DOI: 10.2195/lj_Proc_schieweck_de_201605_01
URN: urn:nbn:de:0009-14-43599

Intralogistik als Anwendungsgebiet der Antwortmengenprogrammierung – Potenzialanalyse

Applying Answer Set Programming to Inhouse Logistics – Potential Analysis

***Steffen Schieweck*[1,2]**
***Gabriele Kern-Isberner*[1]**
***Michael ten Hompel*[2]**

[1]Lehrstuhl Informatik 1 – Information Engineering
Fakultät für Informatik
[2]Lehrstuhl für Förder- und Lagerwesen
Fakultät Maschinenbau
TU Dortmund

Der Stellenwert der Intralogistik ist in den vergangenen Jahren aufgrund eines zunehmend beschleunigten und globalen Marktes deutlich gestiegen. Die Intralogistik bietet Problemstellungen, deren Lösung respektive Lösungsunterstützung weiterer Forschung bedarf. Antwortmengenprogrammierung auf der anderen Seite bietet hervorragende Eigenschaften zur Lösung komplexer Probleme, deren Lösung eine enge Verknüpfung mit vorhandenem Wissen erfordert. In diesem Artikel wird das Paradigma der Antwortmengenprogrammierung mit den Problemstellungen der Intralogistik in einer Potenzialanalyse abgeglichen.

[Schlüsselwörter: Antwortmengenprogrammierung, logische Programmierung, Intralogistik, Materialflussplanung, zellulare Transportsysteme]

Inhouse logistics as a discipline has become a major field of research due to a changing global market. Consequently, inhouse logistics offers problem tasks whose solution has yet to be tackled appropriately. Answer set programming is known to be highly suitable for the computation of complex problems which require close connection to expert knowledge. This article aims at providing a comparison of the demands of intralogistics problems and the advantages answer set programming offers.

[Keywords: Answer set programming, logic programming, inhouse logistic, warehouse planning, cellular transport systems]

1 EINLEITUNG/MOTIVATION

Antwortmengenprogrammierung kombiniert intuitive Regelbasierung mit verifizierbarer Logik unter Berücksichtigung von Unvollständigkeit und Unsicherheit von Wissen. Als Konzept der deklarativen Programmierung bietet Antwortmengenprogrammierung zudem hervorragende Eigenschaften zur Lösung komplexer Probleme, deren Problemlösung eine enge Verknüpfung mit Expertenwissen erfordert [Geb13]. Die automatisierte Entscheidungsfindung respektive -unterstützung intralogistischer Probleme ist seit einiger Zeit Gegenstand intensiver Forschung, insbesondere in Gegenwart des deutlich gesteigerten Stellenwerts der Logistik in Wissenschaft und Forschung. Der vorgestellte Artikel stellt einen ausführlichen Potenzialabgleich vorherrschender Problemstellungen in der Logistik und den Lösungsansätzen, die Antwortmengenprogrammierung bietet, dar.

Antwortmengenprogrammierung ist eine Methode aus dem Gebiet der logischen Programmierung, basierend auf stabilen Modellen nach der Definition von Gelfond und Lifschitz [GL88, GL91]. Die Antwortmengensemantik erweitert die Semantik stabiler Modelle um eine weiterführende Art der Negation [CDP02]. Erster Kernpunkt der Antwortmengenprogrammierung aus Anwendersicht ist die Möglichkeit, das zugrundeliegende Problem direkt zu beschreiben, anstatt wie bisher praktiziert das Vorgehen der Lösungsfindung durch ein Programm zu definieren [Geb13]. Somit wird der Lösungsvorgang an die künstliche Intelligenz übergeben, während der menschliche Entscheider in der Lage ist, sich auf das Problem zu fokussieren. Zweiter Kernpunkt ist die differenzierte Modellierbarkeit von Unsicherheit, fehlendem Wissen und definitivem Nichtwissen durch zwei verschiedene Negationsoperatoren. Antwortmengenprogrammierung findet bereits vielfältige Anwendung in unterschiedlichen Bereichen. Nur einige Beispiele sind automatische Komposition von Musik [BBV+11], Konfiguration von industriellen Produkten [SN98], ein Entscheidungsunterstützungssystem in einem Spaceshuttle der NASA [NBG+01] oder die automatisierte Generierung von Arbeitsplänen unter Berücksichtigung von Teamzusammensetzungen eines Frachthafens [GIL+10].

DOI: 10.2195/lj_Proc_schieweck_de_201605_01
URN: urn:nbn:de:0009-14-43599

Logistische Systeme können in hohem Maße von den Eigenschaften der Antwortmengenprogrammierung profitieren. Besonders die Problemstellungen, die die Automatisierung und Unterstützung von Materialfluss- und Lagerplanung aufwerfen, weisen erhebliches Potenzial zur Anwendung der Antwortmengenprogrammierung auf. Bis dato wurden hier vor allem quantitative Ansätze, wie sie vielfach in der englischsprachigen Literatur zu finden sind, verfolgt. Die Entwicklung der Echtzeitfähigkeit von Solvern für Antwortmengenprogramme eröffnet zudem im Angesicht der in der Industrie 4.0 aufkommenden hochflexiblen und wandelbaren Fördersysteme weitreichende Möglichkeiten.

Der weitere Verlauf des Artikels gliedert sich wie folgt. Zunächst wird eine anwenderbezogene Einführung in die Thematik der Antwortmengenprogrammierung gegeben. Anschließend werden einige Problemstellungen der Intralogistik beispielhaft dargestellt. Auf Basis dieser Erkenntnisse wird abschließend ein Abgleich von Potenzialen der Antwortmengenprogrammierung und Anforderungen logistischer Problemstellungen vorgenommen.

2 ANTWORTMENGENPROGRAMMIERUNG

Antwortmengenprogrammierung (ASP, aus dem engl. Begriff *Answer Set Programming*) ist als neues Paradigma der logischen Programmierung in den späten 1990er Jahren aufgekommen [AKL+05]. Die notwendigen Grundlagen der Semantik stabiler Modelle werden in [Nie99] und [MT99] publiziert. Erstmalig erwähnt wird der Begriff „Answer Set Programming“ in [Lif02]. ASP ist aus den Bereichen des nichtmonotonen Schließens und logischen Programmierens entstanden [GLN+07].

Nichtmonotones Schließen grenzt sich vom klassischen, monotonen Schließen wie folgt ab. Nach den Grundsätzen des monotonen Schließens ist, wenn ein Satz *A* eine logische Konsequenz einer Menge von Sätzen *T* ist, *A* auch immer eine Konsequenz einer beliebigen Obermenge von *T*. Mit anderen Worten wird eine gültige Schlussfolgerung niemals durch zusätzliche Information revidiert. Menschliche Entscheidungen laufen jedoch nicht nach diesem Muster ab, begründet liegt dies in einer zumeist unvollständigen Informationsbasis. Oftmals finden menschliche Entscheidungen aufgrund von Annahmen statt, die wir als *normal* ansehen. Es ist jedoch möglich, dass sich eine solche Annahme als falsch erweist und durch zusätzliche Information über die *abnormale* Situation ein Widerruf der vorherigen Schlussfolgerung notwendig wird [BNT07]. Beispielsweise wird ein Lagerplaner in einer normalen Situation unter der Annahme operieren, genügend arbeitsfähiges Personal zur Abwicklung des Betriebs akquirieren zu können. Durch die zusätzliche Information, dass sich das Lager inmitten des brasilianischen Urwalds befindet, muss diese Annahme revidiert werden. Die zusätzliche Information über den ungewöhnlichen Standort führt zu dieser Revision. Derartige Entscheidungsmechanismen werden als nichtmonoton klassifiziert [BNT07]. ASP verfügt über die Fähigkeit, nichtmonotones Schließen abzubilden und wird gemeinhin sogar als dessen rechentechnische Verkörperung verstanden [AKL+05].

Ein logisches Programm besteht aus einer Menge von Axiomen (Aussagen), aus denen neue Aussagen abgeleitet werden. Darin unterscheidet sich logisches Programmieren essentiell von der (klassischen) imperativen Programmierung, in der ein Programm aus einer Folge von Anweisungen besteht. Ein Kennzeichen eines deklarativen (logischen) Programms ist die Unabhängigkeit der Lösung von der Reihenfolge der Axiome [BET11]. Die Idee des logischen Programmierens ist aus dem Wunsch entstanden, logische Schlussweisen zu formalisieren. Dieser geht bereits auf die Arbeiten von Aristoteles († 322 v. Chr.) zurück [Cla11]. Als Durchbruch auf diesem Gebiet gilt die Entwicklung des Resolutionskalküls von Robinson im Jahre 1965 [R65]. Colmerauer et al. schaffen mit der Erfindung von PROLOG [CKP+73] die Verbindung von Logik und Programmieren [Sch03]. PROLOG wird fälschlicherweise häufig mit logischem Programmieren gleichgesetzt, grenzt sich von der Antwortmengenprogrammierung aber beispielsweise durch fehlende vollständige Deklarativität und fehlende Möglichkeit der Modellierung starker Negation ab.

2.1 GRUNDLEGENDE DEFINITIONEN

Ein Antwortmengenprogramm P besteht aus einer Menge von Regeln der Form

$$r: \quad H \leftarrow A_1, \dots, A_n, not\ B_1, \dots, not\ B_m. \qquad (2)$$

worin $0 \leq m,n$ gilt und H, A_i und B_j als *Literale* bezeichnet werden. Literale sind elementare Aussagen, die mit den Werten wahr oder falsch belegt sein können. Informell drückt Regel (1) aus, dass H wahr ist, solange $A_1,\dots,A_n$ wahr und $B_1,\dots,B_m$ falsch sind oder kein Wissen über $B_1,\dots,B_m$ besteht. H heißt der *Kopf* von r und $A_1,\dots,A_n$, *not* $B_1,\dots,B_n$ bilden den *Rumpf* der Regel r. Die Mengen

$$pos(r) = \{A_1, \dots, A_n\} \text{ und } neg(r) = \{B_1, \dots, B_m\} \qquad (3)$$

werden als positive und negative Rumpfliterale der Regel r bezeichnet. Ist der Rumpf einer Regel leer, wird sie *Fakt* genannt und drückt aus, dass der Kopf der Regel unabhängig von anderen Literalen mit gegebenem Wert bekannt ist [BK14].

$$a \leftarrow . \text{ oder } a. \qquad (4)$$

Als Beispiel dieser einfachsten Form einer Regel kann ausgedrückt werden, dass ein Kommissionierer X derzeit im Lager anwesend ist.

$$anwesend(X). \qquad (1)$$

DOI: 10.2195/lj_Proc_schieweck_de_201605_01
URN: urn:nbn:de:0009-14-43599

Wir nehmen an, dass ein im Lager anwesender Kommissionierer arbeitet, sofern wir keine Kenntnis darüber haben, dass er sich in einer Pause befindet.

$$arbeitet(X) \leftarrow anwesend(X), not\ inPause(X). \quad (5)$$

An dieser Stelle wird deutlich, dass *not* kein gewöhnlicher Negationsoperator ist, sondern vielmehr die Aussage „nicht ableitbar" verkörpert [BET11]. Da sich ein Kommissionierer nur in einem geringen Teil seiner Anwesenheit in einer Pause befindet, nehmen wir bei einem Programm P_1, bestehend aus den Regeln (4) und (5) an, dass dies ohne weitere Spezifizierung nicht der Fall ist und wir *arbeitet* ableiten können. Durch Hinzunahme des Fakts (z.B. nach einer entsprechenden Beobachtung)

$$inPause(X). \quad (6)$$

in ein Programm $P_2 = P_1 \cup \{(6)\}$ für den Kommissionierer *X* muss diese Aussage revidiert werden. An dieser Stelle wird der nichtmonotone Charakter von ASP deutlich. Zusätzlich zu *not* kann in der Antwortmengensemantik der starke Negationsoperator ¬ verwendet werden. Somit ist es möglich, zwischen dem Fall „es gibt keinen Nachweis für dieses Atom" (*not a*) und „es gibt einen Nachweis, dass das Atom nicht wahr ist" (¬*a*) zu unterscheiden. Es kann jedoch gezeigt werden, dass der starke Negationsoperator durch Verwendung eines zusätzlichen Atoms $\overline{a}$ in der Modellierung obsolet wird [GL91]. Es handelt sich demnach um einen Komfort der Syntax, der die Ausdrucksstärke der Sprache nicht vergrößert.

2.2 ANTWORTMENGEN

Die Bildung einer Antwortmenge entspricht, informell ausgedrückt, der Suche nach einer Menge von Literalen, die das vorliegende Programm erfüllt. Eine Antwortmenge ist also die Lösung des Programms, das ein (reales) Problem modelliert. Ein Programm kann keine, eine oder mehrere Antwortmengen haben. Zur besseren Verständlichkeit folgen drei Beispiele, die die Suche nach einer Antwortmenge unter verschiedenen Voraussetzungen illustrieren. Anschließend werden die notwendigen formalen Definitionen entwickelt. Als erstes Beispiel soll das folgende Programm P_3 dienen:

$$inBearbeitung(A) \leftarrow freigegeben(A), verfügbar(A). \quad (7)$$
$$verfügbar(A) \leftarrow \neg reserviert(A). \quad (8)$$
$$freigegeben(A) \leftarrow bezahlt(A). \quad (9)$$
$$bezahlt(A). \quad (10)$$

Ein Kommissionierauftrag *A* wird entsprechend seines Status in die Bearbeitung gegeben oder auch nicht. Der Auftrag ist nach Fakt (10) bezahlt. Mit diesem Wissen folgt aus Regel (9), dass der Auftrag verwaltungsseitig freigegeben ist. Für Regel (8) besteht kein weiteres Wissen, Regel (7) kann aufgrund der Nichtableitbarkeit von *verfügbar* nicht schließen. Der geschlossene Zustand *{bezahlt, freigegeben}* ist der einzige, den das Programm erfüllt. Ein Zustand ist eine Menge von Literalen.

Die Suche nach einem solchen Zustand gewinnt an Komplexität, sobald Default-Negation im Programm auftritt [BET11]. Das Programm P_4 besteht aus den folgenden Regeln:

$$\neg reserviert(A) \leftarrow not\ reserviert(A). \quad (11)$$
$$reserviert(A) \leftarrow not\ \neg reserviert(A). \quad (12)$$

Während die Suche nach einer P_3 erfüllenden Antwortmenge intuitiv gestartet und durchgeführt werden konnte, findet sich für P_4 ohne vereinfachende Annahmen kein Startpunkt. Es wird angenommen, dass *reserviert* nicht ableitbar ist. Folglich kann ¬*reserviert* abgeleitet werden. Unter dieser Annahme ist *{¬reserviert}* ein geschlossener Zustand. Andererseits kann ebenso angenommen werden, dass ¬*reserviert* nicht abgeleitet werden kann, in diesem Fall ergibt sich *{reserviert}* als geschlossener Zustand.

Um die Charakteristik von ASP darzustellen betrachten wir erneut P_3. In seiner aktuellen Form muss aufgrund von Regel (8) für jeden Auftrag gespeichert und abgefragt werden, ob die lagerhaltige Menge bereits durch andere Aufträge reserviert ist. Wir gehen an dieser Stelle jedoch davon aus, dass es sich hierbei um einen Ausnahmefall handelt und normalerweise genügend Artikel vorrätig sind. Wird Regel (8) durch die folgende Regel

$$verfügbar(A) \leftarrow not\ reserviert(A). \quad (13)$$

ersetzt, muss lediglich eine Liste vorgehalten werden für den Fall, dass die Restmenge eines Artikels reserviert ist. Die Regel muss nur für diese (seltenen) Instanzen erzeugt und abgefragt werden. Hieraus ergibt sich eine deutliche Verkleinerung des Programms und somit Beschleunigung der Lösungsgenerierung.

Tritt keine Default-Negation in einem Antwortmengenprogramm auf, sind keine Annahmen zu deren Lösung notwendig und es kann mit einem vergleichsweise intuitiven Bottom-Up-Ansatz ein geschlossener Zustand gefunden werden. Ein solcher Zustand wird als Antwortmenge bezeichnet [BET11]. Zur Darstellung des Suchmechanismus nach Antwortmengen muss im Allgemeinen das Redukt P^S eines Programms *P* für den Zustand *S* definiert werden:

$$\mathrm{P}^S := \{H \leftarrow A_1, \dots, A_n. \mid H \leftarrow A_1, \dots, A_n, not\ B_1, \dots, not\ B_m. \in \mathrm{P}, \{B_1, \dots, B_m\} \cap S = \emptyset\} \quad (14)$$

Dieses Redukt wird nach den Begründern des Antwortmengenparadigmas auch Gelfond-Lifschitz-Reduktion genannt [GL88]. Anders ausgedrückt wird das Redukt in zwei Schritten gebildet [BK14]:

DOI: 10.2195/lj_Proc_schieweck_de_201605_01
URN: urn:nbn:de:0009-14-43599

1. Alle Regeln, deren Rumpf ein *not B* mit $B \in S$ enthalten, werden entfernt. Da *B* in *S* wahr ist, kann keine dieser Regeln mehr schließen.
2. In den verbleibenden Regeln werden alle negativen Rumpfliterale *C* (mit den zugehörigen *not*-Ausdrücken) entfernt. Da $C \notin S$ ist, kann jedes *not C* als wahr angenommen werden.

Die Reduktion erfolgt für alle Zustände *S*, von denen angenommen wird, dass es sich bei ihnen um eine Antwortmenge handelt. Die Bildung dieser sogenannten Kandidatenmenge ist Aufgabe und wesentliches Unterscheidungsmerkmal verschiedener ASP-Solver. Eine kleine Kandidatenmenge, die alle relevanten Zustände *S* beinhaltet, führt zu einem effizienten und schnellen Lösungsprozess. Das gewonnene Redukt kann durch einen einfachen Bottom-Up-Ansatz aufgelöst werden. Entspricht die gewonnene Antwortmenge des Reduktes P^S dem Zustand *S*, heißt *S* Antwortmenge von *P*.

2.3 Erweiterungen der Sprache

ASP wird durch verschiedene Regeltypen erweitert, die das Potenzial der Beschreibung und Modellierung erweitern. Folgend werden Constraints, Auswahlregeln (*single choice rules*) und Disjunktionen (*disjunctions*) erläutert.

Constraints können genutzt werden, um auszuschließen, dass definierte unerwünschte Literalkombinationen als Teil einer Lösung, d.h. als Antwortmenge eines Programms ausgegeben werden.

$$\leftarrow B. \tag{15}$$

Der Einsatz einer solchen Regel in einem Programm *P* bewirkt, dass *M* nur eine Antwortmenge von *P* ist, wenn die Vereinigung der Literale von *B* nicht in *M* vorkommt. Anders ausgedrückt werden die Zustände, die *B* enthalten, aus der Menge der Antwortmengen eliminiert [EIK09]. Beispielsweise schließt die folgende zusätzliche Regel zu Programm P_3 aus, dass ein Auftrag gleichzeitig reserviert und in Bearbeitung ist.

$$\leftarrow inBearbeitung(A), reserviert(A). \tag{16}$$

Notwendig kann diese Constraint werden, wenn durch zusätzliche Regeln *reserviert* abgeleitet werden kann.

Ein ähnliches Vorgehen wird bei Auswahlregeln gewählt. Diese dienen der Modellierung des Falls, in dem exakt ein Atom aus einer Auswahl von Ausprägungen wahr sein soll [BET11]. Ausgeschrieben wird dies zu

$$a \leftarrow B, not\ \bar{a}. \tag{17}$$
$$\bar{a} \leftarrow B, not\ a. \tag{18}$$

und vereinfacht zu

$$\{a\} \leftarrow B. \tag{19}$$

Ist *B* erfüllt, muss also entweder *a* oder $\bar{a}$ erfüllt sein. Zudem ist es möglich, Disjunktionen im Kopf von Regeln zu verwenden. Die folgende Regel drückt zum einen Unwissen darüber aus, ob ein Ladetor *L* geöffnet oder geschlossen ist. Zum anderen schließt sie aber auch aus, dass beide Fälle zugleich zutreffen oder keiner der Fälle zutrifft, sofern das Ladetor *L* existiert [EIK09].

$$offen(L) \vee geschlossen(L) \leftarrow ladetor(L). \tag{20}$$

Weitere Erweiterung der Sprache von ASP umfassen Anweisungen zur Optimierung sowie die Darstellbarkeit von Aggregaten und Kardinalitätsbeschränkungen [GKK+15]. Ersteres ist notwendig, um nicht nur die Frage „gibt es eine Antwortmenge zu dem gegebenen Problem“ zu beantworten, sondern die Möglichkeit zu haben, aus einer Menge von gültigen Antwortmengen unter Angabe einer Zielfunktion die optimale auszuwählen. Die Darstellung von Aggregaten dient der Vereinfachung der Programmierung, wie etwa durch die Anweisungen *max* oder *min*. Kardinalitätsbeschränkungen erweitern Auswahlregeln, indem sie anstatt der Auswahl von genau einem Literal einen Wertebereich festlegen.

2.4 Werkzeuge der Antwortmengenprogrammierung

ASP wird durch die Möglichkeit der Entwicklung hocheffizienter Programme zur Findung von Antwortmengen interessant. Die rechnerbasierte Suche nach Antwortmengen wird häufig in zwei Schritten vorgenommen, dem *grounding* und dem *solving*. Im ersten Schritt, dem grounding, wird das vorliegende nutzergenerierte Programm durch ein äquivalentes Programm *grnd(P)* ersetzt, in dem alle verwendeten Variablen durch deren Instanzen „ausgefüllt“ werden. Heutige Grounder zielen darauf ab, möglichst kleine und effizient lösbare Programme zu erzeugen [BET11].

Tabelle 1. *ASP Grounder und Solver*

Grounder	Solver
Lparse	ASSAT
DLV	clasp
gringo	CMODELS
	DLV
	GnT
	LP2
	SMODELS
	XASP

i.A. [BET11]

Im zweiten Schritt erfolgt die Suche nach Antwortmengen auf Basis des im ersten Schritt erzeugten Programms durch entsprechende Solver. Unterschiedliche Solver nutzen verschiedene Herangehensweisen, um zu den Antwortmengen

DOI: 10.2195/lj_Proc_schieweck_de_201605_01
URN: urn:nbn:de:0009-14-43599

zu gelangen. In regelmäßigen Abständen findet ein Benchmark verschiedener Solver statt, in welchem verschiedene Aufgabenstellungen gelöst werden müssen [FMM+14]. Aufgrund vielfältiger Erfolge in diesem Bereich [Pot15] sowie kontinuierlicher Weiterentwicklung ist clasp häufig Mittel der Wahl. Tabelle 1 listet einige gebräuchliche Grounder und Solver auf. Es sei angemerkt, dass keine durchgehende Kompatibilität zwischen Groundern und Solvern besteht. Originär bilden Lparse und SMODELS [IPT00] sowie gringo und clasp [GKK+11] Paare.

2.5 KOMPLEXITÄT

Die Einordnung von Antwortmengenproblemen in bekannte Problemklassen erfordert ein gewisses Grundverständnis dieser, welches an dieser Stelle nicht als vorausgesetzt angenommen werden kann. Aus diesem Grund wird folgend ein Exkurs gegeben, der die grundlegenden Problemklassen einführt. Im anschließenden Kapitel werden Antwortmengenprogramme in die Problemklassen eingeordnet.

2.5.1 EXKURS: KOMPLEXITÄTSKLASSEN

Die Klassifikation von Problemen nach algorithmischer Schwierigkeit wird unter dem Oberbegriff der Komplexitätstheorie zusammengefasst [Hro11]. Sie ist eine Fortsetzung der Theorie der Berechenbarkeit, die sich damit befasst, ob überhaupt ein Algorithmus zur Lösung eines Problems existiert. Die Komplexitätstheorie vertieft diese Thematik, indem sie eine feinere Klassifizierung danach vornimmt, mit welcher Effizienz ein Problem gelöst werden kann. In der Praxis gibt es durchaus Problemstellungen, zu deren Lösung ein Algorithmus existiert, dieser jedoch eine unannehmbare Laufzeit ab einer bestimmten Größe von Probleminstanzen aufweist. Aus diesem Grund wird häufig auf Heuristiken zurückgegriffen, die vertretbar gute Lösung liefern ohne den gesamten Lösungsraum zu untersuchen. Diese werden an dieser Stelle allerdings nicht weiter behandelt.

Die Ermittlung des Aufwands einer Berechnung kann hinsichtlich verschiedener Ressourcen erfolgen, z.B. Anzahl Prozessoren, Energie, Speicherplatz und Laufzeit. Hier soll der Fokus auf letztgenanntem liegen. Die sogenannte Zeitkomplexität entspricht der Anzahl elementarer Operationen einer Berechnung [Hro11]. Sie steht in direktem Zusammenhang mit dem Energieverbrauch. Grundsätzlich wird ein Problem nach seinem Worst-Case-Aufwand klassifiziert, also der oberen Schranke aller möglichen Probleminstanzen. Es soll eine Laufzeitgarantie ausgewiesen werden. Es muss darauf hingewiesen werden, dass die Klassifikation eines Problems, nicht eines Algorithmus durchgeführt wird. Zur Klassifikation eines Problems muss demnach zunächst nachgewiesen werden, dass ein Algorithmus zur Lösung des Problems optimal ist.

Probleme werden allgemein als effizient lösbar klassifiziert, wenn sie in Abhängigkeit der Größe der Probleminstanz einen polynomiellen Aufwand verursachen. Die hierdurch beschriebene Komplexitätsklasse wird als *P* bezeichnet. Die Problemklasse *NP* umfasst die Probleme, die mit exponentiellem (d.h. höheren) Aufwand gelöst werden können. An dieser Stelle ist der Hinweis angebracht, dass sich der Ausdruck „effizient lösbar“ innerhalb praktischer Grenzen bewegt. Bei entsprechend großer Eingabe wird auch ein P-schweres Problem zu großer Laufzeit führen und dennoch weiter der Klasse P zugehören [AB10]. Ein Problem wird als *NP-vollständig* definiert, wenn es in NP, nicht aber in P enthalten ist [HMU06]. Dass dies für eine Anzahl von Problemen gilt wird angenommen, der Beweis ist jedoch eines der großen ungelösten Probleme der Informatik.

Als Beispiel für ein Problem der Klasse P soll das Finden des kürzesten Weges für ein autonomes Fahrzeug dienen. Autonome Fahrzeuge bewegen sich auf explizit oder implizit angelegten Graphen, die durch die zu erreichenden Knoten und die dazwischenliegenden Pfade bestimmt werden. Erhält ein Fahrzeug einen Fahrauftrag zum Transport eines Behälters, sucht es zunächst den kürzesten Weg zur Quelle des Behälters und anschließend zur Senke des Behälters. Die Laufzeit des für diese Anwendung typischen Dijkstra-Algorithmus beträgt bei n Knoten $O(n^2)$, wächst also höchstens quadratisch in n, und ist somit polynomiell [RR10]. Das Problem wird folgend erweitert, um ein Beispiel aus NP zu illustrieren. Betrachten wir das Beispiel eines Kommissionierers in einem Person-zur-Ware-System, der bestimmte Gassen ablaufen muss, um die gewünschten Artikel zu entnehmen. Die Pfade, auf denen sich der Kommissionierer bewegen muss, lassen sich erneut durch einen Graphen beschreiben. Um den Prozess durch möglichst geringe Wegzeit zu optimieren, soll der Kommissionierer während des Rundgangs keine Gasse zweimal ablaufen. Der Kommissionierer muss am Ende seines Rundgangs wieder an der Basis angelangt sein. Es wird ein möglichst kurzer Pfad gesucht. Das Problem lässt sich analog zu dem eines Hamiltonkreises modellieren. Bei einem Graphen mit m Knoten wächst die Anzahl der möglichen Lösungen (d.h. Pfade) auf $O(m!)$ und übersteigt somit die Grenze des Polynomiellen [HMU06].

Zum weiteren Verständnis des folgenden Kapitels sei noch die NP^{NP}-Komplexität erwähnt. Diese Problemklasse wird entscheidbar in NP für den theoretischen Fall, dass ein NP-Orakel existiert [EGM97]. Ein NP-Orakel erlaubt die „kostenfreie“ Lösung eines NP-Problems in einem Schritt. Dieses theoretische Konstrukt dient vordergründig der Klassifizierung. Es wird jedoch deutlich, dass die Laufzeit gegenüber der Klasse NP weiter steigt.

DOI: 10.2195/lj_Proc_schieweck_de_201605_01
URN: urn:nbn:de:0009-14-43599

2.5.2 KOMPLEXITÄT VON ANTWORTMENGENPROGRAMMEN

Die Komplexität der Entscheidung darüber, ob ein Programm eine bestimmte Antwortmenge hat, ist NP-vollständig [MT99]. Sind eine oder mehrere Disjunktionen Bestandteil eines Programms, steigt die Komplexität auf NP^{NP} [DEG+01]. Prädikatenlogische Programme (d.h. nicht-ground-Programme) weisen eine exponentiell höhere Komplexität auf. ASP-Modellierung ist in der Lage, alle NP-Entscheidungsprobleme so darzustellen, dass die generierten Antwortmengen die Lösung des ursprünglichen Problems darstellen [BET11].

2.6 STÄRKEN DER ANTWORTMENGENPROGRAMMIERUNG

Die Stärken von ASP, die deren Nutzen für Anwender interessant gestalten, setzen sich aus zwei Bereichen zusammen. Zum einen stellen die Methoden der deklarativen Programmierung bereits eine Fülle von Vorteilen gegenüber der imperativen Programmierung bereit. Zum anderen weist ASP als spezielle Methode aus diesem Bereich eine Anzahl von Eigenschaften auf, die sie weiter positiv abgrenzen. Die Stärken von ASP sind vor allem bei der Modellierung von unvollständigem und unsicheren Wissen und der Lösung kombinatorischer Probleme zu finden. Folgend werden die Eigenschaften sowohl der deklarativen Programmierung als auch speziell von ASP beschrieben.

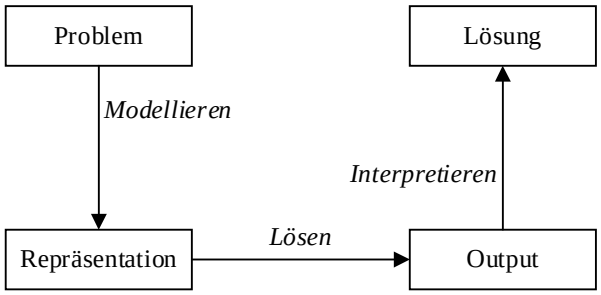

Abbildung 1: Deklarative Problemlösung (i.A. [Geb13])

Kernelement des deklarativen Lösungsansatzes ist die Möglichkeit, das zugrundeliegende Problem direkt zu beschreiben, anstatt wie beim imperativen Programmieren praktiziert das Vorgehen der Lösungsfindung zu definieren [Geb13]. Der Lösungsvorgang wird an die künstliche Intelligenz übergeben, während der menschliche Entscheider in der Lage ist, sich auf das Problem zu fokussieren. Der Ansatz der deklarativen Problemlösung ist in **Fehler! Verweisquelle konnte nicht gefunden werden.** dargestellt. Das Modellieren des Problems dient dessen Formalisierung, um eine geeignete Repräsentation der realen Problemstellung zu generieren. Der Lösungsprozess von der Repräsentation zum Output des Solvers muss vom Anwender nicht definiert oder beschrieben werden. Es muss lediglich eine Interpretation des Outputs zur Gewinnung von Rückschlüssen auf die ursprüngliche Problemstellung vorgenommen werden. Diese ist, je nach selbstgewählter Art der Repräsentation des Modells, relativ einfach durchzuführen (man nehme als Beispiel Modell P_2 aus Abschnitt 2.1 und die Frage „Arbeitet der Kommissionierer?“).

Die Ursprünge von ASP sind in der logischen Programmierung zu finden. Der Fokus der Entwicklung liegt seit jeher auf der Problemlösung aus dem Bereich der Wissensrepräsentation und Schlussfolgerung [Geb13]. Häufig wird die Verwendung von PROLOG fälschlicherweise mit logischer Programmierung gleichgesetzt. Eine Abgrenzung zwischen PROLOG und ASP wird aus diesem Grund wie folgt gegeben. Aus Anwendersicht sind weniger die unterschiedlichen Lösungsmechanismen als vielmehr deren Auswirkung auf die Modellierung von Interesse. PROLOG ist, im Gegensatz zu ASP, nicht vollständig deklarativ [BET11]. Als Beispiel dienen die folgenden beiden PROLOG-Programme:

```
über(X,Y) :- auf(X,Y).                (21)
über(X,Y) :- auf(X,Z),über(Z,Y).      (22)

über(X,Y) :- auf(X,Z),über(Z,Y).      (23)
über(X,Y) :- auf(X,Y).                (24)
```

Wird die Aussage auf(buch,tisch) hinzugefügt und die Anfrage über(buch,tisch) gestellt, bejaht das erste Programm diese Frage. Das zweite Programm jedoch führt zu einer Endlosschleife und generiert kein Ergebnis [Sch03]. Mit anderen Worten kann das Vorgehen der Lösungssuche durch die Reihenfolge der Regeln in PROLOG beeinflusst werden. Diese Kontrolle erhöht die Ausdrucksstärke der Sprache, durch den nicht-deklarativen Charakter aber auch die Komplexität der Modellierung. Es ist in PROLOG außerdem nicht möglich, strikte Negation zu modellieren. Als Vorteil von ASP kann dessen Robustheit gegenüber Reihenfolgeänderungen sowohl innerhalb des Regelrumpfes als auch zwischen unterschiedlichen Regeln herausgestellt werden [BET11]. Zudem wird ASP allgemein als intuitiv und leicht zu erlernen dargestellt, insbesondere ohne weiteres Hintergrundwissen im Bereich der deklarativen Programmierung [DFP09]. Eine zusammenfassende Darstellung dieser und weiterer Stärken von ASP ist folgend stichpunktartig gegeben:

- *Default-Negation:* Zweckmäßige Modellierung von unvollständiger Information und Ausnahmen [AKL+05].
- *Ausdrucksstärke:* Alle Probleme in NP können modelliert werden [BET11].
- *Optimierung:* Selektion von Antwortmengen unter Berücksichtigung von Zielfunktionen [BET11].
- *Werkzeuge:* Der deklarative Aspekt sowie die Verbindung zu Constraint Satisfaction und Aussagenlogik ermöglichen die Entwicklung hocheffizienter Solver [AKL+05].

DOI: 10.2195/lj_Proc_schieweck_de_201605_01
URN: urn:nbn:de:0009-14-43599

- *Modularität:* Trennung von Problembeschreibung und -lösung [Geb13] sowie Wissensbasis und Problembeschreibung [BET11].
- *Flexibilität:* Erlaubt die Verwendung problemspezifischen Wissens und problemspezifischer Sprache [BET11].
- *Verständlichkeit:* Vergleichsweise einfaches und intuitives Erlernen der Modellierung und Sprache [DFP09].
- *Verifikation und Kontrolle:* Vollständige Deklarativität ermöglicht intuitive Struktur und somit einfache Verifikation und Kontrolle des Programms.

3 PROBLEMSTELLUNGEN DER INTRALOGISTIK

Intralogistik hat sich durch gestiegene Anforderungen eines zunehmend globalisierten und beschleunigten Marktes zu einem Kernelement wirtschaftlichen Agierens von erfolgreichen Unternehmen entwickelt. Der relativ junge Begriff der Intralogistik wurde im Rahmen der CeMAT im Jahr 2005 definiert:

„Intralogistik als Branchenname umfasst die Organisation, Durchführung und Optimierung innerbetrieblicher Materialflüsse in Unternehmen der Industrie, des Handels und der öffentlichen Einrichtung mittels technischer Systeme und Dienstleistungen.“ [Arn06]

Die Problemstellungen der Intralogistik sind so vielfältig wie die industrielle Landschaft selbst. Folgend werden diese anhand der Aufgabenbereiche und Ziele der Intralogistik hergeleitet, bevor die Nennung spezifischer Planungsaufgaben erfolgt.

3.1 AUFGABEN DER INTRALOGISTIK

Die Aufgaben der Intralogistik in ihrer Grundform lassen sich aus den Aufgaben der Logistik im Allgemeinen ableiten. Prominente Klassifizierung der Aufgaben der Logistik sind die 5-, 6- oder 7R-Regel: Das richtige Produkt in der richtigen Menge, der richtigen Qualität, am richtigen Ort, zur richtigen Zeit, zu richtigen Kosten für den richtigen Kunden bereitzustellen [BS04, LS04, Kla02]. Diese zunächst simpel klingende Aufgabe erfordert eine Vielzahl von Operationen aufgrund verschiedenartiger Anforderungen, was sich in der Vielfältigkeit intralogistischer Systeme widerspiegelt. Zur Erfüllung der oben genannten Aufgabe ist es notwendig, einen definierten Eingangsstrom von Gütern durch unterschiedliche Transformationen (z.B. Zeit, Zusammenstellung, Ort) in einen definierten Ausgangsstrom zu überführen [McG12]. Konkreter wird die Einteilung der logistischen Aufgaben nach [Ise94]:

- *logistische Kernleistungen*
  - *Lagerhaltung:* Zeitliche und mengenmäßige Überbrückung der Differenz von Ein- und Ausgangsstrom.
  - *Transport:* Überwindung der innerbetrieblichen Distanz.
  - *Auftragsabwicklung:* Datenmäßige Bearbeitung und Kontrolle des Auftrags.
- *Informationsleistungen:* Sammlung, Bereitstellung und Aufbereitung von Daten, die den Zustand des logistischen Systems beschreiben.
- *Zusatzleistungen:* Bspw. Kommissionierung, Verpackung, Palettierung.

Weiterhin können logistische Aufgaben nach ihrem zeitlichen Horizont der strategischen, taktischen oder operativen Ebene zugewiesen werden [LS04]. Dem strategischen Management sind langfristige Planungsaufgaben zugeordnet wie beispielsweise die Layoutplanung oder Aufbau- und Ablauforganisation. Zum taktischen Logistikmanagement gehören für die Auftragsdurchführung notwendige mittelfristige Planungs- und Überwachungsaufgaben. Auf der operativen Materialflussebene sind die unmittelbar notwendigen Handlungen angesiedelt, wie etwa das Umschlagen, Be- und Entladen, Handhaben, Prüfen, Fördern und Lagern.

3.2 ZIELE DER INTRALOGISTIK

Die Durchführung der oben genannten intralogistischen Aufgaben wird unter der Berücksichtigung bestimmter Zielstellungen verfolgt. Diese dienen grundsätzlich der Erreichung des primären unternehmerischen Ziels, der „Maximierung der Rentabilität des betriebsnotwendigen Kapitals“ [Sch11]. Beispielhafte Zielgrößen der Logistik sind wie folgt [Fle08, Kla02, Pfo72, Pfo10, Sch11]:

- *Kostensenkung* sowohl des Logistikprozesses und der Gesamtprozesse als auch der Investitions- und Betriebskosten.
- *Erhöhung der Anpassungsfähigkeit* des Logistiksystems auf Bedarfs- und Umweltveränderungen.
- *Logistische Leistung*, bspw. im Sinne von Servicegrad, Lieferzeit, Lieferzuverlässigkeit, Lieferungsbeschaffenheit, Lieferflexibilität und Durchlaufzeit.
- *Ökologische* Auslegung und Durchführung logistischer Prozesse und Planung.

Das Streben nach Erfüllung der logistischen Ziele ist von Zielkonflikten geprägt. Dabei können sowohl zwischen verschiedenen Zielbereichen als auch innerhalb eines Zielbereichs Konflikte auftreten. Beispielsweise führt hohe logistische Leistung in der Regel zu hohen Kosten und geringer Flexibilität. Andererseits führen geringe Kosten oft zu

DOI: 10.2195/lj_Proc_schieweck_de_201605_01
URN: urn:nbn:de:0009-14-43599

langen Durchlaufzeiten und wiederum geringer Flexibilität [Sch11].

3.3 Planung in der Intralogistik

Der Planungsbegriff wird von der VDI als „gedankliche Vorwegnahme eines angestrebten Ergebnisses einschließlich der zur Erreichung als erforderlich erachteten Handlungsabfolge“ definiert [VDI5200]. Planung ist somit nicht nur eine strategische Aufgabe, sondern auch auf der taktischen und operativen Ebene notwendig (vgl. Abschnitt 3.1). Das folgende Kapitel soll einen Einblick in verschiedene Planungsaufgaben der Intralogistik geben.

Aus dem Bereich der strategischen Aufgaben der Intralogistik wird folgend die Lager- oder Fabrikplanung dargestellt. Diese ist Gegenstand vielfältiger Forschungsarbeiten, da es bis heute nicht gelungen ist, eine Lösung zur umfassenden und vollständigen Entscheidungsunterstützung der Planung bereitzustellen. Die Schwierigkeit der Lagerplanung liegt zunächst in der Eigenschaft, in hohem Maße von der Erfahrung des durchführenden Planers abhängig zu sein [WS14]. Die auf den ersten Blick simpel anmutende Aufgabe der Lagerplanung erlaubt eine immense Anzahl verschiedener Implementierungsarten, die kaum überschaut werden kann. Die Verbindung aus dem hohen Anteil von Expertenwissen und der hohen Anzahl von Ausprägungen und Planungsmethoden führt zu einem zeitintensiven und intransparenten Planungsprozess [YWB+14]. Die einer Lagerplanung zugrundeliegenden Zielstellungen sind sowohl multikriteriell als auch in den meisten Fällen schwer quantifizierbar [WS14]. Weiterhin kann eine qualitative Abschätzung der Güte einer Planungsvariante häufig erst in einem weit fortgeschrittenen Stadium der Planung erfolgen. Die Wichtigkeit eines erfahrenen Planers zeigt sich unter anderem darin, dass dieser bereits in einem frühen Stadium der Planung eine vergleichsweise treffsichere Einschätzung über die Güte einer Planungsvariante geben kann. Diese erfolgt zumeist auf qualitativer Basis. Die Vielfalt der Lagerplanung zeigt sich nicht nur in der hohen Anzahl von Ausprägungen, sondern auch in der großen Bandbreite von unterschiedlichen Aufgaben, die während einer Lagerplanung durchzuführen sind. Dazu gehören die Definition von Anforderungen, Datenanalyse, Prozessplanung, Technologieauswahl und -dimensionierung, Layoutplanung, Personalbedarfsplanung, Erstellung von Lasten- und Pflichtenheften und auch ergonomische Studien.

Die Aufgabe der rechentechnischen Unterstützung der Lagerplanung wird von verschiedenen Ausgangspunkten adressiert. Auf der einen Seite steht eine Vielzahl von analytischen Modellen zur Abschätzung der Leistungsfähigkeit von Subsystemen eines Lagers. Auf der anderen Seite stehen verschiedene Planungsstufenmodelle, die einen groben Rahmen für logistische Planung vorgeben, aufgrund des mannigfaltigen Charakters der Lagerplanung jedoch wenig konkret werden (siehe bspw. [HSN07, Gud12, BC09]). Diese Eigenschaft resultiert auch aus der hohen Kundenindividualität intralogistischer Planung. Bei Betrachtung der oben genannten Aufgaben innerhalb der Lagerplanung gestaltet es sich als besonders schwierig, die zugrundeliegenden Prozesse zu planen und die entsprechenden Technologien auszuwählen. Diese Schritte sind durch eine hohe Anzahl von Interdependenzen, Unsicherheiten und Ausprägungsformen charakterisiert.

Als Beispiel für eine taktische Aufgabe dient die Beschaffungsplanung lagerhaltiger Artikel. Dazu zählt die Entscheidung, zu welchem Zeitpunkt und in welcher Menge ein zu bevorratender Artikel bestellt wird. Diese Entscheidung hat Auswirkungen sowohl auf die Lagerhaltungskosten und die Kapitalbindung wie auch auf die Verfügbarkeit des Artikels bei Nachfrage. Für diese Entscheidung steht eine Anzahl verschiedener analytischer Ansätze bereit, die auch die Automatisierung der Bestellungsauslösung ermöglichen [Goe11]. Entscheidungsvariablen können an dieser Stelle etwa der aktuelle Bestand des Artikels sowie der Durchschnitt und die Standardabweichung der Nachfrage sein.

Die Fülle von operativen Planungsaufgaben in der Intralogistik ist groß. Eine Formalisierung dieser Aufgaben wird besonders in Gegenwart des Trends dezentraler, automatisierter Systeme angestrebt. Eine beispielhafte Aufgabe in einem solchen System ist die Tourenplanung von Fahrzeugen. Die Entscheidung über eine Route muss zu jedem Zeitpunkt getroffen werden, zu dem ein neues Fahrziel angefahren werden soll. Als Entscheidungsgrundlage dienen zumeist die zurückzulegenden Wegstrecken und -zeiten, eine Berücksichtigung der Stauanfälligkeit eines Streckenabschnitts je nach Situation ist zusätzlich durchaus denkbar. Eine Planung unter Berücksichtigung verschiedener Routingobjekte in einem bestimmten Zeithorizont ist demnach sinnvoll. In automatisierten Ware-zur-Person-Systemen kann als weitere operative Planungsaufgabe auch die Zuweisung von Fahraufträgen zu verschiedenen Fahrzeugen und die Zuweisung von Aufträgen zu Kommissionierstationen genannt werden. Beide Entscheidungen können unterschiedlichen Zielstellungen unterliegen. Pragmatisch ist das Streben nach einer geringen Auftragsdurchlaufzeit, aber auch eine gleichmäßige Auslastung der zur Verfügung stehenden Ressourcen ist eine mögliche Zielstellung. Beide Planungsaufgaben sind durch Unsicherheiten der unmittelbaren Zukunft sowie Abhängigkeit der Entscheidungen der anderen im System befindlichen Subsysteme gekennzeichnet.

4 Potenzialanalyse: Antwortmengenprogrammierung und Intralogistik

Der Begriff der Potenzialanalyse wird häufig im Zusammenhang mit Personalplanung und Akquise genutzt. Eine allgemeine Definition, die für den vorliegenden Fall geeignet ist, lautet wie folgt:

DOI: 10.2195/lj_Proc_schieweck_de_201605_01
URN: urn:nbn:de:0009-14-43599

„Potenzialanalyse bezeichnet die strukturierte Untersuchung des Vorhandenseins bestimmter Eigenschaften (Fähigkeiten). Potenzialanalysen liefern strukturierte Informationen zu Fragen nach der Fähigkeit von Mitarbeitern, Ereignissen, Mitteln und Organisationen.“ [Vol14]

Dieser Definition folgend werden in dem folgenden Kapitel zunächst die Eigenschaften definiert, die der Charakter der Problemstellungen der Intralogistik als Nachfrage an ein zur Verfügung stehendes Mittel stellt. Anschließend werden diese mit den Eigenschaften, die ASP zur Verfügung stellt (Angebot), abgeglichen. Grundsätzlich soll evaluiert werden, inwiefern das Mittel ASP zur Anwendung in der Intralogistik tauglich ist. Dies kann in Form von Entscheidungsunterstützungssystemen wie auch automatisierten Entscheidungssystemen erfolgen.

4.1 NACHFRAGE

Wie bereits in Abschnitt 3.3 dargestellt, sind die Problemstellungen der Intralogistik vielfältig in ihrer Ausprägungsform und Komplexität. Gemein haben alle Problemstellungen, dass ihre Lösung die Akzeptanz von Anwendern erfordern und somit als Verbindungsglied zwischen Experten und Anwendern tauglich sein müssen. Im weiteren Verlauf wird der zeitlichen Ebene der Planung anhand der charakterisierten Planungsaufgaben gefolgt. Die Lager- oder Fabrikplanung stellt als strategische Aufgabe einen großen Umfang von nachgefragten Eigenschaften an ein Mittel. Die große Abhängigkeit von Erfahrungswissen erfordert die Möglichkeit der Modellierung von Expertenwissen. Die Vielfältigkeit von Ausprägungsarten logistischer Systeme verlangt die Verfügbarkeit von Werkzeugen, die die effiziente Lösung großer Instanzen komplexer Probleme erlauben. Diese Forderung wird durch die Präsenz einer Vielzahl von Interdependenzen auf und zwischen prozessualer, implementierungs- und dimensionierungstechnischer Ebene verstärkt. Flexibilität, verfügbare Schnittstellen zu einer Vielfalt weiterer Systeme und Modularität werden durch die große Anzahl von verfügbaren und benötigten Planungsmethoden erforderlich. Die hohe Kundenindividualität und Individualität von Planungssituationen setzt ein Mittel voraus, das in der Lage ist, eine Vielzahl unterschiedlicher Eigenschaften und Situationen zu modellieren. Der multikriterielle und qualitative Charakter der Zielstellung von Lagerplanung verlangt nach der Fähigkeit sowohl qualitative und quantitative Randbedingungen und Gewichtungen abbilden zu können, als auch eine Anzahl unterschiedlicher Lösungsalternativen zu generieren.

Auf taktischer Ebene existieren zur Beschaffungsplanung eine Anzahl etablierter mathematischer Modelle. Die Lösung dieser sind für computerbasierter Systeme vergleichsweise einfache Aufgaben, die jedoch in hoher Frequenz gelöst werden müssen. Weiterhin muss ein verwendetes Mittel in der Lage sein, mit einer Vielzahl von Daten umgehen und im optimalen Fall Datenanalysen eigenständig durchführen zu können.

Die Zahl operativer intralogistischer Planungsaufgaben ist groß. Bezugnehmend auf die in Abschnitt 3.3 beschriebenen Problemstellungen sind vergleichsweise einfache Lösungsvorgehen möglich, aber auch komplexe und potenziell zu höherer logistischer Leistung führende Ansätze. Während erstere kaum Anforderungen an verwendete Mittel stellen, erfordern letztgenannte die effiziente Lösung von Problemen der Klasse NP. Es ist davon auszugehen, dass aufgrund der bestehenden Unsicherheit der näheren und ferneren Zukunft und einer Vielzahl von Entscheidungssituationen relativ ähnliche Instanzen derselben Problemstellung mit hoher Frequenz gelöst werden müssen.

4.2 ABGLEICH VON NACHFRAGE UND ANGEBOT

Das Mittel ASP stellt eine Reihe positiver Eigenschaften bereit, die im Folgenden mit der Nachfrage der Intralogistik abgeglichen werden. Berücksichtigt werden außerdem Einschränkungen der Anwendbarkeit von ASP.

Aufgrund der intuitiven Erlernbarkeit von ASP wird vorausgesetzt, dass ASP die notwendige Akzeptanz von Anwendern erfahren kann. Die Möglichkeit, bei der Generierung von ASP-Programmen Regeln mit selbstgewählten Ausdrücken zu verfassen, erlaubt komfortable Repräsentation von Expertenwissen, die Verwendung von fachspezifischem Vokabular und daraus folgend einfache Verifikation der Programme. Dies ist besonders im Bereich der Lagerplanung von Interesse, um dem hohen Stellenwert von Expertenwissen Rechnung zu tragen und die Akzeptanz des Ansatzes weiter zu steigern. Bestehende, potente Werkzeuge zur Suche nach gültigen und/oder optimalen Antwortmengen erlauben die Modellierung komplexer Probleme und deren Lösung in vergleichsweise kurzer Zeit. Besonders die Auswahl von Prozessen und Technologien im Bereich der Lagerplanung ist ein Problem, das hohen Rechenaufwand erfordert. An dieser Stelle ist die Zeitkomponente allerdings vergleichsweise unkritisch. Im Bereich der operativen Planungsaufgaben wird die Zeitkomponente jedoch essentiell. Hier erweitert der Ansatz des *Reactive ASP* die vorherrschenden Lösungsalgorithmen sinnvoll zur hocheffizienten Generierung von Lösungen [GGK+11]. Ziel des Ansatzes ist die Echtzeitfähigkeit durch Verwendung des Wissens vorheriger, ähnlicher Probleminstanzen zur Lösung des akuten Problems. Die Möglichkeit der modularen Modellierung in ASP erlaubt die Generierung komplexer und dennoch übersichtlicher Programme, bei denen Wissensbasis und Problembeschreibung sowie unterschiedliche Wissensbausteine getrennt voneinander abgelegt werden. Dies ist insbesondere beim Aufbau einer großen Wissensbasis, wie zur Lagerplanung, vorteilhaft. Die vollständige Deklarativität von ASP verstärkt diese Eigenschaft zusätzlich. Die Ausdrucksstärke der Sprache erlaubt die Modellierung verschiedenartiger Probleme und

DOI: 10.2195/lj_Proc_schieweck_de_201605_01
URN: urn:nbn:de:0009-14-43599

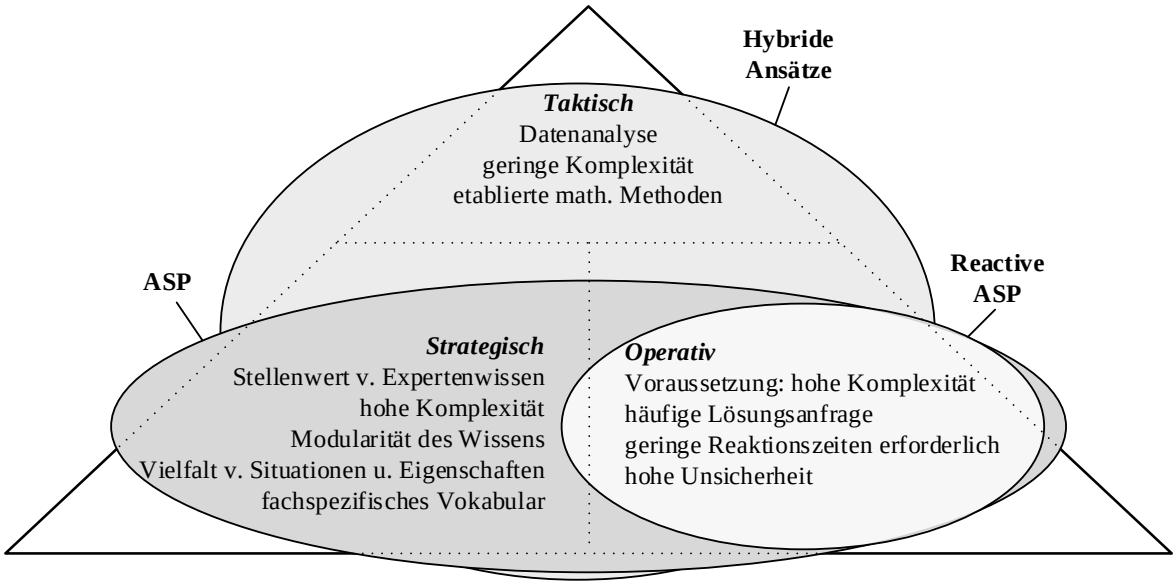

Abbildung 2: Beispielhafte Abgleichstruktur

derer Eigenschaften wie Unsicherheit und Ausnahmeregelungen. Zuletzt ist das primäre Ziel eines Antwortmengenprogramms die Suche nach gültigen Lösungen, weshalb bei den etablierten Solvern auch die Ausgabe aller gültigen Antwortmengen spezifiziert werden kann. Entsprechend ist die Generierung einer Vielzahl von möglichen Lösungsvarianten, auch unter Berücksichtigung eines vorläufigen Rankings der Varianten, möglich. Ein Lagerplaner wird auf diese Art und Weise weiterhin in die Bewertung und Auswahl der Planungsvarianten einbezogen.

Einschränkungen müssen in Hinblick auf die Problemklassen der zu lösenden Aufgaben gemacht werden. ASP-Solver gewinnen im Vergleich zu herkömmlichen Werkzeugen an Effizienz bei Problemen der Klasse NP. Sind die Probleme weniger komplex, sind herkömmliche Werkzeuge vorzuziehen. Das in Abschnitt 3.3 angeführte Beispiel zur Planung auf taktischer Ebene ist demnach nicht sinnvoll durch ASP zu lösen, da es sich um ein relativ homogenes Problem mit vergleichsweise geringer Komplexität handelt. Ebenso sind die einfacheren Herangehensweisen der Fahrauftragsvergabe, Kommissionierstationen-Zuweisung und des Routings eher als ungeeignet einzustufen. Wird die Komplexität durch Vergrößerung des Planungshorizontes und des Optimierungspotenzials jedoch erhöht, ist die Verwendung von ASP in hohem Maße als sinnvoll einzustufen. Das Problem der Lagerplanung ist hochkomplex und in seiner Gesamtheit kaum von einem einzigen Werkzeug abdeckbar. Die Höchstschwierigkeit, die Generierung von Prozess- und Technologievarianten, ist aufgrund der Stärke von ASP bei der Lösung komplexer kombinatorischer Probleme jedoch in hohem Maße für die Anwendung von ASP geeignet. Zur Schaffung umfassender Werkzeuge werden derzeit Ansätze erforscht, die auf eine hybride (sowohl qualitative als auch quantitative) Lösungsweise abzielen (DFG Research Unit 1513). Die Vorteile von ASP können auf diese Art und Weise mit den Stärken erprobter quantitativer Werkzeuge in Verbindung gebracht und genutzt werden. Eine Übersicht der in diesem Artikel behandelten Problemstellungen und deren Abgleich mit ASP ist in Abbildung 2 dargestellt.

5 Zusammenfassung und Ausblick

Der Artikel stellt eine Potenzialanalyse der Anwendung von ASP für Probleme der Intralogistik dar. Zum grundlegenden Verständnis wird ASP zunächst aus einer Anwendersicht dargestellt. Anschließend werden beispielhaft intralogistische Problemstellungen beschrieben. Aufgrund der Fülle von Problemstellungen ist eine allgemeine Aussage anhand aller möglichen Probleme nicht durchführbar. ASP weist besonders zur Lösung von Problemstellungen hoher Komplexität, die zur Lösung eine Verbindung mit Expertenwissen benötigen, Stärken auf. Aufgaben mit vergleichsweise geringer Komplexität wie die Ermittlung des optimalen Bestellungszeitpunktes oder das einfache Routing eines Fahrzeugs scheinen aus diesem Grund nicht geeignet. Wird die Komplexität der Problemstellung jedoch erhöht, in Verbindung mit dem Potenzial einer Leistungssteigerung, können die Vorteile von ASP auch aufgrund hocheffizienter Solver genutzt werden. Die Planung intralogistischer Systeme (Lagerplanung) beinhaltet eine Vielzahl verschiedener Aufgaben. Aufgrund der Flexibilität und Ausdrucksstärke von ASP eignet sie sich besonders, um im Bereich der Generierung von Prozess- und Technologievarianten genutzt zu werden.

In zukünftigen Forschungsarbeiten gilt es, ASP auf ausgewählte intralogistische Fragestellungen anzuwenden, um dessen Tauglichkeit nicht nur zu zeigen, sondern auch für die Domäne der Intralogistik zu nutzen. Dabei sollen

DOI: 10.2195/lj_Proc_schieweck_de_201605_01
URN: urn:nbn:de:0009-14-43599

sowohl Entscheidungsunterstützungssysteme für komplexe Probleme wie bspw. zur Lagerplanung entwickelt werden, als auch Ansätze zur reaktiven, online-fähigen Entscheidung in autonomen Systemen.

FÖRDERHINWEIS

Diese Veröffentlichung entstand im Rahmen der Arbeit des Projektes „Advanced Solving Technology for Dynamic and Reactive Applications“ der Forschergruppe „Hybrid Reasoning for Intelligent Systems“ (FOR 1513), das von der Deutschen Forschungsgemeinschaft (DFG) unter dem Zeichen „KE 1413/8-2“ gefördert wird.

LITERATUR

[Arn06] Arnold, D.: *Einleitung des Herausgebers*. In: Intralogistik. Potentiale, Perspektiven, Prognosen. Berlin/Heidelberg: Springer, 2006, S. 1-4.

[AB10] Arora, S.; Barak, B.: *Computational complexity*. Cambridge: Cambridge University Press, 2010. – ISBN 978-0-521-42426-4

[AKL+05] Anger, C.; Konczak, K.; Linke, T.; Schaub, T.: *A glimpse of answer set programming*. In: Künstliche Intelligenz 19 (1), S. 12-17, 2005.

[BBV+11] Boenn, G.; Brain, M.; de Vos,; Ffitch, J.: A*utomatic music composition using answer set programming*. In: Theory and Practice of Logic Programming 11 (2-3), 2011, S. 397-427.

[BC09] Baker, P.; Canessa, M.: *Warehouse design: A structured approach.* In: European Journal of Operational Research 193 (2), 2009, S. 425-436.

[BET11] Brewka, G.; Eiter, T.; Truszczyński, M.: *Answer set programming at a glance*. In: Communications of the ACM 54 (12), 2011, S. 92.

[BK14] Beierle, C.; Kern-Isberner, G.: *Methoden wissensbasierter Systeme. Grundlagen, Algorithmen, Anwendungen*. Wiesbaden: Springer Vieweg, 2014. – ISBN 978-3-8348-1896-6

[BNT07] Brewka, G.; Niemelä, I.; Truszczyński, M.: *Nonmonotonic Reasoning*. In: Handbook of Knowledge Representation. Elsevier, 2007, S. 239-284

[BS04] Bichler, K.; Schröter, N.: *Praxisorientierte Logistik*. Stuttgart: W. Kohlhammer, 2004. – ISBN 3-173-018147-5

[Cla11] Clausing, A.: *t.Prolog: Logische Programmierung*. In: Programmiersprachen. Heidelberg: Spektrum Akademischer Verlag, 2011. – ISBN 978-3-8274-2850-9, S. 351-423.

[CDP02] Costantini, S.; D'Antona, O.; Provetti, A.: *On the equivalence and range of applicability of graph-based representations of logic programs*. In*:* Information Processing Letters 84 (5), 2002, S. 241-249.

[CKP+73] Colmerauer, A.; Kanoui, H.; Pasero, R.; Roussel, P.: *Un système de communication homme-machine en Francais*. Technical Report. Marseilles, 1973.

[DEG+01] Dantsin, Evgeny; Eiter, Thomas; Gottlob, Georg; Voronkov, Andrei: *Complexity and expressive power of logic programming*. In: ACM Comput. Surv. 33 (3), 2001, S. 374–425.

[DFP09] Dovier, A.; Formisano, A.; Pontelli, E.: *An empirical study of constraint logic programming and answer set programming solutions of combinatorial problems*. In: Journal of Experimental & Theoretical Artificial Intelligence 21 (2), 2009, S. 79-121.

[EGM97] Eiter, T.; Gottlob, G.; Mannila, H.: *Disjunctive datalog*. In: ACM Transactions on Database Systems 22 (3), 1997, S. 364-418.

[EIK09] Eiter, T.; Ianni, G.; Krennwallner, T.: *Answer Set Programming: A Primer*. In: Reasoning Web. Semantic Technologies for Information Systems 5689, Berlin/Heidelberg: Springer, 2009. – ISBN 978-3-642-03753-5, S. 40-110.

[Fle08] Fleischmann, B.: *Grundlagen: Begriffe der Logistik, logistische Systeme und Prozesse.* In: Handbuch Logistik. Berlin/Heidelberg/New York: Springer, 2008, S. 3-12.

DOI: 10.2195/lj_Proc_schieweck_de_201605_01
URN: urn:nbn:de:0009-14-43599

[FMM+14] Francesco C.; Martin G.; Maratea, M.; Ricca, F.: *The Design of the Fifth Answer Set Programming Competition.* Under consideration for publication in Theory and Practice of Logic Programming, 2014.

[Geb13] Gebser, M.: *Answer set solving in practice*. San Rafael, CA: Morgan & Claypool, 2013. – ISBN 978-1-608-45971-1

[GGK+11] Gebser, M.; Grote, T.; Kaminski, R.; Schaub, T.: *Reactive Answer Set Programming*. In: Logic Programming and Nonmonotonic Reasoning. Berlin/Heidelberg: Springer 6645, 2011, S. 54-66.

[GIL+10] Grasso, G.; Iiritano, S.; Leone, N.; Lio, V.; Ricca, F.; Scalise, F.: *An ASP-based System for Team-building in the Gioia-tauro Seaport.* In: Proceedings of the 12th International Conference on Practical Aspects of Declarative Languages. Berlin/Heidelberg: Springer, 2010, S. 40-42.

[GKK+11] Gebser, M.; Kaufmann, B.; Kaminski, R.; Ostrowski, M.; Schaub, T.; Schneider, M.: *Potassco: The Potsdam Answer Set Solving Collection*. In: AI Community 24 (2). Amsterdam: IOS Press, 2011, S. 107-124.

[GKK+15] Gebser, M.; Kaminski, R.; Kaufmann, B.; Lindauer, M.; Ostrowski, M.; Romero, J.; Schaub, T.; Thiele, S.: *Potassco. User Guide*. Second Edition, 2015.

[GL88] Gelfond, M.; Lifschitz, V.: *The stable model semantics for logic programming*. In: The Journal of Symbolic Logic 57 (1), 1988, S. 274-277.

[GL91] Gelfond, M.; Lifschitz, V.: *Classical negation in logic programs and disjunctive databases*. In: New Generation Computing 9 (3-4), 1991, S. 365–385.

[GLN+07] Gebser, M.; Liu, L.; Namasivayam, G.; Neumann, A.; Schaub, T.; Truszczyński, M.: *The First Answer Set Programming System Competition*. In: Logic Programming and Nonmonotonic Reasoning 4483, 2007, S. 3-17.

[Goe11] Goetschalckx, Marc: *Supply Chain Engineering*. Boston, MA: Springer Science+Business Media LLC, 2011.

[Gud12] Gudehus, Timm: *Logistik 1: Grundlagen, Verfahren und Strategien*. Berlin/Heidelberg/New York: Springer, 2012. – ISBN 3-642-29358-0.

[HMU06] Hopcroft, J. E.; Motwani, R.; Ullman, J. D.: *Einführung in die Automatentheorie, formale Sprachen und Komplexitätstheorie*. München: Pearson Studium, 2006. – ISBN 3-8273-7020-5

[Hro11] Hromkovič, J.: *Komplexitätstheorie*. In: Theoretische Informatik. Wiesbaden: Vieweg+Teubner, 2011. – ISBN 978-3-8348-0650-5, S. 206-261.

[HSN07] ten Hompel, M.; Schmidt, T.; Nagel, L.: *Materialflusssysteme: Förder- und Lagertechnik.* Berlin/Heidelberg/New York: Springer, 2007. – ISBN 978-3-540-73235-8

[IPT00] Niemelä; I.; Simons, P.; Syrjänen, T.: *Smodels: A System for Answer Set Programming*. Under consideration for publication in *A System for Answer Set Programming.* [Proceedings of the 8th International Workshop on Non-Monotonic Reasoning, April 2000, Breckenridge, Colorado], 2011.

[Ise94] Isermann, H.: *Logistik: Beschaffung, Produktion, Distribution*. Landsberg/Lech: Verlag Moderne Industrie, 1994. – ISBN 978-3-478-39630-1

[Kla02] Klaus, P.: *Die dritte Bedeutung der Logistik: Beiträge zur Evolution logistischen Denkens*. Hamburg: Deutscher Verkehrs-Verlag, 2002. – ISBN 978-3-871-54273-2

[Lif02] Lifschitz, V.: *Answer set programming and plan generation*. In: Artificial Intelligence 138 (1-2), 2002, S. 39-54.

[LS04] Luczack, H.; Stich, V.: *Industrielle Logistik.* Aachen: Wissenschafts-Verlag Mainz, 2004. – ISBN 3-86073-615-9

[McG12] McGinnis, L. F.: *An object oriented and axiomatic theory of warehouse design*. In: 12th International Material Handling Research Colloquium, 2012, S. 328-346.

DOI: 10.2195/lj_Proc_schieweck_de_201605_01
URN: urn:nbn:de:0009-14-43599

[MT99] Marek, V. W.; Truszczyński, M.: *Stable Models and an Alternative Logic Programming Paradigm*. In: The Logic Programming Paradigm. Berlin/Heidelberg: Springer, 1999, S. 375-398.

[NGB+01] Nogueira, M.; Balduccini, M.; Gelfond, M.; Watson, R.; Barry, M.: *An A-Prolog Decision Support System for the Space Shuttle*. In: Practical Aspects of Declarative Languages 1990. Berlin/Heidelberg: Springer, 2001, S. 169-183.

[Nie99] Niemelä, I.: *Logic programs with stable model semantics as a constraint programming paradigm*. In: Annals of Mathematics and Artificial Intelligence 25 (3/4), 1999, S. 241-273.

[Pfo72] Pfohl, H.-C.: *Marketing-Logistik: Gestaltung, Steuerung und Kontrolle des Warenflusses im modernen Markt.* Mainz: Distribution-Verlag, 1972.

[Pfo10] Pfohl, H.-C.: *Logistiksysteme: Betriebswirtschaftliche Grundlagen*. Berlin/Heidelberg/New York: Springer, 2010. – ISBN 3-642-04162-0

[Pot15] *Potassco Trophy Case*. URL http://potassco.sourceforge.net/trophy.html – Abrufdatum: 06.08.2015

[R65] Robinson, J. A.: *A Machine-Oriented Logic Based on the Resolution Principle*. In: Journal of the ACM 12 (1), 1965, S. 23-41.

[RR10] Roos, M.; Rothe, J.: *Introduction to Computational Complexity.* Supplement in the Mathematical Programming Glossary. 2010.

[Sch03] Schaub, T.: *Antwortmengenprogrammierung*. In: LOG IN: Informatische Bildung und Computer in der Schule 122/123, 2003, S. 104-106.

[Sch11] Schönsleben, P.: *Logistik-, Operations und Supply Chain Management.* In: Integrales Logistikmanagement. Berlin/Heidelberg: Springer, 2011, S. 3-68.

[SN98] Soininen, T.; Niemelä, I.: *Developing a Declarative Rule Language for Applications in Product Configuration*. In: Practical Aspects of Declarative Languages 1551. Berlin/Heidelberg: Springer, 1989, S. 305-319.

[VDI5200] Verein Deutscher Ingenieure e.V.: *VDI 5200 Blatt 1 – Fabrikplanung, Planungsvorgehen.* Berlin: Beuth, 2011.

[Vol14] Volkelt, L: *Neu in der Geschäftsführung: Basic-Tools: Bewerbung*. Wiesbaden: Springer Fachmedien, 2014. – ISBN 978-3-658-04141-0

[WS14] Wunderle, A.; Sommer, T.: *Planung von Intralogistiksystemen: Erfahrung und Augenmaß zählen.* In: Hebezeuge Fördermittel 54, 2014.

[YWB+14] Yousefifar, R.; Wehking, K.-H.; Beyer, T.; Jazdi, N.; Göhner, P.: *Dezentrale selbstorganisierte Grobplanung von Intralogistiksystemen mit Hilfe eines Software-Agentensystems*. In: Logistics Journal: Proceedings. WGTL Kolloquium, München, 2014.

Steffen Schieweck, M.Sc. MS SCE (USA), wissenschaftlicher Mitarbeiter am Lehrstuhl für Förder- und Lagerwesen und dem Lehrstuhl 1 der Fakultät Informatik an der TU Dortmund.

Prof. Dr. Gabriele Kern-Isberner, Professorin und Leiterin der Arbeitsgruppe „Information Engineering“ am Lehrstuhl 1 der Fakultät Informatik an der TU Dortmund.

Prof. Dr. Michael ten Hompel, Inhaber des Lehrstuhls für Förder- und Lagerwesen der TU Dortmund und geschäftsführender Leiter des Fraunhofer-Instituts für Materialfluss und Logistik.

Adresse: Lehrstuhl Informatik 1 – Information Engineering, TU Dortmund, Otto-Hahn-Str. 12, 44227 Dortmund, Germany,
Phone: +49 231 755-2045, Fax: +49 231 755-6555, E-Mail: gabriele.kern-isberner@cs.uni-dortmund.de

Adresse: Lehrstuhl für Förder- und Lagerwesen, TU Dortmund, Joseph-von-Fraunhofer-Str. 2-4, 44227 Dortmund, Germany,
Phone: +49 231 755-2765, Fax: +49 231 755-4768, E-Mail: steffen.schieweck@tu-dortmund.de