden, dass die Anzahl der Variablen je Stufe konstant gleich n ist und somit insgesamt $l \cdot n$ Variablen zu optimieren sind. Die Zustandsvariablen $s_k, k = 1, \ldots, n-1$ repräsentieren den Zustand nach Stufe k, Anfangs- und Endzustand sind dabei mit $s_0 = a$ und $s_l = b$ vorgegeben, während die Zwischenzustände von der Wahl der Entscheidungsvariablen $u_1, \ldots, u_l$ abhängen. Diese sind aus der Menge der zulässigen Entscheidungen $U_k(s_{k-1})$ in Abhängigkeit des durch die vorgeschalteten Stufen gegebenen Zustands s_{k-1} zu wählen. Diese Mengen werden durch die nichtlinearen Restriktionsfunktionen g und h bestimmt. Der Zustand des Systems in Stufe k, s_k, wird ausgehend von einem Zustand s_{k-1} und einer Entscheidung u_k mittels der Transformationsfunktion $t_k(s_{k-1}, u_k)$ bestimmt.

Betrachtet man die stufenspezifischen Entscheidungsvariablen u_k genauer, so setzen sich diese aus den unabhängigen Variablen $x_k \in \mathbb{R}^{n_1}$, den abhängigen Variablen $y_k \in \mathbb{R}^{n_2}$ und den vorgegebenen Entscheidungen der vorherigen Stufe $\bar{y}_k = y_{k-1} \in \mathbb{R}^{n_2}$ zusammen: $u_k = (y_{k-1}, x_k, y_k)$. Diese Aufteilung ist wesentlich, da die abhängigen Variablen eine einfache Dekompositon in unabhängig zu optimierende Teilprobleme verhinden.

Betrachtet man nun die verschiedenen Szenarien aus 2.3, so gilt, dass sowohl bei Szenario a) als auch bei Szenario c) die Anzahl der Umformstufen vorgegeben ist. Daher handelt es sich um ein Problem einer festen Anzahl von gekoppelten Problemen der Klasse POD. Trivialerweise lässt sich das Gesamtproblem in diesem Fall als ein höherdimensionales Problem der Klasse POD auffassen, was die Entscheidungsvariablen aller Einzelstufen umfasst. Zur Lösung können daher direkt die in AP 3.3 entwickelten Verfahren

herangezogen werden, indem immer nur das Gesamtproblem betrachtet wird. In Szenario b) ist hingegen die Anzahl der Stufen variabel und ebenso wie diverse Prozess- und Geometrieparameter zu optimieren. Daher kann das Gesamtproblem nicht einfach als eines der Klasse POD betrachtet werden und die direkte Anwendung der Algorithmen aus AP 3.3 ist nicht möglich.

3. Lösungsansätze

Im Folgenden werden verschiedene Ansätze beschrieben, wie Probleme der Klasse POM gelöst werden können. Dabei werden die in AP 3.3 entwickelten Verfahren zur Lösung von Problemen der Klasse POD soweit möglich als Komponente verwendet.

3.1. Abbildung auf POD

Der einfachste Ansatz ist der Versuch, ein Problem der Klasse POD so zu definieren, dass dessen Entscheidungsvariablen auf ein Problem der eigentlich zu lösenden Klasse POM abgebildet werden können. Dies ist grundsätzlich möglich, da die maximale Anzahl an Stufen l durch l_{max} begrenzt ist. Nun kann ein POD definiert werden, das ausreichend viele Entscheidungsvariablen für den ungünstigsten Fall, nämlich $l = l_{max}$, besitzt. Zusätzlich muss die Stufenanzahl $l_{min} \leq l \leq l_{max}$ in irgendeiner Form codiert werden. Dies kann z. B. mit Hilfe einer zusätzlichen Variablen geschehen. Da bei der Klasse POD nur reellwertige Variablen vorgesehen sind, muss diese anschließend mittels einer Rundungsfunkion auf eine Ganzzahl abgebildet werden. Abhängig vom Wert dieser Variablen l werden die ersten $l \cdot n$ Variablen des POD-Problems auf die des POM-Problems abgebildet. Die restlichen $(l_{max} - l) \cdot n$ Variablen hingegen haben keine Auswirkung und werden verworfen.

Dieser Ansatz hat den großen Nachteil, dass die Anzahl der Entscheidungsvariablen an den worst case angepasst werden muss. Dadurch entsteht ein hochdimensionales Problem, was nur sehr schwierig zu lösen ist. Zudem führt die Abbildung der Variablen dazu, dass nur ein sehr kleiner Teil des gesamten Suchraums überhaupt von Relevanz ist. Die künstlich als reellwertige Variable modellierte Stufenzahl hat weiterhin die Auswirkung, dass Änderungen an dieser Variablen zu sprunghaften Änderungen im Verhalten der Zielfunktion und den Nebenbedingungen führt. Es ist zu erwarten, dass dies alles bei Einsatz der auf Probleme der Klasse POD hin entworfenen Suchverfahren zu keinen guten Resultaten führt. Daher wird dieser Ansatz nicht weiter verfolgt.

3.2. Zerlegung in POD-Teilprobleme

Ein alternativer Ansatz ist, nicht das Gesamtproblem auf ein einziges POD-Problem abzubilden, sondern es stattdessen als eine Menge von Teilproblemen aufzufassen. Für jede mögliche Stufenzahl $l_{min} \leq l \leq l_{max}$ des POM-Problems ergibt sich ein Teilproblem, das als ein Problem der Klasse POD interpretiert wird. Jedes dieser Probleme hat dabei eine unterschiedliche Anzahl von Entscheidungsvariablen $l \cdot n$ und kann unabhängig voneinander gelöst werden. Die Lösung des Gesamtproblems ergibt sich dann als beste Lösung aller dieser Teilprobleme. Diese kann mit einem direkten Brute-Force-Ansatz oder einer zweistufigen Suche gefunden werden:

a) Vollständige Enumeration

Der einfachste Ansatz ist, alle anfallenden Teilprobleme separat zu lösen und schließlich aus allen Teillösungen die beste auszuwählen. Dazu sind $l_{max} - l_{min}$ Probleme der Klasse POD zu lösen. Diese Teilprobleme unterscheiden sich in der Problemdimension $l \cdot n$ voneinander. Die Probleme für kleine l lassen sich in der Regel deutlich einfacher und mit weniger Auswertungen (aufwendige Simulationsläufe) lösen als solche mit großem l. Abhängig vom verwendeten Algorithmus lassen sich bei den kleineren Teilproblemen eventuell nicht alle zur Verfügung stehenden Rechenressourcen effizient nutzen, während dies bei den höherdimensionalen Problemen in der Regel möglich sein wird.

Da die Teilprobleme aber vollkommen unabhängig voneinander gelöst werden können, ist es auch möglich, mehrere dieser Probleme simultan zu lösen. Dabei kann dann die Gesamtzahl der zur Verfügung stehenden Ressourcen (CPUs) auf die Teilprobleme verteilt werden. Da die Effizienz der Algorithmen i. A. mit steigendem Parallelitätsgrad abnimmt (s. auch Bericht 3.3.1 [2] und 3.3.2 [3]) und durch diese Maßnahme der Parallelitätsgrad jedes Teilproblems sinkt, kann so meist eine höhere algorithmische Effizienz erreicht werden. Allerdings stellt sich die Frage, welche Aufteilung für die Gesamteffizienz optimal ist. Bei einer statischen Aufteilung der Ressourcen sinkt die Effizienz, sobald eines der Teilprobleme vorzeitig gelöst ist, da dann die entsprechenden Prozessoren brachliegen.

Für eine optimale Aufteilung muss die relative Laufzeit der einzelnen Teiloptimierungen a-priori abgeschätzt werden. Dies ist im allgemeinen Fall jedoch nur sehr schwer möglich. Die Laufzeit setzt sich zusammen aus der Anzahl der Evaluierungen, die zur Lösung des Problems notwendig sind, der parallelen Effizienz des verwendeten Verfahrens bei gegebener Problemdimension und Parallelitätsgrad sowie der Laufzeit der einzelnen Simulationen. Die Anzahl der notwendigen Evaluierungen steigt i. A. naturgemäß mit der Anzahl der Entscheidungsvariablen. Bei den meisten Suchverfahren und Problemstellungen lässt sich aber kein einfacher Zusammenhang finden. Die parallele Effizienz der Algorithmen sinkt mit steigendem Parallelitätsgrad, ist aber stark abhängig vom verwendeten Verfahren. Bei höheren Problemdimensionen sinkt die Effizienz in der Regel langsamer. Die Laufzeit der Simulationen steigt mit der Anzahl der Stufen, da jeder einzelne Umformschritt rechenaufwendig simuliert werden muss.

Insgesamt ist die Abschätzung der benötigten Laufzeit sehr problematisch und voraussichtlich mit einem großen Fehler behaftet. Nimmt an als Faustregel eine proportionale Abhängigkeit zwischen Problemdimension und der Anzahl der benötigten Auswertungen an, ebenso einen linearen Zusammenhang zwischen der Stufenanzahl und der Rechenzeit pro Simulation, und geht man davon aus, dass der Algorithmus im Bereich sehr hoher paralleler Effizienz betrieben wird, so ergibt sich eine Abschätzung der relativen Laufzeiten zu: $T(l) = l^2 \div p_l$. Die Anzahl der Prozessoren p_l für jedes Teilproblem sollte also unter diesen Annahmen proportional zum Quadrat der Stufenzahl l gewählt werden. Um Effizienzverluste aufgrund einer fehlerhaften a-priori-Abschätzung zu vermeiden, wäre auch eine dynamische Reallokation von brachliegenden Ressourcen auf die noch nicht gelösten Teilprobleme denkbar. Dazu müssen die Optimierungsverfahren sich aber zur Laufzeit an einen wechselnden Parallelitätsgrad anpassen können.

b) Bilevel-Dekomposition

Das in die POD-Teilprobleme zerlegte Gesamtproblem kann auch als zweistufige Suche angesehen werden. dabei bilden die mit den verteilten Optimierungsverfahren zu lösenden Teilprobleme die untere Ebene, während auf der oberen Ebene ein eindimensionales

Problem zu lösen ist, indem die optimale Stufenzahl bestimmt wird. Unter der Annahme der Unimodalität dieser eindimensionalen Funktion (d. h. es existiert lediglich ein lokales Optimum) kann anstelle der vollständigen Enumeration ein intelligenteres Suchverfahren eingesetzt werden. dazu eignen sich Verfahren wie die binäre Suche oder die Fibonacci-Suche. Der Vorteil dieser Verfahren ist, dass mit jedem Teilproblem, was gelöst wird, das Intervall, in dem die optimale Stufenzahl liegen muss, weiter eingeschränkt wird und die Gesamtzahl der auszuwertenden Teilprobleme nur logarithmisch mit der Zahl der möglichen Stufenanzahlen wächst. Dadurch kann die Auswertung zumindest einiger Teilprobleme vermieden werden.

Der Nachteil des Einsatzes eines eindimensionalen Suchverfahrens ist allerdings, dass eine simultane Auswertung der Teilprobleme nicht mehr problemlos möglich ist, da die Entscheidung, welche Teilprobleme ausgewertet werden müssen, zum Teil vom Resultat der vorherigen Auswertungen abhängt. Der mögliche Parallelitätsgrad kann durch Einsatz von spekulativen Suchschritten erhöht werden, allerdings wird dadurch die Anzahl der eingesparten Teilprobleme wieder geringer.

Tabelle 1 illustriert die Einsparungen bei Einsatz der Fibonacci-Suche anhand eines Beispiels. Die Stufenanzahl kann zwischen 3 und 10 variiert werden, d. h. bei Einsazt der vollständigen Enumeration entstehen 8 zu lösende Teilprobleme. In der Zeile „Zielfunktion“ sind die (zunächst unbekannten) Minima der Teilprobleme aufgeführt. Die Zeile „Auswertung“ beschreibt, welche Teilprobleme von der Fibonacci-Suche in welcher Reihenfolge (*a* bis *d*) evaluiert werden. In diesem Beispiel werden also nur vier der acht Teilprobleme berechnet. Insbesondere kann auf die Berechnung der aufwendigen höherdimensionalen Probleme verzichtet werden. Dadurch ergibt sich unter Annahme der relativen Laufzeiten wie oben erläutert eine Gesamteinsparung von ca. 66% der Stufenberechnungen.

Stufen	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Zielfunktion	32	27	24	23	29	34	39	46
Auswertung		c	b	d	a			

Tabelle 1: Auswertungsreihenfolge bei Fibonacci-Suche

3.3. Stufenspezifische Vorwärtsrechnung

Die bisher beschriebenen Lösungsansätze berücksichtigen nicht explizit die Stufenstruktur des Problems, sondern betrachten jedes (Teil-)Problem als eine homogene Blackbox. Die einzelnen Stufen können zwar nicht direkt voneinander separiert werden, um eine unabhängige Optimierung jeder Stufe zu vollziehen, aber der Wert der Zielfunkton und der Nebenbedingungen nach Durchführung nur des Teils des Gesamtprozesses lässt durchaus Rückschlüsse zu, ob das Zwischenergebnis vielversprechend ist und die Anwendung weiterer Tiefziehprozesse noch zu einem guten Ergebnis führen wird. Das Vorgehen, das Zwischenprodukt bereits zu bewerten, entspricht auch dem manuellen Vorgehen eines Ingenieurs, der hier sein Fachwissen einfließen lässt.

Die Idee der sogenannten stufenspezifischen Vorwärtsrechnung [4] ist ähnlich zu der eines Branch-and-Bound-Verfahrens. Dabei wird ein Suchbaum aufgebaut, der jede zu treffende Entscheidung bzgl. der Entscheidungsvariablen als Verzweigung abbildet. Um nicht den gesamten Baum durchsuchen zu müssen, wird anhand von Abschätzungen versucht obere oder untere Schranken für Teilbäume zu ermitteln. Ergibt sich durch den Vergleich einer

Schranke und des bisher gefundenen Optimums, dass der entsprechende Teilbaum zu keiner Verbesserung führen kann, so kann der entsprechende Zweig abgeschnitten werden.

Um auf jeder Ebene des Baumes nur eine endliche Zahl von Verzweigungen zu haben, müssen die Entscheidungsvariablen diskretisiert werden, so dass sie innerhalb der gegebenen Unter- und Obergrenzen nur eine eher kleine Zahl von Werten annehmen können. Eine feine Diskretisierung führt potentiell zu besseren Ergebnissen, lässt den Suchbaum aber schnell anwachsen, während eine grobe Diskretisierung zu einem schnellen Verfahren führt, welches aber wahrscheinlich keine optimale Belegung der Variablen finden kann.

Die Stufenstruktur des Problems findet sich direkt im Aufbau des Suchbaums wieder, indem die Entscheidungen einer jeden Stufe den Verzweigungen einer Ebene des Baums entsprechen. Bei den Entscheidungen wird zwischen den abhängigen und den unabhängigen Variablen unterschieden. Die abhängigen Variablen beeinflussen die Zielfunktionskomponenten von mindestens zwei benachbarten Stufen in nicht-linearer Weise, so dass die Optimierung einer Stufe für diese Variablen nicht ohne Berücksichtigung der Nachbarstufe möglich ist. Bei gegebener Belegung der abhängigen Variablen können hingegen die unabhängigen Variablen separat optimiert werden. Daher werden die abhängigen Variablen wie beschrieben diskretisiert, so dass für jede mögliche Belegung dieser Variablen auf einer Stufe ein Knoten des Suchbaums existiert.

Die unabhängigen Variablen jeder Stufe bleiben hingegen kontinuierlich und werden stufenspezifisch mit einem Optimierungsverfahren zur Lösung von POD-Problemen optimiert. Die Anzahl der abhängigen und unabhängigen Variablen je Stufe ist typischerweise klein, so dass die zu lösenden POD-Teilprobleme niedrigdimensional sind und bei nicht zu feiner Diskretisierung die Anzahl der Verzweigungen im Baum auch klein bleibt. Bei einer Diskretisierung auf d verschiedene Werte pro abhängiger Variable und bei n_2 abhängigen Variablen in jeder Stufe, ergeben sich in jeder Stufe d^{n2} Verzweigungen.

In Abbildung 7 ist der prinzipielle Ablauf des Verfahrens dargestellt. Zunächst wird das Gesamtproblem in einer Initialisierung in abhängige und unabhängige Variablen und die einzelnen Stufen aufgeteilt und die entsprechenden Datenstrukturen aufgebaut. Die aktuell zu betrachtete Stufe wird durch die Variable k repräsentiert, die zunächst auf 1 gesetzt wird. Der erste Schritt ist dann die Diskretisierung der abhängigen Variablen der jeweiligen (also zunächst der ersten) Stufe, d. h. alle möglichen d^{n2} Belegungen werden gebildet. Aus diesen möglichen Belegungen wird durch einen Selektionsalgorithmus eine kleine Zahl ausgewählt. In diese Selektion kann problemspezifisches Wissen einfließen, oder sie kann im einfachsten Fall rein zufällig sein. An dieser Stelle können unter Umständen bereits Lösungen aussortiert werden, für die bereits ersichtlich ist, dass Nebenbedingungen verletzt sind und auch durch nachfolgende Ziehstufen nicht mehr erfüllt werden können.

Für jede ausgewählte Belegung der abhängigen Variablen wird nun ein nichtlineares Teilproblem (NLP) gelöst, indem ein POD-Optimierungsverfahren optimale Werte für die unabhängigen Variablen dieser Stufe findet. Zur Berechnung der Zielfunktion und der Nebenbedingungen wird dazu die Umformsimulation auch nur soweit durchgeführt, wie es zur Berechnung der ersten Stufe notwendig ist. Diejenigen Nebenbedingungen, die sich auf den Gesamtvorgang beziehen und nach den ersten Stufen noch nicht erfüllt sein können, müssen speziell behandelt werden, z. B. indem die Verletzung über eine Straffunktion in den Zielfunktionswert einfließt. Dadurch werden Zwischenlösungen bevorzugt, die gute Voraussetzungen für die nachfolgenden Stufen bieten.

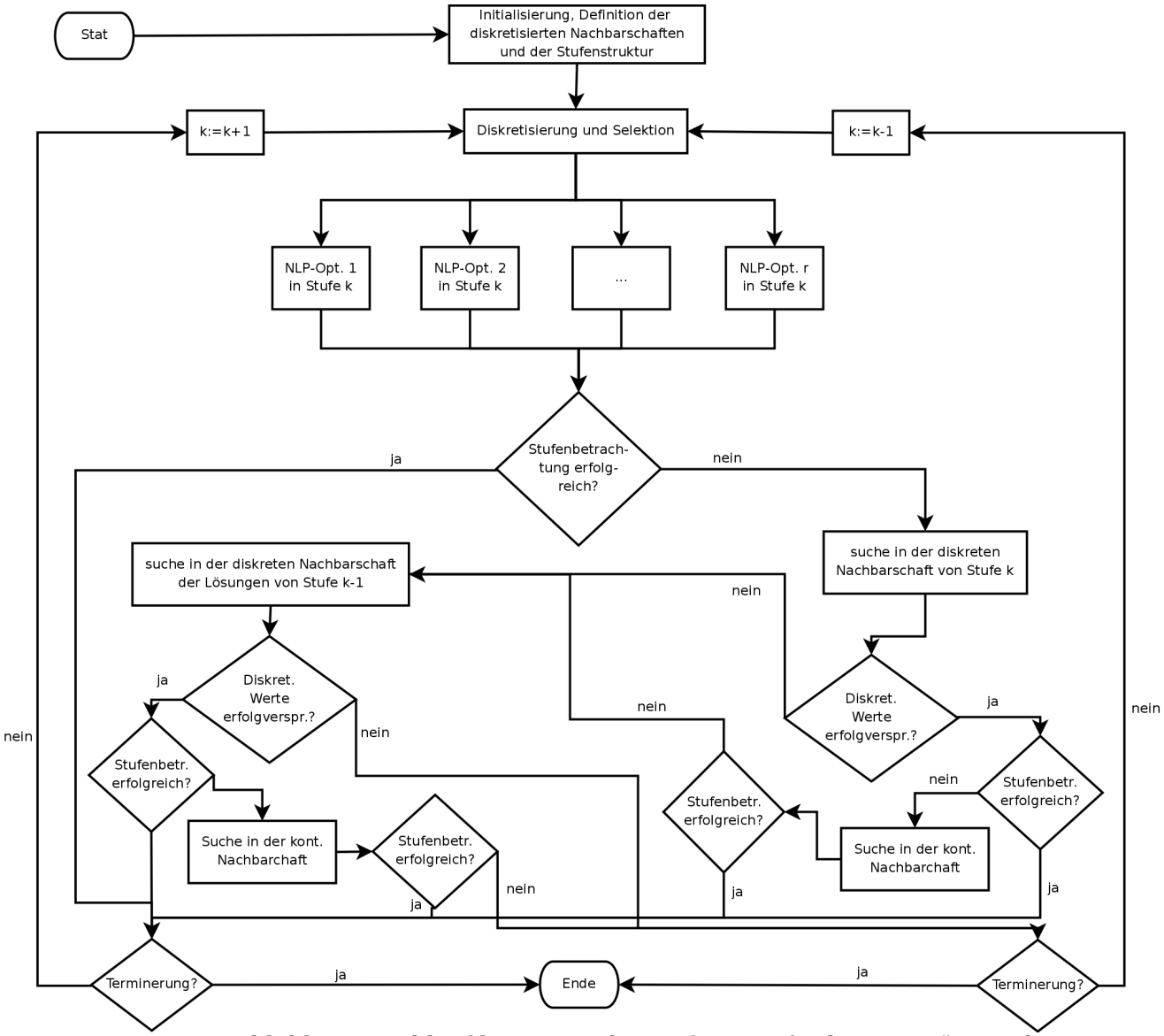

Abbildung 7: Ablaufdiagramm der stufenspezifischen Vorwärtsrechnung

Liegt das Ergebnis der Optimierung der Teilprobleme vor, so wird die Stufenbetrachtung als erfolgreich oder nicht erfolgreich klassifiziert. Eine erfolgreiche Auswertung liegt dann vor, wenn ein neues globales Optimum gefunden wurde. In diesem Fall wird die Stufenvariable *k* erhöht und der Vorgang wird mit der nächsten Stufe fortgesetzt. Konnte keine neues Optimum gefunden werden, so werden die bisher nicht betrachteten Belegungen der abhängigen Variablen der aktuellen Stufe betrachtet. Dazu werden diese mit den besten gefundenen Werten für die unabhängigen Variablen kombiniert und die entstehenden Lösungen bewertet. Nun werden drei Fälle unterschieden: Wurde ein neues Optimum gefunden, wird mit der nächsten Stufe fortgefahren. Liegt der beste Wert nahe am Optimum, so wird versucht, die Lösung zu verbessern, indem die kontinuierlichen Variablen für die neue Belegung der diskretisierten Variablen neu optimiert werden. Hat dieser Versuch Erfolg, so wird abermals mit der nächsten Stufe fortgefahren.

Schlägt dies hingegen fehl oder gab es zuvor bereits keine aussichtsreiche Lösung in der Nähe des bisherigen Optimums, so wird zusätzlich in der diskreten Nachbarschaft der vorherigen Stufe gesucht, d. h. bisher nicht betrachtete Belegungen der abhängigen Variablen der Vorstufe werden mit den kontinuierlichen Variablen der besten Zwischenlösung kombiniert. Wird nun eine neue beste Lösung gefunden, so war der Schritt erfolgreich und es wird zur nächsten Stufe verzweigt. War dies nicht der Fall, wird noch versucht, die kontinuierlichen Variablen per Optimierung anzupassen. Schlägt auch dies fehl, so gilt die Stufenbetrachtung endgültig als gescheitert, k wird dekrementiert und es wird zur Betrachtung der vorherigen Stufe zurückgekehrt, wo bisher noch nicht selektierte diskrete Lösungen betrachtet werden.

Das Verfahren terminiert, wenn entweder der komplette Suchbaum durchsucht oder beschnitten wurde, oder alternativ wenn weiteren Kriterien wie die maximale Anzahl der Iterationen oder die maximale Anzahl der Stufenrückschritte erreicht werden. Das Ergebnis besteht aus der Stufenzahl der besten gefunden Lösung, den diskretisierten Werten für die abhängigen Variablen und den Werten für die unabhängigen, kontinuierlichen Variablen. Abschließend kann eine lokale Suche eingesetzt werden, um vor allem die Werte der diskretisierten Variablen noch leicht zu verbessern, da das Optimum mit hoher Wahrscheinlichkeit nach nicht auf den gewählten Diskretisierungsstufen liegt.

Die Vorteile des Verfahrens sind, dass die anfallenden POD-Teilprobleme sehr niedrigdimensional sind und sich daher erfahrungsgemäß gut und schnell lösen lassen. Außerdem ist zur Bestimmung der Zielfunktion und der Nebenbedingungen nur die Simulation des Umformschritts der betrachteten Stufe notwendig, während bei den anderen Ansätzen immer der komplette mehrstufige Prozess simuliert werden muss. Dadurch kann die Laufzeit der aufwendigen Simulation jeweils deutlich abgesenkt werden. Eine Parallelisierung bietet sich an mehreren Stellen an. Zunächst einmal kann innerhalb jedes POD-Teilproblems die übliche Parallelität der Algorithmen aus AP 3.3 eingesetzt werden. Die durch den Selektionsschritt zu Beginn der Stufenbetrachtung entstehenden Teilprobleme können zudem unabhängig voneinander simultan gelöst werden. Durch Anpassung der Zahl der selektierten Entscheidungen kann der Parallelitätsgrad hier leicht angepasst werden. Ebenso kann die Auswertung der Lösungen, die durch Kombination der bisher noch nicht betrachteten Belegungen der diskreten Variablen mit den optimierten kontinuierlichen Variablen entstehen, simultan erfolgen.

Während von den in Abschnitt 2.3 beschriebenen Szenarien nur eines die Verwendung spezialisierter Verfahren erfordert, lassen sich grundsätzlich auch die Szenarien a) und c) mit fester Stufenzahl mit hier beschriebenen Verfahren der stufenspezifischen Vorwärtsrechnung lösen. Es muss lediglich berücksichtigt werden, dass Lösungen erst als globales Optimum akzeptiert werden, wenn die entsprechende Stufenzahl erreicht ist.

4. Evaluierung

Ein geeignetes Optimierungsverfahren ist nur eine Komponente, die notwendg ist, um ein reales Problem aus der Umformtechnik zu optimieren. Weitere Komponenten im re- und Postprocessing der Simulation sind notwendig, um einerseits die vom Suchverfahren bestimmten Werte der Entscheidungsvariablen in Geometrie- oder Prozessparameter umzusetzen und andererseits das Ergebnis der Umformsimulation zu bewerten und in Form der Zielfunktion bzw. Nebenbedingungen wieder dem Algorithmus zur Verfügung zu stellen. Technisch problematisch ist hier insbesondere die Einbindung des CAD-Programms CATIA, welches dafür zuständig ist, aus einer Beschreibung der Geometrie die Eingabedaten für die Simulation INDEED in Form eines Gitternetzes zu generieren. Diese

Netzgenerierung muss immer dann durchlaufen werden, wenn sich ein Geometrieparameter (wie z. B. die Dicke der Platine) ändert. CATIA ist prinzipiell über eine Visual-Basic-Schnittstelle programmierbar. Es gibt aber zunächst noch eine Reihe von technischen Problemen zu lösen, bevor eine Anbindung der Netzgenerierung an eine Optimierung im Grid möglich ist.

Um die hier beschriebenen Verfahren aber bereits einmal prototypisch implementieren und testen zu können, wurde versucht, durch Einsatz einer Ersatzfunktion die wichtigsten Charakteristika der mehrstufigen Umformprobleme nachzubilden.

4.1. Ersatzfunktion

Die Ersatzfunktion für Probleme der Klasse POM muss mehrere wichtige Eigenschaften der realen Umformtechnik bzw. derer praxisnaher Simulation nachbilden. Zum ersten ist dies die grundlegende Mehrstufigkeit, wobei sich die Anzahl der Stufen anpassen lassen muss. Die Ersatzfunktion ist als Minimierungsproblem wie folgt definiert (s. auch [5]):

$$\min_{x,l}\left\{ f(x,l) \;=\; f(x,0)+\sum_{k=1}^{l}\left\| \begin{array}{l} -50\cdot e \;+\; (k-1)^2 \\ +100\cdot(x_{4k-2}-x_{4k-3}^2)^2 \;+\; (1-x_{4k-3})^2 \\ +100\cdot(x_{4k}-x_{4k-1}^2)^2 \;+\; (1-x_{4k-1})^2 \\ +(x_{4k}-(k-1))^2 \;\cdot\; (x_{4k+1}-(k-1))^2 \end{array} \right\| \; l\in\mathbb{N}, x\in U_{MS}\subseteq\mathbb{R}^{4l+1} \right\}$$

Die Anzahl der Stufen ist dabei über die Variable *l* gegeben. Die Zielfunktion besteht aus einer Summe über die Anzahl der Stufen. Jeder Summand besteht aus vier additiven Termen. Der erste ist ein Skalierungsterm. Über den Parameter *e* lässt sich zudem kontrollieren, bei welcher Stufenanzahl das globale Minimum erreicht wird. Es folgen zwei Terme, die an die bekannte Rosenbrock-Testfunktion angelehnt sind. Schließlich folgt der sogenannte gemischte Term, der die Kopplung zur benachbarten Stufe herstellt, da die dort verwendete Entscheidungsvariable x_{4k+1} mit $x_{4k'-3}$ der nächsten Stufe mit $k' = k+1$ identisch ist. Über diese Kopplung wird auch die wichtige Eigenschaft der Nichtseparierbarkeit erreicht, d. h. dass es nicht möglich ist, die Gesamtfunktion einfach in Teilprobleme zu zerlegen, die dann getrennt optimiert werden können.

Neben der Zielfunktion sind Restriktionen zu berücksichtigen, die den gültigen Bereich der Entscheidungsvariablen einschränken. Zunächst werden für alle Variablen gleiche Unter- und Obergrenzen definiert:

$$x_{4k-i}\in[0,3], i\in\{-1,0,1,2,3\}$$

Als nächstes werden Nebenbedingungen für die unabhängigen Variablen jeder Stufe definiert. Diese Variablen können unabhängig von den Entscheidungen der vorhergehenden und der nachfolgenden Stufe optimal bestimmt werden. Die entsprechenden Restriktionen beziehen sich ebenfalls nur auf die Variablen einer Stufe:

$$-(x_{4k-2}\cdot x_{4k-1})+G\leqslant 0, \quad G\in[0,9]$$

Durch Wahl des Parameters G kann der zulässige Bereich der unabhängigen Variablen erweitert oder eingeschränkt werden.

Die Restriktionen der abhängigen Variablen unterteilen sich einerseits in stufenspezifische Restriktionen und anderer in Restriktionen die sich auf den Gesamtprozess über alle *l* Stufen beziehen. Die stufenspezifischen Restriktionen sind:

$$g(x_{4k}) = x_{4k}^2 \leqslant H, \quad \forall k = 1,\ldots,n, H > 0$$

$$h(x_{4k+1}) = x_{4k+1}^2 \leqslant I, \quad \forall k = 1,\ldots,n-1, I > 0$$

Die Restriktionen des Gesamtprozesses sind:

$$\sum_{k=1}^{n} g(x_{4k}) \leqslant J, \quad J > 0$$

$$\sum_{k=1}^{n} h(x_{4k+1}) \geqslant K, \quad K > 0$$

Während die stufenspezifischen Restriktionen lediglich den Wertebereich jeder einzelnen abhängigen Variablen weiter einschränken, sind die Variablen in der Gesamtprozessrestriktion miteinander verknüpft. Die Summe ihrer Quadrate ist für die ersten abhängigen Variablen nach oben und für die zweiten nach unten begrenzt. Über die Wahl der Parameter H, I, J und K kann der zulässige Bereich der abhängigen Variablen in weiten Grenzen variiert werden.

In Tabelle 2 sind die gewählten Parameter der Ersatzfunktion zusammengefasst:

Parameter	e	f(x,0)	G	H	I	J	K
Wert	0.25	250*e	0.85	4	2.5	6	15

Tabelle 2: Wahl der Parameter für die Ersatzfunktion

Der Parameter *e* wurde mit 0.25 so gewählt, dass das globale Minimum ohne Berücksichtigung der Restriktionen bei *l=4* mit f(x,4) = 26.923524 erreicht wird. Ausgehend von diesem Optimum wurden die Werte für H, I, J und K so bestimmt, dass das Optimum im gültigen Bereich liegt. Der Parameter G, der die Nebenbedingung für die unabhängigen Variablen beeinflusst, wurde so gewählt, dass etwa 70% des Suchraums dieser Variablen im gültigen Bereich liegt, das Optimum des nichtbeschränkten Problems aber außerhalb liegt.

4.2. Ergebnisse

Die Verfahren der vollständigen Enumeration (s. 3.2.a), der Bi-Level-Dekomposition (s. 3.2.b) und der stufenspezifischen Vorwärtsrechnung (s. 3.3) wurden prototypisch implementiert und mit der mehrstufigen Ersatzfunktion getestet. Da die Berechnungszeit der Ersatzfunktion vernachlässigbar klein ist,sie aber stellvertretend für eine rechenaufwendige Simulation sein soll, wurde die Berechnung nicht real verteilt durchgeführt, sondern stattdessen die in Bericht 3.3.1 [2] beschriebene “virtuelle Clusterumgebung“ verwendet. In Tabelle 3 sind die Optimierungsergebnisse für die 3 Szenarien unter Verwendung von 75 virtuellen CPUs aufgezählt. Als Optimierungsverfahren zur Lösung der auftretenden POD-Teilprobleme wurde jeweils ein paralleler Scatter-Search-Ansatz (PSS [2]) verwendet. Dieser Algorithmus weist auch bei kleinen

Problemdimensionen bei 75 CPUs eine sehr hohe parallele Effizienz auf, so dass zunächst alle Teilprobleme sequentiell nacheinander anstatt simultan gelöst wurden.

Als minimale und maximale Stufenanzahl wurden 1 und 5 gesetzt. Durch diese recht engen Grenzen konnte die bei der Bi-Level-Dekomposition eingesetzte Fibonaccisuche lediglich die Auswertung der minimalen Stufenanzahl von 1 einsparen, die ohnehin die wenigsten Auswertungen benötigt. Für die stufenspezifische Vorwärtsrechnung wurde eine eher grobe Diskretisierung von 5 Werten pro Entscheidungsvariable gewählt. Zum Vergleich der Laufzeiten wurde die insgesamt benötigte Anzahl der sogenannten Stufenauswertungen gezählt. Bei einem 4-stufigen Problem muss der Optimierungsalgorithmus bei der vollständigen Enumeration beispielsweise für jede Berechnung der Zielfunktion entsprechend 4 Stufenauswertungen in Anspruch nehmen. Dahinter steckt die vereinfachende Annahme, dass die Simulation eines 4-stufigen Umformvorgangs sich aus 4 Einzelsimulatonen gleichen Aufwands zusammensetzt. Da die stufenspezifische Vorwärtsrechnung jeweils nur einen isolierten Umformvorgang betrachtet, kostet hier jede Berechnung der Zielfunktion nur eine Stufenauswertung.

Lösungsansatz	**Zielfunktionswert**	**Anzahl Stufenauswertungen**
vollständige Enumeration	30.86	242514
Bi-Level-Dekomposition	30.86	226346
stufenspezifische Vorwärtsrechnung	31.58	47202

Tabelle 3: Optimierungsergebnisse der Ersatzfunktion

Vergleicht man die Ergebnisse miteinander, so sieht man, dass die stufenspezifische Vorwärtsrechnung deutlich weniger Auswertungen benötigt als die beiden anderen Verfahren. Der beste gefundene Zielfunktionswert ist dabei allerdings schlechter, was u. a. auf die grobe Diskretisierung zurückzuführen sein kann. Der Einsatz der Fibonaccisuche erreicht im hier gegebenen Fall erwartungsgemäß nur eine kleine Verbesserung. Bei einer Erhöhung der maximal erlaubten Stufenanzahl auf 6 oder 7 sollte das Verfahren deutlich mehr an Einsparung bieten.

Die Anzahl der verfügbaren Prozessoren, die Parameter der Verfahren und der POD-Optimierungsalgorithmus wurden hier der Einfachheit wegen so gewählt, dass die Ressourcen fast ideal ausgenutzt werden konnten. Im Allgemeinen wird dies nicht immer so einfach möglich sein, wodurch es dann zu Einbußen bei der parallelen Effizienz kommen wird. Durch Abschätzungen der Laufzeiten (s. auch Abschnitt 3.2), eine geschickte Aufteilung der Ressourcen auf simultan berechnete Teilprobleme und eine angepasste Wahl der Parameter der Algorithmen sollte es dennoch möglich sein, eine hihe Effizienz zu erreichen. Dazu sind allerdings weitere Analysen und Testrechnungen notwendig, um die Wahl dieser Parameter zu automatisieren und auch im heterogenen Gridumfeld eine robuste Lösung anbieten zu können.

5. Ausblick

Die grundsätzliche Funktionsfähigkeit der verschiedenen Ansätze wurde demonstriert und verschiedene Vor- und Nachteile der Verfahren analysiert sowie eine Reihe von Skalierbarkeitsaspekten diskutiert. Die Verfahren wurden anhand eines ersten Optimierungstests mit einer Ersatzfunktion überprüft. Es sind allerdings weitere Tests mit veränderten Bedingungen, weiteren Optimierungsalgorithmen aus AP 3.3 und

verschiedenen Parallelisierungsgraden notwendig. Die Verfahren müssen schließlich in Form eines Grid-Service in die serviceorientierte Architektur integriert werden.

Die Anbindung an die Umformsimulation INDEED und das CAD-Programm CATIA müssen softwaretechnisch mit dem Ziel angegangen werden, diese als Service zu kapseln und mit den Optimierungsverfahren koppeln zu können. Schließlich sollte es auch möglich sein, das sequentielle INDEED durch das parallele FETI-INDEED zu ersetzen, und so noch einen weiteren Grad der Parallelisierung hinzuzufügen.

6. Literatur

[1] Thilo, F., AP 2.3, Bericht 2.3.1, Innovative Grid-Entwicklungen für ingenieurtechnische Anwendungen (InGrid), 2006.

[2] Thilo, F., AP 3.3, Bericht 3.3.1, Innovative Grid-Entwicklungen für ingenieurtechnische Anwendungen (InGrid), 2006.

[3] Thilo, F., AP 3.3, Bericht 3.3.2, Innovative Grid-Entwicklungen für ingenieurtechnische Anwendungen (InGrid), 2007.

[4] Kutsch, C., „Investigating Distributed Approaches for Solving Discrete, Multistage Optimization Problems“, Diplomarbeit, Siegen, 2004.

[5] Gerdes, M., „Verteilte Optimierung mehrstufiger Fertigungsprozesse“, Dissertation, Siegen, 2005.