Auf dem Weg zum intelligenten Leitsystem

Beitrag in: födern und heben: f+h 49, Nr. 10 (1999), S. LS8-LS11

PROF. DR.-ING. W. A. GÜNTHNER,
DIPL.-ING. F. STEGHERR,

In der automatisierten variantenreichen Großserienfertigung, in der der Mensch nur mehr bedingt als Entscheidungsträger zur Verfügung steht, sind Leitsysteme für einen wirtschaftlichen Betrieb unerläßlich. Systeme, die auf konventionellen Steuerungsstrategien basieren, können heutigen Anforderungen kaum mehr genügen. Flexibilität und vereinfachte Strategieentwicklung müssen im Vordergrund zukünftiger Entwicklungen stehen. Das Konzept des Reinforcement Learnings kann darauf Antworten geben.

Leitsysteme

In den modernen termin-, qualitäts- und kostenorientierten Materialfluß- und Logistiksystemen werden in der dispositiven Steuerungsebene in zunehmenden Maße Leitsysteme eingesetzt. Diese sollen eine optimierte automatisierte Steuerung gewährleisten. So steuern zum Beispiel Fertigungsleitsysteme, die sich in der Hierarchie zwischen PPS- und BDE-Systemen befinden [6], die Fertigung nach den vom PPS-System vorgegebenen Eckdaten weitestgehend autark.

Dem Fertigungsleitsystem fallen somit unterschiedlichste Aufgaben in der fertigungsnahen Auftragssteuerung zu, wobei vor allem die Teilziele der Fertigungssteuerung erfüllt werden müssen. In der variantenreichen Großserienfertigung liegen die Steuergrößen zur Erfüllung dieses Zieles vor allem in der Auftragsgröße und der Sortenreihenfolge [2]. Zusätzlich müssen, ähnlich wie in Materialflußleisystemen, Randbedingungen wie Echtzeitfähigkeit oder Flexibilität gegenüber Um- und Neuplanungen Rechnung getragen werden.

Heuristiken

Heuristiken werden in vielen Fällen für Steuerungsaufgaben in Leitsystemen eingesetzt. Sie stellen ein modellunabhängiges systematisches Suchverfahren dar, das in begrenzter Zeit eine Näherungslösung berechnen kann, die möglichst nahe dem Optimum kommt. Eine Auswahl der in der Literatur für den vorliegenden Einsatz bekannten Heuristiken, bezieht sich zu großen Teilen auf den Kompromiß aus Konvergenzgeschwindigkeit und Güte der Näherungslösung [4]. Hinzu kommt die relativ komplizierte Entwicklung bzw. Anpassung der Heuristiken, die meist in den kurzen Planungszeiträumen kaum Raum finden kann.

Neuronale Netze

Auch Ansätze aus der künstlichen Intelligenz haben Anwendung gefunden. Einige Beispiele [1], die auch schon Einzug in die alltägliche Praxis gefunden haben, zeigen Möglichkeiten auf, wie neuronale Netze deutliche Vorteile gegenüber bisher vorherrschenden Systemen bringen.

Künstliche neuronale Netze, wie zum Beispiel Multi-Layer Perzeptronen, haben ihre originäre Anwendung im Bereich der Mustererkennung. Sie können von einem Eingabe- auf ein Ausgabemuster schließen [5]. Dies kann man sich zunutze machen, um aus einem gegebenen Betriebsszenario notwendige steuernde Eingriffe abzuleiten. Jedoch müssen, um solch ein Netz anlernen zu können, viele dieser Musterpaare, d.h. die optimale Lösung, bzw. ein Teil davon, im Vorfeld bereits bekannt sein.

Hopfield-Netze bieten eine Alternative. Entsprechend codiert können sie die minimale Lösung eines Gleichungssystems ermitteln. Falls die Restriktionen der Auftragsverteilung in einem Materialflußsystem mit Hilfe eines Gleichungssystems abbildbar ist, kann dieser Netztyp dazu verwendet werden, eine optimale Lösung für die Auftragsfolge zu berechnen [1].

Reinforcement Learning

Lernen durch Interaktion - das ist die Grundidee, die hinter Reinforcement Learning (RL) steht. In den letzten Jahren entwickelte sich dieses Forschungsfeld stark interdisziplinär. So sind die AI-Forschung, die Psychologie, die Regelungstechnik, das Operations Research oder die Neurologie an den aktuellen Forschungsvorhaben beteiligt. Auch finden sich Ansätze aus künstlichen neuronalen Netzen, den genetischen Algorithmen oder der dynamischen Programmierung wieder.

RL-Modell

RL basiert auf dem Lernen aus Bewertungen (s. Bild 1). So führt ein sogenannter RL-Agent in seiner Umgebung mit der Menge aller Zustände S eine zunächst willkürliche Aktion a aus der Menge aller möglichen Aktionen A aus. Dadurch bewirkt er eine Zustandsänderung des Systems vom Zustand s nach s'. Für diese Änderung erfährt er eine Bewertung r, die auch zeitlich verzögert erfolgen kann. Daraus kann er sich ein Bild seiner Umgebung generieren und Strategien entwickeln, die es ihm erlauben, ausgehend vom aktuellen Zustand des Systems die Aktionen zu errechnen, die auf lange Sicht seine kumulierte Bewertung maximieren.

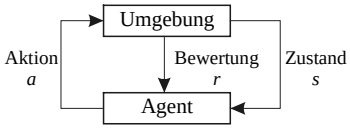

Abbildung 1: Reinforcement Learning

Dem Lernenden muß demnach nicht, wie in den meisten Fällen des Machine Learnings, die Aktion a vorgegeben werden (Supervised Learning), die in einem Zustand s des Systems auszuführen ist.

Die beiden Merkmale, die Trial-and-Error-Suche und die zeitlich verzögerte qualitative Bewertung stellen den größten Unterschied zu anderen maschinellen Lernverfahren dar [7].

RL im Fertigungsleitsystem

Gerade in Fertigungsleitsystemen sind die genannten Eigenschaften des Supervised Learnings von Vorteil. Vor allem die Tatsache, daß eine Bewertung erst zu einem späteren Zeitpunkt und nur qualitativ erfolgen muß, spricht für dessen Einsatz.

Da RL jedoch auf dem Prinzip von Versuch und Irrtum basiert, ist es unmöglich, einen Agenten ohne vorheriges Anlernen in Umgebungen einzusetzen, in denen Entscheidungsfehler zu finanziellen oder materiellen Schäden führen können. Somit ist ein Simulationssystem, das dem Verhalten des realen Materialflußsystems möglichst nahe

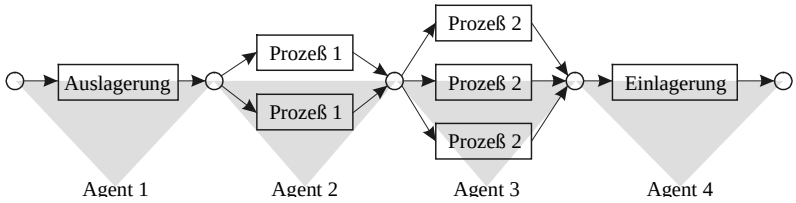

Abbildung 2: Zugeordnete Bereiche verteilter Agenten

kommt, Grundvoraussetzung für den Einsatz. Denn hier können RL-Agenten ohne negative Auswirkungen angelernt werden.

Verteilte, hierarchische Systeme

Materialflußsysteme stellen ein komplexes Netz aus vielen Entscheidungsknoten dar, die miteinander verwoben sind. So wirken sich lokale Entscheidungen auch auf das globale Verhalten aus. Der Einsatz von RL-Agenten muß darauf reagieren können.

Wird nur ein Agent eingesetzt, der das System im ganzen steuert, so kann gewährleistet werden, daß Entscheidungen auf Grundlage aller verfügbaren Informationen getroffen werden. Jedoch ist aufgrund der Fülle an Daten und Entscheidungen, die im Gesamtsystem zu verwalten und zu treffen sind, eine einzige Instanz bei weitem überfordert. Zudem ist für eine Detailentscheidung die Kenntnis des Gesamtsystems im Einzelnen nicht erforderlich.

Durch die Entwicklung eines Multiagenten Systems [3] kann diese Problematik beherrscht werden. So kann jedem Agenten ein genau spezifizierter Bereich zugeordnet werden (s. Bild 2). In diesem sollten alle für eine Entscheidung relevanten Informationen verfügbar sein. Dies bedeutet aber auch, daß der Agent nicht nur den Rüstzustand und den Füllgrad des Puffers seiner zugeordneten Maschine kennt, sondern auch zusätzlich Informationen über den nachfolgenden Bereich erhalten sollte.

Ebenfalls von Vorteil ist die Bildung von Agentenhierarchien. So können auf höheren Ebenen grobe Entscheidungen getroffen werden, die auf unterster Ebene nur mehr Detailentscheidungen erfordern. Mit relativ geringem Aufwand sind daher mächtige Steuerungsstrukturen realisierbar, die der Komplexität in einem Materialflußsystem Rechnung tragen können.

Bewertungsproblematik

Obwohl eine wie auch immer geartete manuelle Ermittlung der optimalen Steuerungsstrategien, wie sie das Supervised Learning erfordert, nicht notwendig ist, stellt sich dennoch die Frage nach der Bewertung des Verhaltens verteilter und hierarchischer Systeme aus RL-Agenten.

Grundsätzlich wird ein RL-Algorithmus über einen skalaren Wert bewertet, der in der Regel im Bereich [-1.0;+1.0] liegt. Je höher der Wert ausfällt, um so besser war die getroffene Entscheidung.

Von großem Nutzen ist hier, daß eine Bewertung nicht sofort, sondern zu einem nahezu beliebigen späteren Zeitpunkt erfolgen kann. So kann man Punkte im System ermitteln, an denen ein Auftrag bewertet werden kann. Hier bietet sich das Ende der Prozeßkette an, an dem die Gesamt- und Einzeldurchlaufzeiten bekannt sind. Diese Zeiten können herangezogen werden, um alle an der Entscheidungskette beteiligten Agenten entsprechend ihres Einflusses bewerten zu können.

Jedoch kann, um eventuell einen Teilbereich zu optimieren, auch nach diesem ein Punkt definiert werden, an dem eine Bewertung auf der Grundlage der bis dahin vorliegenden Daten vorgenommen werden kann.

Generalisierung

Probleme bereiten große Materialflußsysteme unter anderem durch die Menge aller Zustände S, die sie annehmen können. Eine Erforschung des gesamten Raumes ist einerseits unmöglich, andererseits auch nicht zwingend erforderlich. Zum einen liegt dies daran, daß ein Großteil der Zustände zu keiner vernünftigen Lösung führen würde, zum anderen können RL-Agenten durch den Einsatz von Funktionsapproximatoren, wie zum Beispiel Neuronalen Netzen, von unbekannten auf schon einmal gelernte Zustände schließen.

Das heißt, daß ein Agent, der auf einen Zustand trifft, welcher ihm unbekannt ist, mit einer Aktion reagieren kann, die er in einem ähnlichen Zustand gelernt hat und ihm damals als optimal erschienen ist. Dadurch erhält er die Fähigkeit, auf wechselnde Betriebsszenarien reagieren zu können. Eine Modifizierung von Teilen des Systems zieht demnach nicht unbedingt eine neue Lernphase aller Agenten und damit des Fertigungsleitsystems nach sich.

Lernumfang

Wie schon angesprochen, bedeutet die Größe des Zustandsraumes S eine große Herausforderung für ein auf RL-Agenten basierendes Fertigungsleitsystem. So führt dies auch dazu, daß die für das Anlernen notwendige Zeit bei komplexen Systemen einen nicht unerheblichen Umfang annimmt, vor allem dann, wenn komplett modellunabhängiges Lernen ermöglicht werden soll. Das bedeutet, daß sich der Agent nur auf die Informationen, die er durch die Bewertung erfährt stützen kann und kein in irgendeiner Art codiertes Erfahrungswissen zugrunde gelegt werden kann.

Dies hat allerdings den Vorteil, daß eine Implementierung der Agenten erfolgen kann, die auf keinen Spezialfall eines Materialflußsystems fixiert ist, jedoch mit dem Nachteil verbunden ist, längere Lernzeiten in Kauf nehmen zu müssen.

Wird jedoch in die Bewertungsfunktion Erfahrungswissen aufgenommen, so kann dieser Nachteil abgemildert werden. D.h., wenn zum Beispiel die Erfahrung sagt, daß ein Umrüstvorgang in einer bestimmten Situation ungünstig wäre, der Agent jedoch einen Auftrag wählt, der dies erfordert, so kann die Bewertung entsprechend schlecht ausfallen. In Experimenten hat sich gezeigt, daß dieses Vorgehen zu stark verkürzten Lernzeiten führt.

Eine Frage in diesem Zusammenhang ist meist, wann der Lernvorgang abgebrochen werden kann. Dies kann nur sehr schwer pauschal beantwortet werden. Oftmals wird die Schwankung eines Kennwertes herangezogen. So kann als Kriterium die mittlere Durchlaufzeit verwendet werden. Wenn sie nur noch innerhalb eines vorgegebenen Bandes schwingt, so kann der Lernvorgang als abgeschlossen betrachtet werden.

n2-Problem

Ein relativ einfaches Beispiel, das n2-Problem soll zeigen, wie ein System gelernt und optimiert gesteuert werden kann. Das Beispiel ist so gewählt, daß die optimale Strategie konventionell ermittelt werden kann und man anhand dieser die Ergebnisse des RL nachvollziehen und überprüfen kann.

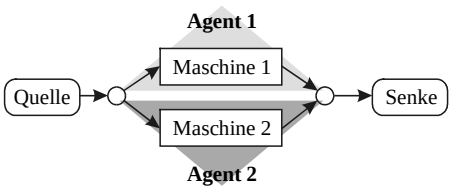

Abbildung 3: Layout des n2-Problems

Das Bild 3 zeigt das System, das aus 2 Maschinen aufgebaut ist, die n Aufträge

abarbeiten müssen. Diese Aufträge lauten auf das Fertigen von drei verschiedenen Produkttypen A, B und C. Beide Maschinen können alle Produkttypen fertigen, jedoch mit unterschiedlichen Prozeß- und Rüstzeiten, wie aus den Tabellen 1 und 2 ersichtlich.

Produkttyp	Maschine 1	Maschine 2
A	20	100
B	50	50
C	100	20

Tabelle 1: Prozeßzeiten

Rüsten von → nach	Maschine 1	Maschine 2
A → B	50	50
A → C	100	20
B → A	20	100
B → C	100	20
C → A	20	100
C → B	50	50

Tabelle 2: Rüstzeiten

Die Quelle erzeugt alle 10 Zeiteinheiten einen Auftrag zur Fertigung eines der drei Produkte und legt diesen in die erste Auftragsliste, die maximal sieben Positionen umfaßt. Ein RL-Agent, dessen Maschine derzeit kein Produkt zu bearbeiten hat, entnimmt der Liste einen Auftrag und gibt ihn seiner Maschine zur Bearbeitung. Danach gelangt dieser zur Senke und wird vernichtet. Ziel der Agenten soll sein, jeden Auftrag in möglichst kurzer Zeit zu bearbeiten.
Ganz offensichtlich ist die beste Strategie, alle Produkte vom Typ A auf Maschine 1 und alle vom Typ C auf Maschine 2 zu bearbeiten. Den Produkttyp B sollten sie zu gleichen Teilen fertigen.
Um die Agenten anlernen zu können sind zwei Dinge zu beachten. Einerseits die Umgebungsrepräsentation, andererseits die Bewertungsfunktion.
In Versuchen wurden einige Umgebungsrepräsentationen entwickelt und deren Auswirkungen auf das Lernverhalten und das Steuerungsergebnis untersucht. Als günstigste hat sich folgende Darstellung herausgestellt.

- Rüstzustand der zugeordneten Maschine
- Anzahl Aufträge für Produkt A
- Anzahl Aufträge für Produkt B
- Anzahl Aufträge für Produkt C

Die Bewertungsfunktion sollte darauf abzielen, daß Aufträge in möglichst kurzer Zeit bearbeitet werden. D.h. ein Auftrag, der in sehr kurzer Zeit durch das System gelaufen ist, sollte eine hohe, ein Auftrag der länger benötigt hat, eine entsprechend niedrigere Bewertung nach sich ziehen. Nachdem die kürzest mögliche Durchlaufzeit 20 Zeiteinheiten beträgt, wird diese mit dem Wert 1.0 bewertet, die maximale Durchlaufzeit von 200 Zeiteinheiten sollte mit -0.9

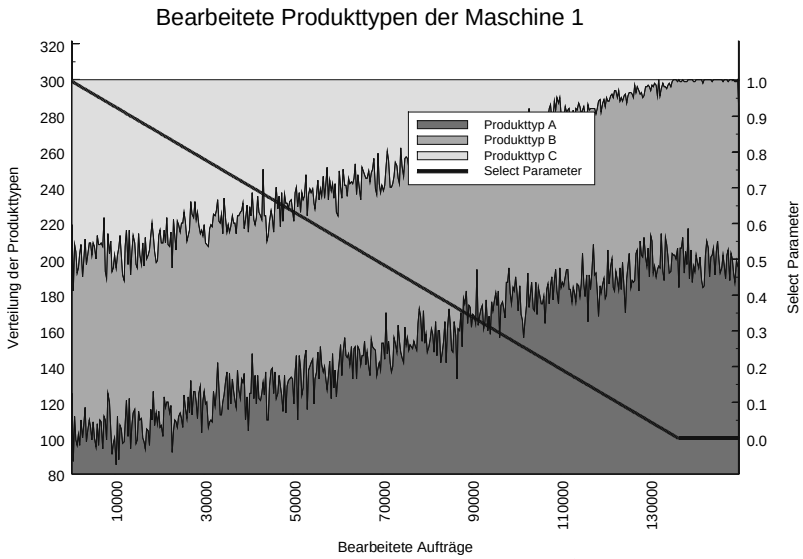

Abbildung 4: n2-Problem Auswertung

bewertet werden. Damit die Rüstzeiten t_R stärkeres Gewicht gegenüber den Prozeßzeiten t_P erfahren, werden diese mit dem Faktor 3 gewichtet. Demnach ergibt sich folgende Bewertungsfunktion:

$$r = -\frac{1}{200}(3t_R + t_P) + 1{,}1$$

Versucht ein Agent aus der Liste einen Auftrag für einen Produkttypen zu entnehmen, der nicht vorhanden ist, so wird dies mit einen Wert von -1.0 bestraft.
Es werden nun 300.000 Aufträge durch das System geschleust und die Agenten nach Fertigstellung eines Auftrages bewertet. Damit diese ihre Umgebung zunächst erkunden und nicht einer vermeintlich optimalen Lösung folgen, kann über den Select-Parameter eingestellt werden, mit welcher Wahrscheinlichkeit sie Aufträge annehmen sollen, die nicht dem aktuell gefundenen Optimum entsprechen. Ein Wert von 1.0 bedeutet, daß in jedem Fall ein zufälliger Auftrag gewählt wird, ein Wert von 0.5 bedeutet, daß dies mit 50% Wahrscheinlichkeit erfolgt. Dieser Parameter wird im Verlauf des Lernens auf nahe 0 gesenkt.
Im Bild 4 sieht man den Lernfortschritt des Agenten, der der Maschine 1 zugeordnet ist. So entnimmt dieser zu Beginn mit gleicher Wahrscheinlichkeit der Liste Aufträge aller Produkttypen. Mit der Zeit lernt er jedoch, daß Aufträge des Produkttyps C sehr ungünstig sind und entnimmt daher auf dessen Kosten verstärkt Aufträge des Typs A an.
Diese Lernaufgabe hat er demnach korrekt erfüllt. Interessant ist jedoch auch, ob die Durchlaufzeiten ebenfalls der herkömmlich ermittelten Steuerungsstrategie entsprechen. So wurde nach der Lernphase eine Liste mit 1000 Aufträgen durch das System geschleust und die Zeiten genommen. Die rüst- und zeitoptimierte Strategie benötigte für die 1000 Aufträge 19160, die beiden Agenten 20730 Zeiteinheiten, was einer geringfügigen Verlängerung um den Faktor 1,08 entspricht.
Demnach kommt das von zwei angelernten RL-Agenten gesteuerte System auf das nahezu identische Systemverhalten, ohne daß das System zuvor manuell detailliert analysiert werden mußte. Bei weit komplexeren System ist damit zu rechnen, daß der Algorithmus gleich gute oder bessere Resultate erzielen kann.

Fazit

Das Konzept des Reinforcement Learnings bietet deutliche Vorteile gegenüber herkömmlichen Ansätzen in Leitsystemen und auch gegenüber bisher realisierten Ansätzen aus der künstlichen Intelligenz.
Die Transparenz eines solchen Systems kann allerdings leiden. Es werden keine Strategien im konventionellen Sinn entwickelt, wodurch keine Kontrolle der Ergebnisse auf dieser Ebene möglich ist. Zudem muß eine eventuell lange offline Lernphase in Kauf genommen werden, vor allem bei schlechter Wahl der Bewertungsfunktion.
Wird der Algorithmus jedoch in ein Assistenzsystem integriert, so können entsprechende Bewertungsfunktionen implementiert werden, die eine gute Konvergenzgeschwindigkeit gewährleisten. Der Planer erhält dadurch ein einfach handzuhabendes, modellunabhängiges System, das ein sehr gutes Steuerverhalten erzielt. Und dies ohne die Notwendigkeit der umfangreichen Analyse des Systems von Seiten des Planers. Zudem zeigen die Agenten nach der Lernphase ein Antwortverhalten und eine Flexibilität, die sie für die echzeitorientierte fertigungsnahe Steuerung prädestiniert.

Literatur

[1] C.H. Dagli, Herausgeber: Artificial Neural Networks for Intelligent Manufacturing. Chapman & Hall, London, 1994.

[2] H. Günther und H. Tempelmeier: Produktion und Logistik. Springer Verlag, 3. Auflage, 1993.

[3] M.L. Littmann: Markov Games as a Framework for Multi-Agent Reinforcement Learning. In: Journal of Artificial Intelligence Research, 5, 1997.

[4] J. Liu und B. MacCarthy: Effective Heuristics for Single Machine Sequencing Problems with Ready Times. In: International Journal of Production Research, 29:1521-1533, 1991.

[5] R.Rojas: Theorie der neuronalen Netze. Springer Verlag, 1996.

[6] G.Spur: Fabrikbetrieb. Hanser Verlag, 1994.

[7] R.S. Sutton und A. Barto: Reinforcement Learning: An Introduction. MIT Press, Cambridge, 1998.