UNIVERSITÄT LEIPZIG

Institut für Produktionswirtschaft
und Industrielle Informationswirtschaft

Arbeitsbericht Nr.2

Anwendung und Beurteilung heuristischer Verbesserungsverfahren für die Maschinenbelegungsplanung

-

Ein exemplarischer Vergleich zwischen Neuronalen Netzen, Simulated Annealing und genetischen Algorithmen

Dipl.-Kfm. Jukka Siedentopf

Leipzig 1994

Inhalt

Zusammenfassung

Der vorliegende Bericht schildert Vorgehensweise und Erfahrungen bei der Modellierung und Implementierung verschiedener heuristischer Verfahren zur Maschinenbelegungsplanung. Bei den Verfahren handelt es sich um ein Neuronales Netz (Hopfield-Netz), ein Verfahren auf Basis des Simulated Annealing und einen genetischen Algorithmus - und somit durchgängig um Verfahren, deren Grundprinzipien sich an natürlichen Adaptionsvorgängen orientieren. Die Darstellungen beinhalten die grundlegende Funktionsweise der gewählten Verfahren, jeweils einen Ansatz für die Modellierung der Maschinenbelegungsplanung sowie einige ausgewählte Aspekte zur (kritischen) Beurteilung der Verfahren.

1 Die "Evolution" von Problemlösungen

Zur Lösung kombinatorischer Optimierungsprobleme werden - auch für betriebswirtschaftliche Aufgabenstellungen - zunehmend Verfahren vorgeschlagen, die als Modelle spezifischer Adaptionsvorgänge in natürlichen Systemen aufgefaßt werden können. Beispiele für natürliche Adaptionsvorgänge und sich daran orientierender Modelle sind das Ausglühen kristalliner Stoffe mit den darauf basierenden Verfahren des Simulated Annealing und der Boltzmann-Maschinen[1)] oder die Evolution biologischer Populationen als Vorbild für Evolutionsprogramme[2)]. Gemeinsames Merkmal der entwickelten Adaptionsanalogien ist neben ihrer Herkunft (Beobachtung natürlicher Phänomene) das Prinzip, sich Problemlösungen anzunähern[3)], indem potentielle Lösungskandidaten manipuliert, die Manipulationen bewertet und schließlich neue Lösungskandidaten auf Basis der Bewertungen ausgewählt werden[4)].

Im folgenden werden Erfahrungen[5)] bei der Entwicklung und Implementierung verschiedener Verfahren zur Feinterminierung[6)] in der Werkstattfertigung (Job Shop Scheduling) dargestellt. Dabei werden insbesondere ein Verfahren auf Basis des Simulated Annealing und ein spezifischer genetischer Algorithmus skizziert. Um die Entwicklung vollständig und transparent wiederzugeben, wird zunächst auch ein Ansatz auf Basis eines Neuronalen Netzwerkes (Hopfield-Netzwerk) vorgestellt, der Ausgangspunkt für die weiteren Entwicklungen war.

---

1) Zu Simulated Annealing und Boltzmann-Maschinen vgl. z.B. Aarts/Korst (1989).

2) Der Terminus ´Evolutionsprogramm´ dient hier in Anlehnung an Michalewicz (1992) als Oberbegriff für alle Verfahren, die sich am Vorbild der natürlichen Evolution orientieren ("We use a common term, Evolution Programs (EP), for all evolution-based systems", Michalewicz (1992), p. 1; dies stellt eine Verallgemeinerung der im Vorwort der Arbeit angegebenen Explikation "Evolution programs can be perceived as a generalization of genetic algorithms" dar, Michalewicz (1992), p. VII).

3) Für einige Verfahren, z.B. für das Simulated Annealing, können Konvergenzbeweise geführt werden. Die Annahmen, die der Beweisführung zugrundeliegen, etwa die Stetigkeit der Temperaturabnahme, sind jedoch für praktische Anwendungen der Verfahren nicht aufrechtzuerhalten. Im folgenden wird daher grundsätzlich ein Optimalitätsverzicht akzeptiert.

4) Obwohl (zumindest temporär) auch Manipulationen akzeptiert werden können, die zu Verschlechterungen führen, werden die untersuchten Verfahren in der vorliegenden Arbeit als *Verbesserungsverfahren* bezeichnet, da die Zielsetzung der Manipulation die Verbesserung (i.S. höherer resultierender Zielfunktionswerte) von bekannten Lösungen ist.

5) Die Ergebnisse wurden unter Mitarbeit des Autors teilweise in einem von der DFG geförderten Forschungsvorhaben zur ´Nutzung und Erweiterung konnektionistischer Verfahren für betriebswirtschaftliche Anwendungen´ erarbeitet, das zwischen 1991 und 1993 an der Westfälischen Wilhelms-

2 Das Anwendungsszenario

2.1 Anforderungen

Besondere Anforderungen an die kurzfristige Produktionsplanung und -steuerung werden bei kundenorientierter Einzel- oder Kleinserienfertigung gestellt[7]. Aus der Notwendigkeit, auf kurzfristige Datenänderungen schnell reagieren zu können, resultierte die Entwicklung interaktiver Planungswerkzeuge in Form elektronischer Fertigungsleitstände[8]. Obwohl *manuelle* Planungstätigkeiten, insbesondere in Form der Feinabstimmung von Maschinenbelegungsplänen[9], ein wesentliches Merkmal dieser Systeme sind, besteht aufgrund der Planungskomplexität[10] weiterhin ein Bedarf an effizienten automatischen Verfahren zur Ermittlung von Basisplänen (Initialisierungsplanung)[11].

Universität Münster unter Leitung von Herrn Prof. Dr. Kurbel am Institut für Wirtschaftsinformatik durchgeführt wurde.

6) Die Begriffe *Feinterminierung* und *Maschinenbelegungsplanung* werden im folgenden synonym benutzt.

7) Zu einer detaillierten Anforderungsanalyse vgl. z.B. Nietsch u.a. (1991).

8) Zu elektronischen Fertigungsleitständen vgl. z.B. Kurbel/Meynert (1988) oder Kurbel (1993), S. 235-278.

9) Für Maschinenbelegungspläne besteht diese Feinabstimmung z.B. darin, daß der Fertigungsleiter eine graphische Darstellung des Plans (in Form eines Gantt-Charts/einer Plantafel) direkt am Monitor eines Arbeitsplatzrechners manipuliert. Dabei werden z.B. Arbeitsgänge mit der Maus erfaßt und durch Verschiebung auf anderen Betriebsmitteln eingelastet. Die Unterstützung seitens des Systems besteht vor allem in der Bereitstellung aller benötigten Informationen zu einzelnen Arbeitsgängen, Fertigungsaufträgen und Betriebsmitteln (z.B. Statusinformationen) sowie in der Durchführung von Konsistenzprüfungen (z.B. wenn durch das Umlasten von Arbeitsgängen Reihenfolgebedingungen verletzt werden oder die Einhaltung vorgegebener Termine gefährdet ist).

10) Die Bestimmung optimaler Lösungen für das Job Shop Scheduling stellt innerhalb der Klasse der NP-schweren Probleme eine ´besonders harte Nuß´ dar, vgl. z.B. Domschke u.a. (1993), S. 363ff.

11) Dies wird auch durch empirische Daten über die *Häufigkeit*, mit der Planungen durchgeführt werden, unterstrichen, vgl. Kurbel (1993), S.238. Die Entwicklung interaktiver Planungswerkzeuge wird oft auch durch die Unzulänglichkeiten vorhandener (konventioneller) Verfahren motiviert. Der Argumentation, die konventionellen Planungswerkzeuge (z.B. die Verfahren der linearen und der gemischt-ganzzahligen Programmierung) seien nicht leistungsfähig genug, wird hier allerdings nicht unumschränkt gefolgt. Zum einen bleiben ´Alternativen´ ihrerseits einen überzeugenden Leistungsnachweis oft schuldig. Zum anderen ignoriert diese pauschalisierende Argumentation die signifikanten Fortschritte, die auch im Bereich der konventionellen mathematischen Programmierung gemacht werden; vgl. als Beispiel zu Performance-Fortschritten im Bereich der mathematischen Programmierung z.B. Suhl/Szymanski (1994), p. 5, oder Suhl (1994), p. 20.

Ein Einsatz spezifischer adaptiver Verfahren kann jedoch auch durch flexibilitätsfördernde strukturelle Merkmale begründet werden: Die strikte Trennung etwa, die viele der Verfahren zwischen Repräsentation und Manipulation von Lösungen einerseits und Bewertung dieser Lösungen andererseits vornehmen, erlaubt es, die Verfahren mit geringem Aufwand an unterschiedliche Zielsetzungen anzupassen. Bei Evolutionsprogrammen eröffnet diese Entkopplung des Suchraums vom Lösungsraum darüber hinaus eine interessante Möglichkeit, Zielsetzungen zu verknüpfen: Storer et al.

Erste Anforderungen an die Konzeption eines Moduls zur automatischen Feinterminierung in einem Fertigungsleitstand resultieren aus dem **Einsatz im interaktiven Betrieb**:

- Die Einplanung soll eine **hohe Effizienz** aufweisen, da die Akzeptanz interaktiver Systeme und die Notwendigkeit, auf kurzfristige Änderungen adäquat reagieren zu können, kurze Antwortzeiten erfordern.
- Eine **hohe Effektivität** des Planungsvorgangs soll durch die Reduzierung (hinsichtlich der Quantität und der Qualität) manuell vorzunehmender Änderungen angezeigt werden.[12)]

Elektronische Fertigungsleitstände stellen i.d.R. Individualsoftware in dem Sinne dar, daß sie an die Bedürfnisse (und die existierenden DV-Umgebungen) des jeweiligen Anwenders spezifisch angepaßt werden (müssen). Der Anpassungsaufwand ist dabei meist beträchtlich[13)]. Rationalisierungspotentiale können durch die Entwicklung modularer Systeme (Baukastensysteme) einerseits und die Verwendung von Softwaremoduln hoher Adaptibilität andererseits erschlossen werden. Bei der Konzeption eines Moduls zur *Feinterminierung* wird die Forderung nach **Individualisierbarkeit** wie folgt spezifiziert:

- Das Modul soll eine hohe Adaptibilität bzgl. der **Planungsziele** besitzen.
- Das Modul soll eine hohe Adaptibilität bzgl. der **Berücksichtigung betriebsspezifischer Nebenbedingungen** besitzen.

Eine Forderung nach *optimalen* Lösungen wird nicht erhoben. Erwünscht sind vielmehr **robuste Planungsergebnisse** in dem Sinn, daß bei - ohnehin erwarteten - Störungen im

---

(1992) skizzieren beispielsweise einen Ansatz zum Job Shop Scheduling, in dem Lösungskandidaten zufällige Abweichungen von gegebenen Operationszeiten kodieren. Auf Basis dieser ´verfälschten´ Zeiten wird anschließend mittels einer Heuristik ein Belegungsplan erstellt und dieser anhand der Zykluszeit bewertet, die jedoch mittels der unverfälschten, originalen Operationszeiten berechnet wird. Als Ergebnis wird eine ´robuste´ Planung durchgeführt, die unter der Zielsetzung der Minimierung der Zykluszeit Lösungen favorisiert, die unter Berücksichtigung von Störungen gute Ergebnisse liefern.

12) Die Aussagekraft dieser ´Anforderungsdefiniton´ ist begrenzt, zumal *Effizienz* und *Effektivität* nicht expliziert werden. Da eine Bestimmung von Effizienz- und Effektivitätskriterien i.d.R. anwendungsspezifisch erfolgt, wird hier eine intuitive Vorstellung von Effizienz und Effektivität vorausgesetzt.

13) Verallgemeinerbare qantifizierte Aussagen über Kosten und Zeitbedarf der Anpassungen sind nach Kenntnis des Autors nicht publiziert. Aus Gesprächen mit Entwicklern führender Systemanbieter in Deutschland kann jedoch vermutet werden, daß

1. die Kosten für Anpassungsarbeiten oft die Kosten des Grundsystems übersteigen, vgl. hierzu z.B. auch Kurbel (1993), S. 331, und
2. der Zeitbedarf für Anpassungsarbeiten i.d.R. Monate beträgt und oft mehrere Anpassungs-/Probebetriebs-Zyklen ´gefahren´ werden.

Planungsumfeld der Umfang erforderlicher Planänderungen möglichst gering gehalten werden kann und ein Plan über ein möglichst breites Spektrum möglicher (und wahrscheinlicher) Störungen hinweg eine hohe Qualität aufweisen soll[14)].

2.2 Planungsprämissen

Den nachfolgend vorgestellten Modellen zur Maschinenbelegungsplanung liegen folgende Prämissen zugrunde:

Ziel: Minimierung der Zykluszeit

1. betriebsmittelbezogene Prämissen[15)]
   - keine intensitätsmäßigen Anpassungen
   - keine Ausfallzeiten
   - pro Betriebsmittel höchstens 1 Fertigungsauftrag[16)] zur selben Zeit
2. auftragsbezogene Prämissen
   - gegebener Auftragsbestand
   - bekannte, zykelfreie, nicht notwendigerweise identische Maschinenfolgen
   - keine Unterbrechung der Bearbeitung
   - keine Prioritäten oder Liefertermine
   - keine überlappende Fertigung
3. sonstige Prämissen
   - reihenfolgeunabhängige Rüstzeiten

---

14) Ähnlich wie bei den oben angeführten Begriffen der Effizienz und Effektivität bleibt auch der Begriff *robuster Planungsergebnisse* auf dem verwendeten Abstraktionsniveau sehr vage. Auch in diesem Fall muß eine exakte Definition anwendungsspezifisch vorgenommen und hier eine intuitive Vorstellung des Begriffes vorausgesetzt werden. Unter der (plausiblen) Annahme, daß die ´Qualität´ eines Plans durch Störungen verringert wird, ist das Ziel *robuster Planungsergebnisse* in jedem Fall eine Verknüpfung der konfliktären Zielsetzungen *Qualitätsmaximierung* und *Maximierung des abgedeckten Störungsspektrums.*

15) Die Begriffe *Maschine* und *Betriebsmittel* werden im folgenden - und ausschließlich im Kontext der Maschinenbelegungsplanung - synonym benutzt.

16) Für die Bearbeitung eines Fertigungsauftrags an einem Betriebsmittel, die nicht durch die Bearbeitung mindestens eines anderen Fertigungsauftrags an dem selben Betriebsmittel unterbrochen wird, werden im folgenden die Begriffe *Fertigungsarbeitsgang*, *Arbeitsgang* oder *Operation* synonym benutzt.

Unter diesen restriktiven und stark vereinfachenden Annahmen können Lösungen i.S. von Planungsmoduln, die in einem Szenario gemäß Kap. 2.1 praktisch eingesetzt werden können, nicht entwickelt werden. Ziel bei der Formulierung, der Implementierung und dem Test der hier vorgestellten Modelle war es vielmehr, ein ´Fingerspitzengefühl´ für die Modellierung von Optimierungsproblemen auf Basis adaptiver Verfahren zu bekommen. Die angegebenen Prämissen charakterisieren dabei eine Problemklasse, dessen Lösung *mindestens* ermöglicht werden sollte. Eine Flexibilisierung (i.S. steigender Adaptibilität) bei der Entwicklung einer Feinplanungskomponente kann durch eine (sukzessive) Vernachlässigung der angegebenen Prämissen erreicht werden. Bei den hier dargestellten Entwicklungen wurde von den Prämissen teilweise in Details abgewichen.

3 Ein konnektionistisches Modell als Gedankenexperiment

3.1 Informationsverarbeitung in Neuronalen Netzwerken

Künstliche *Neuronale Netzwerke*[17] sind Modelle informationsverarbeitender Systeme, die auf Prinzipien kognitiver Prozesse in natürlichen Neuronenverbänden basieren. Anwendungsorientierte Interpretationen Neuronaler Netzwerke werden als *konnektionistische Modelle* bezeichnet. Die Informationsverarbeitung erfolgt in Neuronalen Netzwerken i.d.R. hochgradig *parallel* in einem Netz einfach aufgebauter, vielfältig miteinander verknüpfter Verarbeitungseinheiten (*Neuronen*, *Knoten*, *Prozessorelemente*).[18]

Abb. 1 skizziert den Informationsverarbeitungsprozeß auf Neuronenebene: Die Knoten eines Netzwerkes tauschen über *gewichtete Verbindungen* Signale miteinander aus. Die Stärke eines Signals hängt vom Gewicht einer Verbindung (w_{ji}) sowie von der sogenannten *Aktivierung* des sendenden Neurons j ab, die wiederum von dessen Eingangssignalen beeinflußt wird. Die *Netzeingabe* (*Input*) eines Neurons i zu einem Zeitpunkt t[19] wird aus allen ankommenden Signalen ($o_j(t-1)$[20]) über eine *Input*- oder *Propagierungsfunktion* $net_i(t)$ berechnet. Aus diesem Input werden die *Aktivierung* des Neurons über die *Aktivierungs*- oder *Transferfunktion* $a_i(t)$ und schließlich das *Ausgangssignal* (*Output*) über eine *Ausgabe*- oder *Outputfunktion* $o_i(t)$ bestimmt.

---

17) Im folgenden wird auf den Zusatz ´künstlich´ verzichtet.

18) Vgl. Kemke (1988), S. 144f.

19) Die diskrete Darstellung der Zeit resultiert aus der Modellierung analoger Prozesse auf digitalen Rechnern, vgl. hierzu z.B. Glover/Greenberg (1989), p. 122, vgl. auch Anmerkung 24).

20) Das Argument (t-1) zeigt an, daß die im Zeitpunkt t an Neuron i ankommenden Signale identisch mit den von anderen Neuronen zum Zeitpunkt t-1 ausgesendeten Signalen sind.

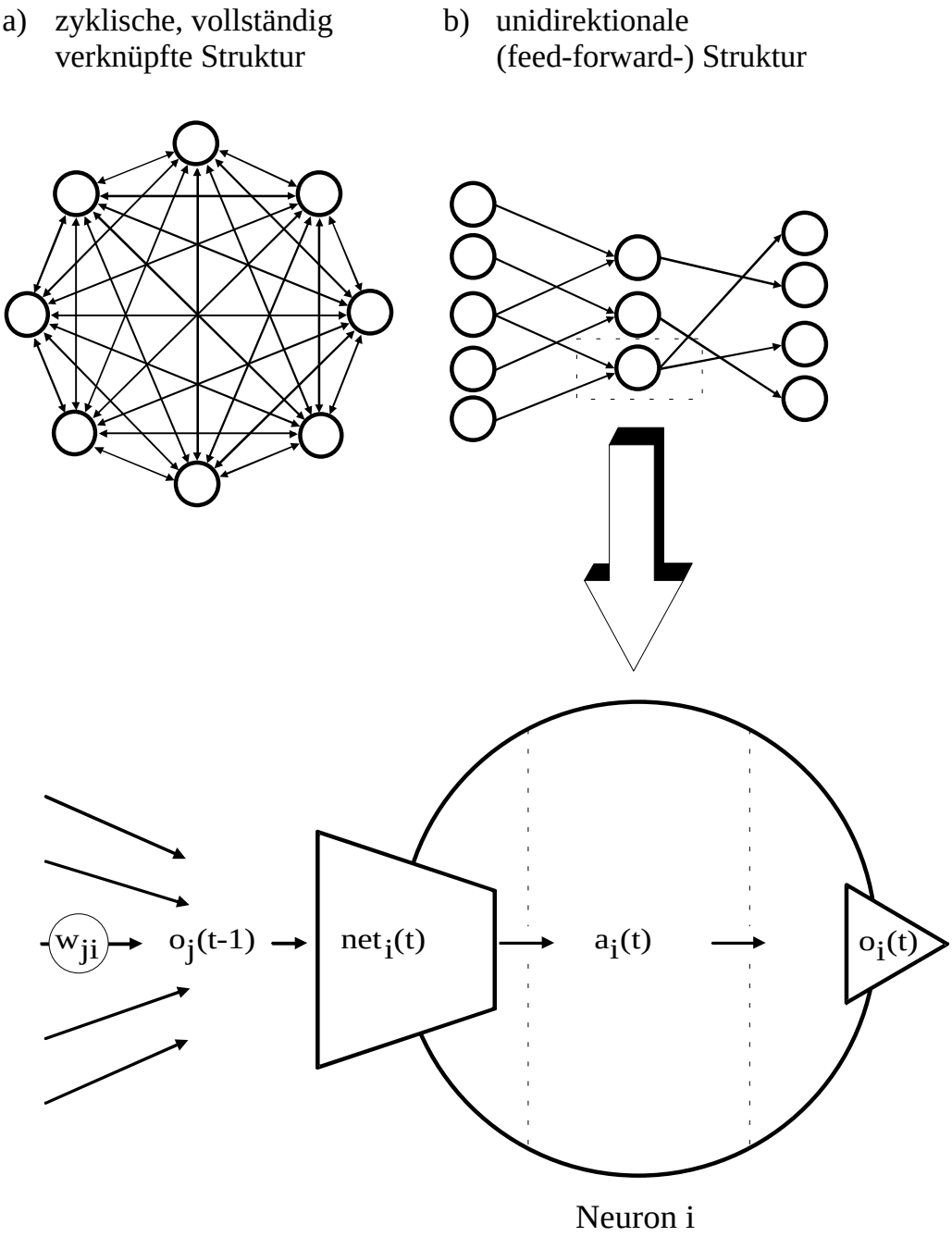

Abb. 1: Informationsverarbeitung in einem Neuronalen Netz

Der Informationsverarbeitungsprozeß wird durch die externe Aktivierung bestimmter nach außen ´sichtbarer´ Knoten, der *Inputneuronen*, initiiert und so die Neuberechnung der Aktivierungen aller mit den Inputneuronen verbundenen Knoten ausgelöst. Falls im Netz Signale nur in eine Richtung gesendet werden (Fall b) in Abb. 1), ist dieser Vorgang beendet, wenn keine nachfolgenden Knoten mehr existieren. Sind hingegen Rückkopplungen (Zyklen) zugelassen (Fall a) in Abb. 1), wird eine Terminierungsbedingung definiert, z.B. das Erreichen eines *stabilen* Zustands, in dem sich die Aktivierungen der

Neuronen nicht mehr ändern[21]. Das Ergebnis der Informationsverarbeitung kann an bestimmten nach außen sichtbaren Knoten, den *Outputneuronen*, abgelesen werden.

Für eine gegebene Netzwerktopologie i.S. gegebener Art und Anzahl der Neuronen sowie gegebener Verbindungen bestimmen die Stärken der Verbindungen einerseits und die Initialisierung der Aktivierungen der Inputneuronen andererseits das Ergebnis des Informationsverarbeitungsprozesses[22]. Die Gewichte der Verbindungen werden je nach Netzwerktyp und Einsatzbereich entweder in einer Trainings- oder Lernphase ermittelt (´gelernt´) oder aber direkt berechnet[23]. Insbesondere bei der seriellen Simulation komplexer Netzwerke[24] mit zyklischen Strukturen werden Verarbeitungsergebnisse darüber hinaus durch die *Updatestrategie* beeinflußt, die eine Ordnung festlegt, in der die Aktivierungen der Neuronen im Zeitablauf berechnet werden.

---

21) Um die Konvergenz (i.S. des Erreichens eines stabilen Zustandes) zu beschleunigen, können ´pseudo-stabile´ Zustände definiert werden, z.B. für den Fall, daß innerhalb eines gegeben Zeitraumes nur hinreichend wenige und/oder geringe Änderungen auftreten. Die Angemessenheit dieser Vorgehensweise ist problemspezifisch zu prüfen.

22) Dabei wird von - bei einigen Modellen möglichen und auch erwünschten - stochastischen Einflüssen auf den Verarbeitungsprozeß abgesehen.

23) Lernen wird in Neuronalen Netzwerken i.d.R. realisiert, indem zunächst die Gewichte mit (zufälligen) Startwerten initialisiert und anschließend dem Netz Trainingsdaten präsentiert werden (durch entsprechende Aktivierung der Inputneuronen). Sind die jeweils erwünschten Ergebnisse bekannt, so kann auf Basis von Soll-/Ist-Vergleichen zwischen den erzeugten und den jeweils erwünschten Output-Werten eine Adaption der Gewichte derart erfolgen, daß der aufgetretene Fehler minimiert wird (überwachtes Lernen). Andere Verfahren können auch angewendet werden, falls keine bekannten Output-Werte existieren (unüberwachtes Lernen): So können z.B. alle Verbindungen zwischen den Inputneuronen und dem Outputneuron mit der höchsten Aktivierung verstärkt werden (konkurrierendes Lernen), oder es werden nur die Verbindungen zwischen Neuronen verstärkt, die gleiche oder ähnliche Aktivierungen aufweisen (Lernen durch Verstärkung, Hebb´sche Lernregel). Während das Lernen i.d.R. ein repetitiver Prozeß ist, bei dem die Trainigsdaten dem Netz wiederholt präsentiert werden und die Präsentationshäufigkeit einen für die Qualität durchaus kritischen Parameter darstellt, können in speziellen Modellen, in denen das Netz als Assoziativspeicher dient (und die zu speichernden Vektoren - als zu speichernde Muster - linear unabhängig sind), Gewichte auch direkt aus den zu speichernden Mustern berechnet werden. Vgl. zum Lernen z.B. Rojas (1993), S. 73-119 und S. 203-224, sowie zum Lernen in Assoziativspeichern, insbesondere im Hopfield-Netz, auch S. 259-268 und S. 291-300.

24) In natürlichen Neuronenverbänden erfolgt die Signalübermittlung parallel. Unter *serieller Simulation* wird hier die Simulation auf allen konventionellen Ein- oder auch Multiprozessor-Systemen verstanden, bei der nicht für jedes Neuron ein Prozessor exklusiv zur Verfügung steht, sondern die Signalverarbeitung für mehrere Neuronen sukzessive berechnet wird. Selbst wenn die Parallelität dabei künstlich erzeugt wird, z.B. indem alle Aktivierungen erst berechnet und dann ´zeitgleich´ gesetzt werden, erfolgt eine künstliche Synchonisierung der Verarbeitung, die auf der Simulation auf seriellen Hardware- und Softwarestrukuren beruht.

3.2 Eigenschaften und Anwendungen Neuronaler Netzwerke

Zusammenfassend werden die Elemente Neuronaler Netzwerke, auf die der Netzwerkmodellierer gestaltend einwirken kann, aufgeführt:

1. Netzwerkstruktur
   a) atomare Komponenten
      - Neuronen
        - Input- oder Propagierungsfunktion
        - Aktivierungs- oder Transferfunktion
        - Ausgabe- oder Outputfunktion
      - Verbindungen
        - Verbindungsrichtungen
        - Verbindungsgewichte

   b) Gruppierung der Neuronen (*Schichten*, *Layer*)
2. Dynamik
   a) Lernstrategie
   a) Updatestrategie

Die Entwicklung konnektionistischer Modelle kann als Parametrisierung eines Grundmodells Neuronaler Netzwerke i.S. der Ausgestaltung der genannten Elemente aufgefaßt werden[25]. Trotz der Kombinationsvielfalt potentieller Parameterausprägungen weisen konnektionistische Modelle - allerdings nicht durchgängig - gemeinsame Eigenschaften auf:

- die verteilte Repräsentation von Informationen
- die Parallelität der Informationsverarbeitung
- eine begrenzte Selbstorganisation
- Generalisierungsfähigkeit
- Lernfähigkeit
- Fehlertoleranz

---

25) Diese Interpretation setzt voraus, daß eine Parametrisierung nicht nur quantitativ (über Parameter*werte*), sondern auch qualitativ (über Parameter*ausprägungen*, die z.B. auch Funktionen und/oder Operatoren darstellen können) erfolgen kann.

Die genannten Eigenschaften motivieren[26] eine Anwendung Neuronaler Netzwerke auch auf Satisfizierungs- und Optimierungsprobleme. Vorteile verspricht dabei zum einen die Möglichkeit der Parallelisierung - sofern eine Realisierung auf parallelen Architekturen erfolgt. Zum anderen verspricht die Fähigkeit der Selbstorganisation eine Reduzierung des Modellierungsaufwandes, da nicht alle Wirkungszusammenhänge zwischen Ein- und Ausgabedaten spezifiziert werden müssen. Das Einsatzspektrum kann dann u.U. auch auf Problemstellungen erweitert werden, in denen diese Zusammenhänge nicht alle a priori bekannt sind.

Ansätze zur Lösung komplexer Optimierungsprobleme wurden vor allem auf Basis von sogenannten Hopfield-Netzwerken entwickelt und nutzen die Eigenschaft entsprechender Netzwerke, eine über der Netzwerkstruktur definierte ´Energiefunktion´ zu minimieren. Ein solcher Ansatz soll im folgenden skizziert werden.[27]

3.3 Ein Hopfield-Netzwerk zur Maschinenbelegungsplanung

3.3.1 Hopfield-Netzwerke

Hopfield-Netzwerke[28] sind in der Grundform vollständig und symmetrisch verknüpfte Netzwerke: Jedes Neuron sendet an alle anderen Neuronen (vollständige Verknüpfung, vgl. Fall a) in Abb. 1) außer an sich selbst (keine direkte Rückkopplung) Signale über Verbindungen, deren Gewichte jeweils in beide Richtungen gleich sind (symmetrische Verknüpfung, $w_{ij} = w_{ji}$). Die Neuronen des Hopfield-Netzwerkes entsprechen einfachen sogenannten McCulloch-Pitts-Neuronen[29], die als Aktivierung und Output nur Binärwerte erlauben (je nach Modell gilt $a_i(t)$, $o_i(t) \in \{0, 1\}$ oder $a_i(t)$, $o_i(t) \in \{-1, 1\}$).

---

26) Vgl. Horster u.a. (1993), S. 4.

27) Eine interessante Alternative ist ein Ansatz, der auf selbstorganisierenden Netzwerken nach T. Kohonen basiert und in dem Elemente einer Problemlösung, z.B. die Operationen in einem Maschinenbelegungsproblem, um Positionen in einer potentiellen Lösung (dem Maschinenbelegungsplans) über das Wettbewerbslernen konkurrieren. Ein solcher Ansatz zum Scheduling von Prozessen in einem Computersystem wird in Hemani/Postula (1990) beschrieben.

28) Zu Hopfield-Netzwerken vgl. Hopfield (1982) oder z.B. auch Rojas (1993), S. 277ff. Einen ersten Ansatz zur Lösung kombinatorischer Optimierungsprobleme mit Hopfield-Netzwerken stellten J. Hopfield und D. Tank 1985 vor, vgl. Hopfield/Tank (1985).

29) Das entsprechende einfache Neuronenmodell wurde in McCulloch/Pitts (1943) vorgestellt.

Die Dynamik eines Hopfield-Netzwerkes wird durch die Inputfunktion als Summation der gewichteten Eingangssignale, die Aktivierungsfunktion als Treppenfunktion[30] und die Outputfunktion als Identitätsfunktion beschrieben (hier: $a_i(t)$, $o_i(t) \in \{0, 1\}$):

$$\textbf{Input}: \quad net_i(t) = \sum_j w_{ji} \cdot o_j(t\text{-}1) \tag{1}$$

$$\textbf{Aktivierung}: \quad a_i(t) = \begin{cases} 1, & \text{falls } net_i(t) > S_i \\ 0, & \text{falls } net_i(t) < S_i \\ a_i(t\text{-}1) & \text{falls } net_i(t) = S_i \end{cases} \tag{2}$$

$$\textbf{Output}: \quad o_i(t) = a_i(t) \tag{3}$$

Die Anwendung von Hopfield-Netzwerken auf Optimierungsprobleme beruht auf der Eigenschaft, daß über der Netzwerkstruktur eine Klasse von Funktionen definiert werden kann, deren Wert aufgrund der angegebenen Dynamik im Zeitablauf nur abnimmt[31]. In diese sogenannten *Energiefunktionen* gehen die Aktivierungen (oder Outputsignale) der Neuronen paarweise multipliziert und mit dem jeweiligen Verbindungsgewicht bewertet ein[32]:

$$\textbf{Energie}: \quad E(t) = -1/2 \cdot \sum_i \sum_j (w_{ij} \cdot a_i(t) \cdot a_j(t)) + \sum_i (S_i \cdot a_i(t)) \tag{4}$$

Die Lösung eines Optimierungsproblems mit Hilfe eines Hopfield-Netzwerkes wird ermöglicht, falls es gelingt, die Zustände des Optimierungsproblems so auf die die Topologie (d.h. auf die (binären) Neuronen sowie die Verbindungsgewichte) eines Hopfield-Netzwerkes abzubilden, daß die (zu minimierende) Zielfunktion des Optimierungsproblems einer über dem Netzwerk definierten Energiefunktion entspricht.

---

30) Die Treppenfunktion (*Signum*) kann durch spezifische Schwellenwerte S_i auf der Abzisse (über dem Input) verschoben werden.

31) Vgl. z.B. Rojas (1993), S. 284.

32) Die Funktion entspricht dem (erweiterten) Skalarprodukt binärer Vektoren (Hamilton-Funktion). Die hier notierte Energiefunktion entspricht der in der Literatur üblicherweise angegebenen Form. Der Ausdruck S_i gibt dabei (in Anlehnung an die Herkunft des Hopfield-Netzwerkes als physikalisches Beschreibungsmodell für sogenannte Spingläser) die Stärke eines externen, lokal wirkenden Feldes an, das in der abstrakten Formulierung eines Hopfield-Netzwerkes seine Entsprechung in den Schwellenwerten der einzelnen Neuronen findet.

3.3.2 Die Modellierung des Maschinenbelegungsproblems

3.3.2.1 Die Darstellung von Plänen

Maschinenbelegungspläne können im Hopfield-Modell repräsentiert werden, indem

- Fertigungsaufträge,
- Betriebsmittel und
- Positionen (von Fertigungsaufträgen auf Betriebsmitteln)

auf die Neuronen eines Hopfield-Netzwerkes abgebildet werden, das dementsprechend als 3-dimensionales Neuronenmodell dargestellt werden kann (vgl. Abb. 2). Fertigungsaufträge und Betriebsmittel werden dabei als aufsteigend durchnumeriert - beginnend bei 1 - interpretiert.

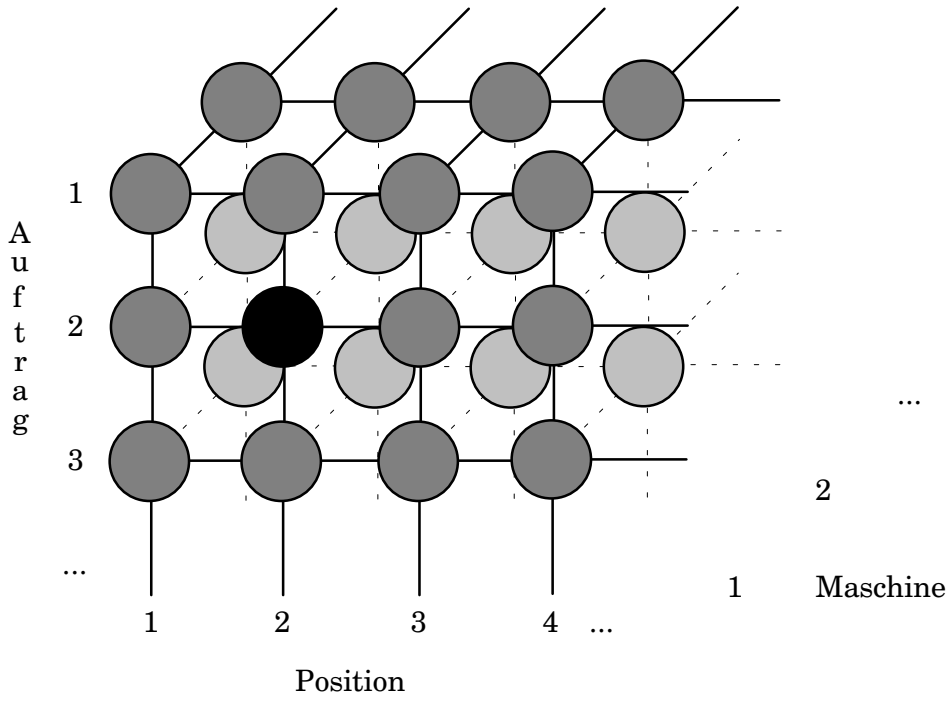

Abb. 2: 3-D-Darstellung eines Belegungsplans

Positionen werden in dem Modell aus Betriebsmittelsicht dargestellt, beispielsweise zeigt die Aktivierung des mit ´●´ gekennzeichneten Neurons an, daß Auftrag 2 in der Auftragsfolge von Betriebsmittel 1 an zweiter Position steht oder, anders ausgedrückt, daß

der Auftrag 2 der zweite Auftrag ist, der an Betriebsmittel 1 bearbeitet wird[33]. Für die Binärdarstellung wird vereinbart:

$$a_{ijk} = \begin{cases} 1, & \text{falls Auftrag i auf Betriebsmittel j an k-ter Position bearbeitet wird,} \\ 0 & \text{sonst} \end{cases} \quad (5)$$

Eine Modellierung gemäß Abb. 2 hat zunächst rein deskriptiven Charakter. Damit das Problem der Maschinenbelegungsplanung in einem Modell auf Basis des einfachen skizzierten Hopfield-Netzwerkes gelöst werden kann, müssen die Nebenbedingungen und die Zielfunktion des Problems integriert werden.

3.3.2.2 Die Berücksichtigung von Nebenbedingungen

Die gewählte Darstellungsform allein schließt eine Generierung unzulässiger Pläne nicht aus. Nach Festlegung der Netzstruktur in der dargestellten Weise ist eine geeignete Wahl der Verbindungsgewichte das einzige Instrument, über das ein Ausschluß unzulässiger Pläne ohne externe Operationen[34] realisiert werden kann. Als Beispiel soll hier die Vermeidung von Doppelbelegungen erläutert werden.

Eine Doppelbelegung tritt im Modell genau dann auf, falls mindestens 2 Aufträge an derselben Position an einem Betriebsmittel eingelastet werden, d.h., falls gilt:

$$a_{ijk} = a_{hjk} = 1 \text{ für } i \neq h \quad (6)$$

Abb. 3 veranschaulicht das Auftreten von Doppelbelegungen exemplarisch. Die schwarz eingefärbten Neuronen (´●´) zeigen an, daß die Aufträge 1 und 3 jeweils als zweiter zu bearbeitender Auftrag (an zweiter Position) an Betriebsmittel 1 eingelastet wurden.

---

33) Eine Darstellung aus Auftragssicht wäre demgegenüber so zu interpretieren, daß Betriebsmittel 1 an zweiter Position in der Maschinenfolge von Auftrag 2 steht.

34) Externe Operationen können beispielsweise durch *Repair-Algorithmen* realisiert werden, die unzulässige Pläne in zulässige Pläne transformieren; vgl. zu Repair-Algorithmen z.B. Bierwirth (1993), S. 88. Um die Konvergenzfähigkeit des Systems zu erhalten, sollte das Planungsverfahren, hier also das Neuronale Netzwerk, auch in diesem Fall ´zielgerichtet´ arbeiten und entweder zulässige Pläne mit kurzen Zykluszeiten erzeugen oder solche unzulässigen Pläne, die der Repair-Algorithmus in zulässige Pläne mit kurzen Zykluszeiten transformieren kann.

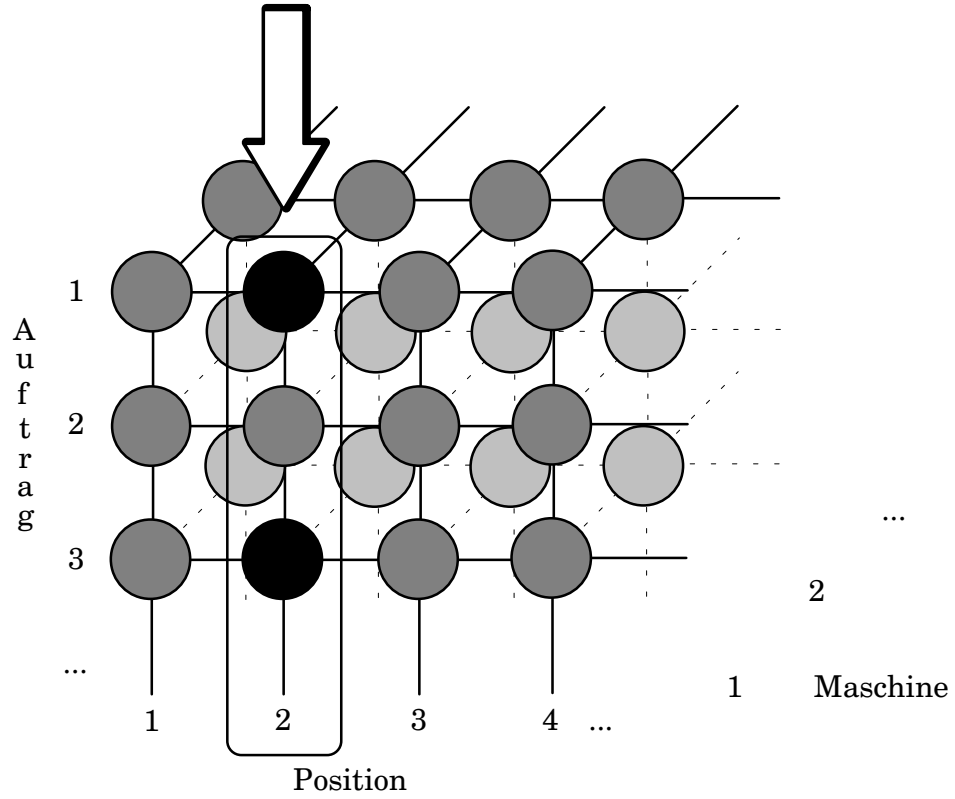

Abb. 3: Doppelbelegungen im Neuronenmodell

Die Dynamik des Hopfield-Netzwerkes stellt kein Instrument zur Verfügung, die gleichzeitige Aktivierung zweier Neuronen zu *verbieten* - jedoch kann sie durch eine geeignete Wahl der Verbindungsgewichte *bestraft* werden. Zu diesem Zweck müssen die Gewichte so gewählt werden, daß die Energie des Netzes gemäß Ausdruck (4) bei einer gleichzeitigen Aktivierung der betreffenden Neuronen steigt.

Um im dargestellten Beispiel die Anzahl ´geplanter´ Doppelbelegungen im Netz zu zählen, wird zunächst für alle Betriebsmittel h geprüft, ob es irgendwelche Aufträge i und k gibt, die beide gleichzeitig an einer beliebigen Positionen j bearbeitet werden sollen[35]:

$$\#\ \text{Doppelbelegungen} = \sum_j \sum_h \sum_i \sum_{k>i} (a_{ihj} \cdot a_{khj}) \qquad (7)$$

Da die in Ausdruck (7) enthaltenen Aktivierungsprodukte ($a_{ihj} \cdot a_{khj}$) auch in Ausdruck (4) enthalten sind, kann durch eine geeignete Wahl der Gewichte $w_{ihj,khj}$ erreicht werden, daß sich eine Doppelbelegung (isoliert betrachtet, s.u.) energiesteigernd auswirkt

35) Die Einschränkung, daß mit einem Auftrag i nur solche Aufträge k verglichen werden, deren Index größer als i ist (k > i), dient der Vermeidung von ´Doppelzählungen´: Beispielsweise gilt für i=3 und k=2, daß der Ausdruck ($a_{3hj} \cdot a_{2hj}$) schon vorher für i=2 und k=3 als ($a_{2hj} \cdot a_{3hj}$) berechnet wurde und somit nicht mehr gezählt werden muß.

und somit (tendenziell) vermieden wird. Werden alle möglichen Doppelbelegungen (d.h. auf allen Positionen aller Betriebsmitteln für alle Aufträge) mit dem gleichen Verbindungsgewicht A ´bewertet´, so kann folgender Term als Summand von Ausdruck (4) angegeben werden:

$$A \cdot \sum_j \sum_h \sum_i \sum_{k>i} (a_{ihj} \cdot a_{khj}) \tag{8}$$

Im vorliegenden Beispiel muß der Faktor A einen negativen Wert erhalten, um in Ausdruck (4) energiesteigernd zu wirken. Es muß jedoch beachtet werden, daß eine Vermeidung von Doppelbelegungen durch eine entsprechende Wahl der Gewichte nicht sicher ausgeschlossen werden kann, da die Aktivierung eines Elementes über *alle* Verbindungen des entsprechenden Neurons in die Energiefunktion eingeht und die energiesteigernde Wirkung u.U. durch energiemindernde Einflüsse der Verbindungen zu anderen (von Doppelbelegungen nicht betroffenen) Neuronen neutralisiert werden kann.

3.3.2.3 Zusatzinformationen

Die angegebene Modellierung ist unvollständig. Zum einen hat sie statischen Charakter: Es werden keinerlei Zeitgrößen berücksichtigt, und die Zielfunktion (Minimierung der Zykluszeit) kann somit nicht explizit modelliert werden. Dies hat zur Folge, daß ein kodierter Plan hinsichtlich seines Zielerfüllungsgrades jeweils ex post bewertet werden muß, z.B. indem ein einfacher Scheduler die einzelnen Fertigungsarbeitsgänge gemäß der im Modell durch die Positionen vorgegebenen Reihenfolgen zeitgenau auf den Betriebsmitteln einlastet und die resultierende Zykluszeit bestimmt[36]. Ein solcher Scheduler muß Zugriff auf die Bearbeitungszeiten[37] erhalten:

$$T = \begin{bmatrix} t_{11} & \dots & t_{1m} \\ . & & . \\ . & & . \\ . & & . \\ t_{n1} & \dots & t_{nm} \end{bmatrix} \tag{9}$$

t_{ij} : Bearbeitungzeit von Auftrag i auf Betriebsmittel j; $i \in \{1,...,n\}$, $j \in \{1,...,m\}$

---

36) Eben weil mit den im Netz vorhandenen Informationen allein die Zykluszeit (das Ziel) nicht bestimmt werden kann, mußte eine Energiefunktion gefunden werden, die ersatzweise optimiert (minimiert) wird, vgl. auch Kap. 3.3.3.

37) Die hier aus Gründen der Anschaulichkeit angeführte Matrix-Notation der Bearbeitungszeiten setzt - ebenso wie die folgenden Notationen der Maschinenfolgen und Starttermine - voraus, daß zusätzlich zu den in Kap. 2.2 genannten Prämissen jeder Auftrag nur einmal auf einem Betriebsmittel bearbeitet wird.

Ein weiterer Mangel des Modells ist, daß ohne Erweiterungen nicht alle Nebenbedingungen erfaßt werden können. Während beispielsweise aufgrund der Interpretation des Modells (Positionen aus Betriebsmittelsicht) zwar betriebsmittelbezogene Reihenfolgebedingungen berücksichtigt werden können, müssen auftragsbezogene Reihenfolgebedingungen wie die Einhaltung vorgegebener Maschinenfolgen extern überprüft werden, da die benötigten Informationen nicht modelliert wurden. Als Konsequenz wird dem Scheduler auch die Aufgabe der Konsistenzsicherung übertragen und ein Zugriff auf extern repräsentierte Maschinenfolgen ermöglicht:

$$Q = \begin{bmatrix} q_{11} & \cdots & q_{1m} \\ . & & . \\ . & & . \\ . & & . \\ q_{n1} & \cdots & q_{nm} \end{bmatrix} \tag{10}$$

q_{ij} : Betriebsmittel j steht an q-ter Position in der Maschinenfolge von Auftrag i;
$i \in \{1,...,n\}, j \in \{1,...,m\}$)

Der Scheduler prüft also die vom Netzwerk ermittelten Reihenfolgen auf Zulässigkeit und ermittelt ggf. mit Hilfe der Zusatzinformationen über Bearbeitungszeiten und Maschinenfolgen die entsprechenden zulässigen Maschinenbelegungspläne. Zur weiteren Disposition wird eine Betriebsmittelbelegung mit (ebenfalls während des Scheduling ermittelten) Startterminen versehen:

$$H = \begin{bmatrix} h_{11} & \cdots & h_{1m} \\ . & & . \\ . & & . \\ . & & . \\ h_{n1} & \cdots & h_{nm} \end{bmatrix} \tag{11}$$

h_{ij} : Starttermin von Auftrag i auf Betriebsmittel j; $i \in \{1,...,n\}, j \in \{1,...,m\}$

Anschließend kann der Plan im Fertigungsleitstand angezeigt und weiterbearbeitet werden.

3.3.3 Kritik

Der Ansatz, die Maschinenbelegungsplanung in der skizzierten Weise auf Basis eines Hopfield-Netzwerkes durchzuführen, weist als Mangel vorrangig die unvollständige Modellierung des Ausgangsproblems auf. Negative Wirkungen sind im einzelnen:

- **Generierung unzulässiger Lösungen**

  Jeder Lösungskandidat muß (teuren) Zulässigkeitstests unterzogen und ggf. verworfen werden. Die Kosten der Tests werden von der Art und Anzahl der zu berücksichtigenden und nicht modellierbaren Nebenbedingungen bestimmt.

- **topologieabhängige Energiefunktion**

  Die zu optimierende Funktion ist implizit durch das Modell vorgegeben und der Abbildungsspielraum für den ´Netzwerkmodellierer´ somit stark eingeschränkt. Falls es - wie im Beispiel - nicht gelingt, die Zielfunktion des realen Problems eineindeutig auf die über dem Hopfield-Netzwerk definierte Energiefunktion abzubilden, ist nicht sichergestellt, das ein Zustand minimaler Energie auch einer optimalen Problemlösung entspricht.

- **geringe Informationsdichte**

  Die (symbolisch interpretierte) Binär-Repräsentation im Hopfield-Netzwerk führt zu explodierenden Netzstrukturen, da alle abzubildenden Eigenschaften des realen Problems mittels eines 2-wertigen Alphabets (Bool´sche Werte) dargestellt werden müssen. Während beispielsweise die Position eines Auftrages auf einem Betriebsmittel mit Hilfe der natürlichen Zahlen immer in einem Wert kodiert werden könnte, entspricht die Anzahl der hierfür im dargestellten Hopfield-Modell benötigten Neuronen der Anzahl der Aufträge[38)].

  Allgemein wächst bei der angegeben Darstellungsweise die Anzahl k der Neuronen quadratisch mit der Anzahl m der Betriebsmittel und proportional zur Anzahl n einzulastender Aufträge (vgl. Abb. 2): $k = n \cdot m^2$. Für die Laufzeit eines Systems ist zu berücksichtigen, daß sich die Anzahl der Verbindungen - und damit die Anzahl der zur Berechnung einer Aktivierung jeweils durchzuführenden Multiplikationen - für k Neuronen als $\frac{k \cdot (k-1)}{2}$ berechnet!

- **willkürliche Bestimmung der Gewichte**

  Bei der Formulierung der *Penalty-Terme* (Bestrafung von Restriktionsverletzungen, vgl. Kap. 3.3.2.2) in der Energiefunktion werden die Gewichte, die als *Penalty-Faktoren* dienen, nicht berechnet, sondern willkürlich festgelegt. Der Prozeß der Lösungsfindung wird damit ebenfalls willkürlich beeinflußt: Eine *milde* Bestrafung von Restriktionsverletzungen (betragsmäßig kleine Penalty-Faktoren) kann die Ge-

38) Auch dies gilt nur unter der Bedingung, daß jeder Auftrag genau einmal auf einem Betriebsmittel bearbeitet wird.

nerierung unzulässiger Lösungen nicht effektiv vermeiden, da die Wahrscheinlichkeit wächst, daß der (kleine) energiesteigernde Effekt der Bestrafung durch energiemindernde ´Nebeneffekte´ überlagert wird. Zu *strenge* Bestrafungen wiederum können verhindern, daß bestimmte unzulässige Zustände angenommen werden, die ´auf dem Weg zum Optimum liegen´[39].

- **Terminierung in lokalen Energieminima**

  Die Eigenschaft von Hopfield-Netzwerken, Zustandsänderungen nur im Falle abnehmender Energieniveaus zuzulassen, verhindert, daß Zustände verlassen werden, in deren ´näheren Umgebung´[40] alle Zustände ein höheres Energieniveau aufweisen. Handelt es sich bei einem solchen Zustand um ein lokales Optimum, so kann das globale Optimum vom Netzwerk nicht mehr gefunden werden.

Auch wenn eine Verbesserung hinsichtlich der Vermeidung einiger der angeführten Schwächen mit teilweise geringem Aufwand erreichbar ist[41], weist die dargestellte Heuristik auf Basis eines Hopfield-Netzwerkes keinerlei Vorteile auf, die ein solches Vorgehen rechtfertigen. Im folgenden Kapitel wird daher ein Verfahren vorgestellt, das die Mängel des ersten Ansatzes umgeht.

---

39) Da der Lösungsprozeß im Hopfield-Netzwerk nicht ´sprunghaft´ sondern durch kleine (bitweise) Änderungen geprägt ist, kann es u.U. nötig sein, unzulässige Lösungen vorübergehend zu akzeptieren, um den Lösungsraum vollständig durchsuchen zu können.

40) Im Falle der Bit-Repräsentation könnte die ´nähere Umgebung´ z.B. alle Zustände mit einer Hamming-Distanz von 1 umfassen, d.h. alle Zustände, die sich nur durch ein Bit vom Ausgangszustand unterscheiden.

41) Eine Terminierung in lokalen Energieminima läßt sich z.B. durch die Implementierung stochastischer Aktivierungsfunktionen umgehen, bei denen die Aktivierung ein Neurons gemäß einer Wahrscheinlichkeit erfolgt. Eine derartige Erweiterung einfacher deterministischer Netzwerke stellen die sogenannten Boltzmann-Maschinen dar, bei denen eine Aktivierung gemäß des Metropolis-Kriteriums (Metropolis et al. (1953)) erfolgt, vgl. z.B. Aarts/Korst (1989), pp. 13 und pp. 131. Das gleiche Prinzip der stochastischen Akzeptanz wird auch bei dem im folgenden Abschnitt dargestellten Verfahren des Simulated Annealing angewendet.

4 Simulated Annealing

4.1 Anforderungen

Auf Basis der Ergebnisse des Experiments zur Maschinenbelegungsplanung mit einem Hopfield-Netzwerk werden an einen modifizierten Ansatz folgende Anforderungen gestellt:

1. **Zulässigkeitsgarantie**

   Der durch Laufzeit-intensive Konsistenzprüfungen verursachte Overhead soll vermieden werden. Für das Problem der Maschinenbelegungsplanung wird dies vor allem durch die Komplexität der Pläne begründet: Die Feststellung von Inkonsistenzen kann i.d.R. nicht anhand direkt abrufbarer Attribute vorgenommen werden, sondern erfolgt durch ´probeweises´ Einplanen der vorgeschlagenen Auftragsreihenfolgen - die Untersuchung eines unzulässigen Plans verursacht damit approximativ die Hälfte des Rechenaufwandes, den die Feststellung des Zielfunktionswertes (Zykluszeit) eines zulässigen Plans beansprucht. Da i.d.R. nur ein Bruchteil der repräsentierbaren Pläne auch zulässig ist[42], wird der ´Löwenanteil´ des Rechenzeitbedarfs durch die Feststellung der *Un*zulässigkeit von Plänen verursacht.

2. **globale Optimierung**

   Die Einschränkung, nur lokale Optima ermitteln zu können, ist nicht akzeptabel - zumal i.d.R. keine Abschätzung für die Qualität eines lokalen Optimums angegeben werden kann.

3. **kompaktes Modell**

   Die Probleme des ersten Ansatzes werden auch durch die ungeeignete Modellierung verursacht. Beispielsweise erlaubt die hochkomplexe Netzwerkstruktur die Abbildung eines Zustandsraumes, in dem der Raum zulässiger Lösungen nur eine kleine Teilmenge darstellt (s.o.)[43] - die Generierung unzulässiger Lösungen wird somit teilweise durch die Repräsentation erst ermöglicht. Außerdem wird durch die

---

42) Vgl. auch Kap. 5.3.

43) Zur Problematik der Überdeckung des Lösungsraums durch den Suchraum vgl. auch Kap. 5.3. Das in Kap. 3 dargestellte Modell verhält sich hinsichtlich dieser Überdeckung für $n < (2 \cdot m - 1)$ sogar noch schlechter als der dort angegebene genetische Algorithmus, da in diesem Fall der Suchraum noch höher dimensioniert ist.

Verwaltung der resultierenden komplexen Strukturen ein hoher Rechenaufwand verursacht[44].

Zur Umsetzung der genannten Anforderungen wird bei der Formulierung eines neuen Ansatzes auf folgende Hilfsmittel zurückgegriffen:

- **Einsatz einer Einplanungsstrategie**

  Das Konzept, vollständige Maschinenbelegungspläne über einen Scheduler zu ermitteln, im ersten Ansatz eher ´aus der Not geboren´, wird gezielt verfeinert: Lösungskandidaten i.S. potentieller Maschinenbelegungspläne werden als einfache lineare Arbeitsganglisten dargestellt. Vollständige Pläne werden aus diesen Listen durch einen speziellen Scheduler erzeugt, der eine einfache Planungsheuristik verwendet (vgl. Kap. 4.3). Die Interpretation der Listen durch den Scheduler stellt dabei sicher, daß nur zulässige Pläne erzeugt (und damit auch repräsentiert) werden. Erkauft wird die kompakte Darstellung von Lösungskandidaten in linearen Listen durch den Verzicht auf eine explizite Zuordnung von Arbeitsgängen zu Betriebsmitteln - diese Zuordnung wird ex post durch die Planungsheuristik hergestellt.

- **Einsatz des Simulated Annealing**

  Der Scheduler dient der Bewertung von Lösungskandidaten sowie der Ermittlung der zur Weiterverarbeitung im Fertigungsleitstand nötigen Informationen. Der eigentliche Suchprozeß wird über der Struktur der linearen Operationsliste mit Hilfe des Simulated Annealing implementiert (vgl. Kap. 4.2).

4.2 Die Struktur des Simulated Annealing

Beim Simulated Annealing handelt es sich um ein stochastisches, heuristisches Verbesserungsverfahren, das mit einer im Zeitablauf abnehmenden Wahrscheinlichkeit auch temporäre Verschlechterungen zuläßt[45]. In Abb. 4 wird die Grobstruktur des Simulated Annealing dargestellt.

---

44) Für das Neuronale Netz betrifft dies die Initialisierung der Gewichte und Aktivierungen sowie insbesondere die laufende Neuberechnung der Netzeingaben und Aktivierungen (Ausgaben) der Neuronen.

45) Zum Simulated Annealing vgl. z.B. Aarts/Korst (1989). Zur Einordnung in das Verfahrensspektrum des Operations Research vgl. z.B. Domschke/Drexl (1991), S.113 u. 123.

Zur Initialisierung des Verfahrens wird zunächst ein *Lösungskandidat* LK, ein Startwert für den externen Parameter *Temperatur* T und eine Vorschrift AP für die Verringerung dieses Parameters im Zeitablauf (*Abkühlungsplan*) sowie ein *Abbruchkriterium* AK bestimmt. Auf die Ausgangslösung wird ein Verfahren angewendet, das kleine (´lokale´) Änderungen in der ´Nachbarschaft´ von LK vornimmt[46]. Der resultierende neue Lösungskandidat LK_{neu} wird als neue Ausgangslösung akzeptiert, falls er einen besseren Zielfunktionswert aufweist als die bisherige Ausgangslösung LK. Ist der Zielfunktionswert hingegen schlechter, wird LK_{neu} nur mit einer gewissen Wahrscheinlichkeit als neue Ausgangslösung akzeptiert. Die *Akzeptanzwahrscheinlichkeit* AW hängt von der Differenz der Zielfunktionswerte sowie von Temperatur T ab. Anschließend wird der Wert der Temperatur (gemäß Abkühlungsplan) modifiziert, das Abbruchkriterium geprüft und je nach Ergebnis das Verfahren entweder abgebrochen oder - beginnend mit der Anwendung des lokalen ´Änderungsverfahrens´ - wiederholt.

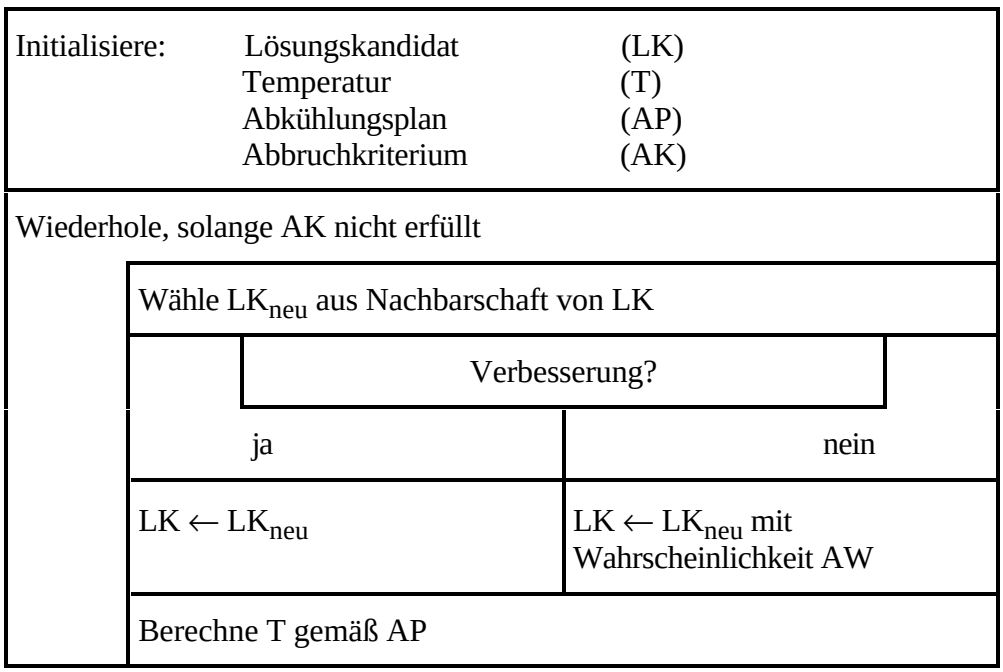

Abb. 4: Grobstruktur des Simulated Annealing

In einer phasenorientierten Darstellung lassen sich die während des Lösungsprozesses durchzuführenden Operationen (unter weitgehender Vernachlässigung der algorithmischen Struktur) in 4 Phasen einteilen:

---

46) Vgl. z.B. Aarts/Korst (1989), p. 14. Zu der formalen Darstellung und der Bedeutung von Nachbarschaftsstrukturen in lokalen Suchverfahren vgl. Aarts/Korst (1989), pp. 6.

1. **Initialisierungsphase**
   - Arbeitsgangliste erzeugen
   - Liste einplanen
   - Plan bewerten
   - Liste als Ausgangslösung speichern
   - Bewertung als vorläufiges Optimum speichern

2. **Modifikationsphase**
   - Arbeitsgangliste modifizieren (lokale Änderung)
   - modifizierte Liste einplanen
   - neuen Plan bewerten

3. **Akzeptanzphase**
   - Bewertungen vergleichen
   - ggf. neues vorläufiges Optimum speichern
   - evtl. neue Ausgangslösung akzeptieren[47)]

4. **Steuerungsphase**
   - Terminierungskriterium prüfen
   - abbrechen oder Temperatur anpassen und mit 2. fortsetzen

4.3 Die Einplanungsstrategie

Die Einplanung der in einer linearen Liste organisierten Arbeitsgänge in den Phasen 1 (Initialisierung) und 2 (Modifikation) erfolgt mittels einer einfachen Heuristik in folgenden Schritten:

1. erstes Element der Arbeitsgangliste wählen
2. Betriebsmittel bestimmen, auf dem der Arbeitsgang am schnellsten beendet wird

---

47) Ein Trennung der Schritte ´Optimum speichern´ und ´neue Ausgangslöung akzeptieren´ erfolgt, da es opportun erscheint, sich ein besseres vorläufiges Optimum auch dann zu merken, falls der entsprechende Plan nicht als neue Ausgangslösung akzeptiert wird (also im Falle der Akpzeptanz schlechterer Lösungen).

3. falls Alternativen bestehen, das in einer intern geführten Liste erstgenannte Betriebsmittel wählen

4. Arbeitsgang auf dem ausgewählten Betriebsmittel einplanen und jeweils frühestmöglichen Start- und Endzeitpunkt bestimmen

5. abbrechen, falls das Ende der Arbeitsgangliste erreicht ist, sonst nächstes Element wählen und fortsetzen mit Schritt 2.

Mit Hilfe der angegebenen Heuristik werden nicht notwendigerweise aktive Belegungspläne erzeugt, so daß selbst für eine Anwendung auf alle möglichen Operations-Permutationen nicht sichergestellt werden kann, daß ein *optimaler* Plan generiert wird. Ein Austausch gegen andere Einplanungstrategien, die dies u.U. gewährleisten[48], ist grundsätzlich problemlos möglich, da keinerlei Eingriffe in den Suchprozeß (Simulated Annealing) selbst nötig sind.

4.4 Die Modifikation der Arbeitsgangliste

Eine lokale Suche nach Lösungsverbesserungen kann beispielsweise durch eine einfache zulässige Verschiebung einer Operation innerhalb der Liste realisiert werden:

1. zufällig ein Element der Arbeitsgangliste wählen

2. zulässigen Verschiebebereich feststellen, indem der Vorgänger- und der Nachfolgerarbeitsgang innerhalb desselben Fertigungsauftrages bestimmt werden (in Abb. 5 kennzeichnen die schattierten Kästchen Arbeitsgänge eines Fertigungsauftrages):

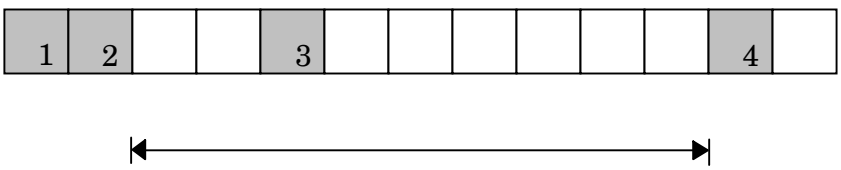

Abb. 5: Zulässiger Verschiebebereich innerhalb der Arbeitsgangliste

3. neue Position des Arbeitsganges (in Abb. 6 dunkelgrau markiert) innerhalb der Liste über eine gleichverteilte Zufallsvariable bestimmen

---

48) Das vielleicht bekannteste Verfahren dieser Art ist der Algorithmus von Giffler und Thompson (Giffler/Thompson (1960)), mit dem alle *aktiven* Belegungspläne erzeugt werden können; zur Klassifikation von Belegungsplänen vgl. z.B. Sprecher (1994), pp. 26.

4. Arbeitsgang an die neue Position verschieben, indem er aus der Liste entfernt wird, alle Arbeitsgänge zwischen der alten und der neuen Position des Arbeitsganges um eine Position eingerückt werden und der Arbeitsgang schließlich an der neuen Position wieder eingefügt wird:

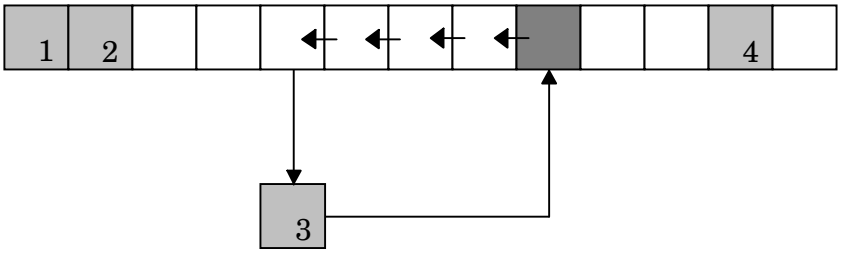

Abb. 6: Verschieben eines Arbeitsganges

4.5 Die Akzeptanz neuer Lösungen

Die Wahrscheinlichkeitsfunktion für die Akzeptanz einer Arbeitsgangliste als neue Ausgangslösung in Abhängigkeit von der Differenz ihrer Bewertung zu der der alten Ausgangslösung und dem externen Parameter T lautet allgemein:

$$p(\Delta B, T) = \begin{cases} 1, & \text{falls } \Delta B \geq 0, \\ \exp(\Delta B/T) & \text{sonst} \end{cases} \qquad (12)$$

mit $\Delta B = B(LK) - B(LK_{neu})$

Ein neuer Lösungskandidat wird im Falle eines besseren (oder gleichen) Zielfunktionswertes ($\Delta B \geq 0$) also mit Sicherheit als neue Ausgangslösung akzeptiert, während im Falle einer Verschlechterung ($\Delta B < 0$) die Wahrscheinlichkeit einer Akzeptanz auch durch den Parameter Temperatur bestimmt wird. Die Funktion $\exp(\Delta B/T)$ konvergiert für eine gegebene negative Bewertungsdifferenz ΔB für sehr große Werte von T ($T \to \infty$) gegen 1 ($p(T) \to 1$) und für sehr kleine (positive) Werte von T ($T \to 0$) gegen 0 ($p(T) \to 0$[49]). Ein typischer Verlauf der Wahrscheinlichkeitsfunktion für Lösungsverschlechterungen (hier: $\Delta B = const = -10$) ist in Abb. 7 dargestellt.

---

49) Für $T \to 0$ entspricht das Verfahren damit einem echten Gradientenabstiegsverfahren.

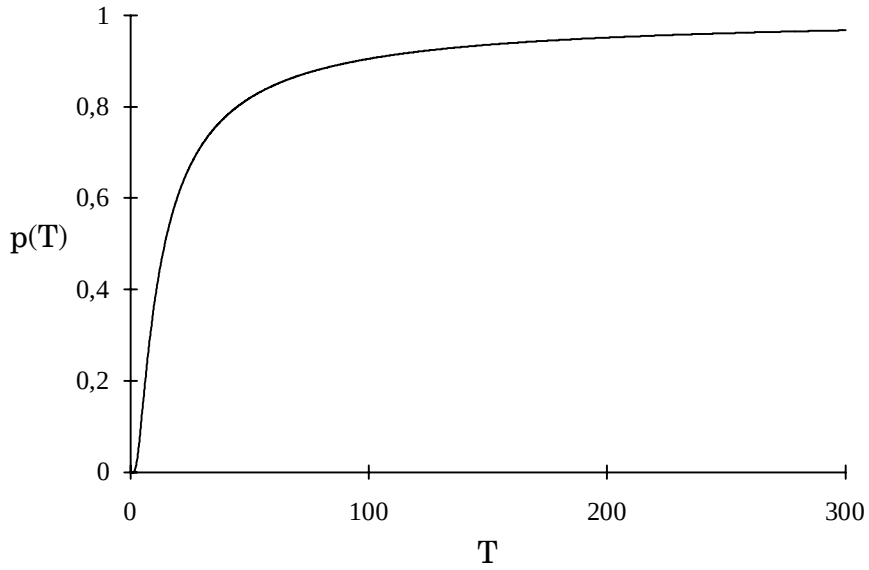

Abb. 7: Akzeptanzwahrscheinlichkeit

Der Abkühlung (Verringerung des Temperaturwertes) erfolgt mit abnehmender Geschwindigkeit und wird durch eine logarithmische Abkühlungsfunktion (Abkühlungsplan) beschrieben, die es gestattet, die Temperatur über einen längeren Zeitraum in einem für die erreichbare Lösungsqualität besonders kritischen Bereich zu halten[50]. Zu diesem Zweck wird zunächst für jede spezifische Aufgabenstellung eine Konstante S definiert:

$$\text{const } S = 4 \cdot [\#\ \text{Operationen}]^{51)} \tag{13}$$

Die Zeit wird im Abkühlungsplan über einen Schrittzähler *Count* implementiert, für den gilt:

$$\text{Startwert(Count)} = 1{,}01 \tag{14}$$

$$\text{Inkrement(Count)} = +\,0{,}01 \tag{15}$$

Die Temperatur schließlich wird aus den Parametern S und Count als

$$T = S/\ln(\text{Count}) \tag{16}$$

ermittelt. Abb. 8 stellt den Verlauf der Abkühlungsfunktion für ein Szenario mit 100 einzuplanenden Operationen dar.

---

50) Zu Anforderungen an und Konstruktion von Abkühlungsplänen vgl. z.B. Aarts/Korst (1989), pp. 57.

51) Der Faktor 4 hat sich empirisch (für gegebene Testdatenbestände in Testläufen) als konvergenzfördernd erwiesen.

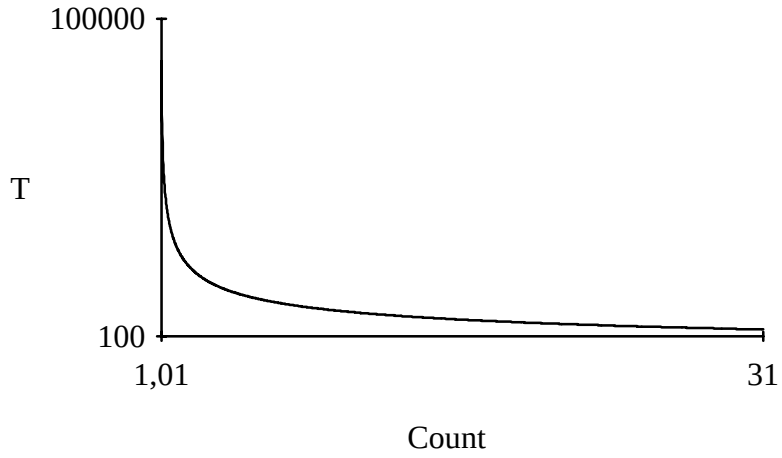

Abb. 8: Die Abkühlungsfunktion

4.6 Kritik

Aufgrund der geringeren Komplexität kann das Modell zur Generierung von Maschinenbelegungsplänen auf Basis des Simulated Annealing im Gegensatz zu dem in Kap. 3 vorgestellten Verfahren auf Basis eines Neuronalen Netzwerkes leichter implementiert werden[52]. Einige Eigenschaften des Modells und Charakteristika seiner Handhabung bieten dennoch Anlaß zur Kritik:

1. **Die Parametrisierung**

   Das Verfahren des Simulated Annealing reagiert sehr sensibel auf die Wahl der verschiedenen Parameter. Insbesondere beeinflußt die Abkühlung die Lösungsqualität. Für die Gestaltung des Abkühlungsplans bzw. die Wahl der Abkühlungsfunktion gibt es jedoch keine allgemeingültigen Vorschriften[53], so daß ´gute´ Entscheidungen i.d.R. nur aufgrund langwieriger Tests i.S. des Vergleichs alternativer Parameter-Sets getroffen werden können - ohne dabei die Qualität der getroffenen Entscheidung allgemeingültig bestimmen zu können[54].

---

52) Vgl. Ruppel/Siedentopf (1992).

53) Empfehlungen liegen allerdings vor, z.B. Hajek (1988) oder auch Aarts/Korts (1989), pp. 57.

54) Die Qualität von Lösungen wird zwar kardinal gemessen (über die Zykluszeit) - eine Skalierung kann jedoch nur auf einer Intervallskala und nicht auf einer Verhältnisskala oder einer absoluten Skala vorgenommen werden, da das Optimum (als Nullpunkt der Skala) i.d.R. nicht bekannt ist.

Das Problem der Parametrisierung wird in einigen dem Simulated Annealing sehr ähnlichen Verfahren durch eine Reduzierung der Anzahl der Parameter deutlich gemildert. Bei diesen Verfahren

2. Die Einplanungsheuristik

Durch die Verwendung einer Einplanungsheuristik (bzw. eines Schedulers) wird - wie bereits im ersten Ansatz - die Problematik der *Plan*erzeugung in ein externes Modul verlagert und die Lösungsqualität somit durch die Qualität dieses Moduls begrenzt[55].

3. Die Nachbarschaftsdefinition

Durch die Wahl des Verfahrens, das in der Nachbarschaft eines Lösungskandidaten nach neuen Lösungskandidaten sucht, wird implizit eine *Nachbarschaftsdefinition* getroffen, die die Qualität des Verfahrens ebenfalls signifikant beeinflussen kann: Die Wahl einer *kleinen* Nachbarschaft[56] kann z.B. verhindern, daß lokale Optima verlassen werden, wenn alle Lösungen in der Nachbarschaft der Ausgangslösung eine deutlich geringere Qualität als die Ausgangslösung selbst aufweisen[57].

Die Qualität des Verfahrens - und damit auch die Erfüllung der Anforderungen an ein Modul zur Initialisierungsplanung in Fertigungsleitständen - ist somit schlecht prognostizierbar und kann nur im Einzelfall - evtl. durch Vergleiche mit alternativen Verfahren - beurteilt werden.

---

handelt es sich vor allem um den Great Deluge Algorithm (Sintflutalgorithmus) und den Record-to-Record Travel, vgl. Dueck (1993).

55) Für die verwendete Heuristik in einem Szenario mit Betriebsmittelnalternativen, die sich auch durch unterschiedliche Leistungsgrade auszeichnen können, wurde ein Beweis, daß aus den Permutationen der Arbeitsgänge eine optimale Lösung überhaupt regelmäßig generiert werden kann, nicht geführt. (In einem entsprechenden Beweis wäre beispielsweise zu zeigen, daß mit der Heuristik *alle* aktiven Belegungspläne generiert werden können, vgl. auch Anmerkung 48).)

56) Eine *kleine Nachbarschaft* sei allgemein dadurch gekennzeichnet, daß nur geringe Änderungen vorgenommen werden. Die Bestimmung des Attributs *gering* ist dabei repräsentationsabhängig - es wird hier eine intuitive Vorstellung *geringer* Änderungen unterstellt. Im angeführten Beispiel zur Modifikation der Arbeitsgangliste kann das Ausmaß der Änderungen z.B. durch die Anzahl der Arbeitsgänge, deren Position verändert wurde (entspricht der Distanz, über die ein ausgewählter Arbeitsgang in der Liste verschoben wird), angegeben werden.

57) Dieser Effekt kann durch eine ungünstige Wahl der Temperatur und des Abkühlungsplanes verstärkt werden, da eine Nachbarschaft mit ausschließlich schlechteren Bewertungen nur über den durch die Temperatur involvierten probabilistischen Effekt verlassen werden kann. Zu kleine Temperaturwerte oder zu starkes Abkühlen verringert dann die Chance, einen derartigen Bereich verlassen zu können.

4.7 Exkurs: Parallelisierungskonzepte für das Simulated Annealing

Beim Simulated Annealing und dessen ´Derivaten´ handelt es sich um sequentielle Algorithmen[58]. Ihr Vorteil, für viele Problemstellungen einfache Lösungsansätze bereitzustellen, die bzgl. der Effizienz und der Lösungsqualität durchaus befriedigen können, scheint daher durch eine wachsende Verbreitung paralleler DV-Strukturen an Bedeutung zu verlieren. Um einen ´fairen´ Vergleich der Verfahren in einer solchen Umgebung zu ermöglichen, sollten auch Parallelisierungspotentiale für das Simulated Annealing untersucht und umgesetzt werden.

Eine Parallelisierung i.S. einer Zerlegung des Ausgangsproblems in mehrere kleinere Teilprobleme, deren getrennte Lösung sowie die abschließende Synthese der Einzellösungen zu einer Gesamtlösung scheidet für das Problem der Maschinenbelegungsplanung aus: Weder kann eine disjunkte Zerlegung des Arbeitsvorrats und die isolierte Einplanung der Teilvorräte durch einzelne, unabhängige Prozessoren vorgenommen werden, noch existiert eine solche Aufteilungsmöglichkeit für die Betriebsmittel[59].

Eine Möglichkeit der Parallelisierung bietet daher entweder die gleichzeitige Bearbeitung der gleichen Problemstellung durch mehrere Prozessoren oder die gleichzeitige Bearbeitung verschiedener, jedoch jeweils vollständiger Problemstellungen durch mehrere Prozessoren. Für die Parallelisierung der Maschinenbelegungsplanung im gegebenen Planungsumfeld können Alternativen solcher ´naiven´ Parallelisierungansätze auf Basis simultan arbeitender sequentieller Algorithmen formuliert werden, die

- eine gemeinsame Aufgabe bearbeiten,
- unterschiedliche Initialisierungen aufweisen und
- eingeschränkt miteinander kommunizieren.

Ausprägungen derartiger Parallelisierungsansätze sind:

a) Ein Auftragsbestand wird von mehreren Prozessoren mit unterschiedlichen Initialisierungen der auftragsunabhängigen Parameter eingeplant. Für das Simulated Annealing sind diese Parameter vor allem die Starttemperatur, der Abkühlungsplan (Schrittweiten der Temperaturabnahme) sowie die Werte für die Abbruchbedingungen. Die Wahl dieser initialen Parameter ist zwar in gewissen Grenzen unkri-

---

58) Vgl. z.B. Aarts/Korst (1989), p. 97.

59) Vgl. hierzu auch die Problematik der Blockierungen, die im Ansatz von Nakano/Yamada (1991) zur Einführung der globalen Harmonisierung führen (vgl. Kap.5.2.3).

tisch im Hinblick auf die Lösungsqualität, sie besitzt jedoch in jedem Fall einen starken Einfluß auf die Effizienz der Planung.

Durch Variation der Parameter auf verschiedenen Prozessoren kann insbesondere unterschiedlichen zeitlichen Anforderungen an die Ermittlung einer Lösung Rechnung getragen werden: Während einige Prozessoren qualitativ hochwertige Lösungen relativ langsam ermitteln, konvergieren andere sehr viel schneller unter der Gefahr, in suboptimalen Zuständen ´hängenzubleiben´.

b) Ein weiterer Ansatz verfolgt die (nicht notwendigerweise disjunkte) Aufteilung des Suchraums zwischen den Prozessoren: Betrachtet man die Funktion zur Bewertung der Qualität von Belegungsplänen als 'Energiegebirge', so wird die Suche nach dem Ziel (bei Minimierungsproblemen, etwa der Minimierung der Durchlaufzeit, das ´tiefste Tal des Gebirges´) von mehreren Prozessoren von unterschiedlichen Positionen innerhalb des Gebirges aus aufgenommen. Man kann erwarten, daß das Ziel im Durchschnitt umso schneller gefunden wird, je mehr Prozessoren danach von unterschiedlichen Positionen (Initialisierungsplänen) aus suchen.

   Dieser Ansatz unterscheidet sich von dem zuvor beschriebenen dadurch, daß alle Planungskomponenten mit denselben initialen Parametern, jedoch mit unterschiedlichen Initialisierungsplänen starten, während Ansatz a) identische Startpläne, jedoch unterschiedliche Initialisierungen auftragsunabhängiger Parameter verwendet.

c) Auch die Kombination der Ansätze a) und b) ist möglich, da vermutet werden kann, daß die ´Grenznutzen´ (i.S. der Zunahme von Effizienz und/oder Lösungsqualität) zusätzlich eingesetzter Prozessoren für beide Verfahren nicht identisch sind und außerdem relativ schnell abnehmen. Zum einen kann eine Obergrenze für die Anzahl der von einer Position (initialer Belegungsplan) aus startenden ´Spürhunde´ (Prozessoren) definiert werden, so daß ein Überschreiten dieser Grenze jeweils zur Wahl einer neuen Startpositionen führt. Zum anderen können nach einer ausreichenden Abdeckung des Suchraums zusätzliche Prozessoren zur Mehrfachbesetzung von Startpositionen (mit unterschiedlichen Parameter-Sets) dienen. Für die Ermittlung eines guten ´Mischungsverhältnisses´ können dabei die Vorgehensweisen der (statischen) a-priori-Festlegung der jeweiligen Prozessoranzahlen und alternativ der dynamischen Festlegung auf Basis abnehmender Nutzenzuwächse getestet werden. Darüber hinaus sollten einzelne Prozessoren von den Fortschritten anderer profitieren, beispielsweise falls sie in einem suboptimalen Zustand verharren, obwohl andere Prozessoren bereits bessere Lösungen ermittelt haben: In diesem Fall kann ein ´Ortswechsel´ erfolgen und eine Suche entweder mit den alten Parametern, mit gänzlich neuen Parametern oder (nur bei stochastischen

Suchstrategien sinnvoll) mit den Parametern des Prozessors, von dem die Position übernommen wurde, fortgeführt werden.

5 Ein klassischer genetischer Algorithmus

5.1 Der Problemlösungprozeß in genetischen Algorithmen

Evolutionsprogramme bilden Problemlösungsprozesse auf *evolutionäre Suchprozesse* ab. In einem wiederholten *Evaluations-/Selektions-/Rekombinations-Zyklus* wird aus einer Menge (*Population*) potentieller Lösungskandidaten (*Chromosomen*) jeweils eine Folgepopulation abgeleitet. Zunächst werden dabei die Elemente einer Population evaluiert, dann wird auf Basis dieser Bewertung eine Teilmenge der Population (*Mating Pool*) selektiert, aus der die Folgepopulation generiert wird. Dabei werden durch Rekombinationen der ausgewählten Elemente so lange neue Lösungskandidaten erzeugt, bis die gewünschte Mächtigkeit der Folgepopulation erreicht ist. Folgende Merkmale sind charakteristisch für evolutionäre Suchprozesse:

- Die Wahrscheinlichkeit, daß ein Lösungskandidat in den Mating Pool übernommen wird, hängt von seiner Bewertung (*Fitness)* ab *(´survival of the fittest´*).
- Durch die Rekombination wird ein *Informationsaustausch* zwischen unterschiedlichen Lösungskandidaten initiiert. Dabei können Lösungskandidaten sich ´kreuzen´ (*Crossover*), indem sie z.B. Teile ihrer Chromosomen austauschen.
- Der Lösungprozeß unterliegt *stochastischen Einflüssen* während der Selektion (Auswahl über Selektions*wahrscheinlichkeiten*) und der Rekombination (stochastische Auswahl der zu rekombinierenden Lösungskandidaten, Einführung eines *Mutationsoperators*, der zufällig ´kleine´ Änderungen an Lösungskandidaten vornimmt).

Die Grobstruktur eines genetischen Algorithmus gibt Abb. 9 wider. Generationen werden in der Darstellung durch den Parameter t gezählt.

Initialisiere:	Generationszähler Population Abbruchkriterium	(t ← 0) (P(0)) (AK)
Bewerte P(0)		
Wiederhole, solange AK nicht erfüllt		
	Erhöhe Generationszähler: t ← t+1	
	Selektiere und rekombiniere P(t) aus P(t-1)	
	Bewerte P(t)	

Abb. 9: Grobstruktur eines genetischen Algorithmus

5.2 Der klassische Ansatz von Nakano/Yamada

In *klassischen* genetischen Algorithmen werden Lösungskandidaten in Folgen binärer Werte (*Bitstrings*) kodiert und Problemlösungen unter Anwendung sogenannter Standard-Rekombinationsoperatoren abgeleitet[60]. Vorteil dieser Verfahren ist, daß zum einen das *´Fundamentaltheorem genetischer Algorithmen´*[61] als (rudimentärer) Erklärungsansatz für die Funktions- und Leistungsfähigkeit zur Verfügung steht und zum anderen durch die Verwendung der Standardoperatoren die Effizienz (nicht die Effektivität!) der Rekombination gewährleistet ist.

Der Ansatz von Nakano/Yamada[62] benutzt einen zweiteiligen *Repair-Algorithmus*, die sogenannte *Harmonisierung*, um - zunächst nur zum Zwecke der Bewertung - unzuläs-

---

60) In dieser Interpretation werden als klassische genetische Algorithmen somit die von John Holland (Holland (1975)) beschriebenen Algorithmen bezeichnet. Die Standard-Rekombinationsoperatoren sind die Mutation in Form eines einfachen ´Bitflips´ (eine ´0´ wird durch eine ´1´ ersetzt und umgekehrt) sowie das Crossover durch den Austausch von Teilstrings zwischen zwei Lösungskandidaten; vgl. zu klassischen genetischen Algorithmen z.B. Michalewicz (1992), pp. 13.

61) Das *Fundamentaltheorem* genetischer Algorithmen (nach Goldberg (1989), p. 36) als allgemeiner Erklärungsansatz für klassische genetische Algorithmen geht auf Holland (1975) zurück, und setzt sich aus dem sogenannten Schematheorem sowie der darauf aufbauenden sogenannten Baustein-Hypothese (Building Block Hypothesis) zusammen, vgl. z.B. Goldberg (1989), pp. 28. Ein *allgemeines Konvergenztheorem* für genetische Algorithmen stellt das Fundamentaltheorem allerdings nicht dar, vgl. z.B. Bierwirth (1993), S. 62f.

62) Vgl. Nakano/Yamada (1991). Die Autoren stellten 1992 auch einen hybriden Ansatz auf Basis einer symbolischen (nicht-binären) Repräsentation von Arbeitsgangreihenfolgen vor, der eine Heuristik zur Erzeugung der (aktiven) Pläne verwendet, vgl. Yamada/Nakano (1992).

sige in zulässige Pläne zu transformieren. Dabei werden in der *lokalen* Harmonisierung zuerst Inkonsistenzen auf Betriebsmittelebene beseitigt (vgl. Kap. 5.2.2). Danach wird in der *globalen* Harmonisierung ggf. die Konsistenz auf Ablaufplanebene hergestellt (vgl. Kap. 5.2.3). Durch das sogenannte *Forcing* wird darüber hinaus ein unzulässiger Plan durch den während der Harmonisierung abgeleiteten zulässigen Plan ersetzt, falls er den Selektionsprozeß ´überlebt´[63].

Der Ansatz wird im folgenden aus Anschaulichkeitsgründen in seiner ursprünglichen Form dargestellt, in der davon ausgegangen wird, daß jeder Auftrag *genau einmal* auf einem Betriebsmittel bearbeitet wird. Da diese - in Kap. 2.2 nicht aufgeführte - Prämisse bei den ´Vorgängeransätzen´ nicht aufgestellt wurde und die Verfahren auch mit Datenbeständen getestet wurden, die dieser Prämisse nicht entsprechen (vgl. Kap. 7.1), wurde der Ansatz in einer erweiterten Version realisiert, die die mehrmalige Bearbeitung eines Auftrages an einem Betriebsmittel unter Einhaltung fester Reihenfolgen für die entsprechenden Operationen erlaubt[64]. Die folgenden Ausführungen beschränken sich auf die Darstellung der Repräsentation von Lösungskandidaten sowie die Darstellung der Harmonisierung, die in einen klassischen genetischen Algorithmus als Bestandteil der Operation ´Bewertung´ in Abb. 9 eingebunden wird.

5.2.1 Die binäre Repräsentation

Maschinenbelegungspläne werden als Listen binärer Vorrangbeziehungen dargestellt. Zu diesem Zweck werden alle möglichen Vorrangbeziehungen zwischen Arbeitsgängen auf den einzelnen Betriebsmittel notiert. Die Vorrangfunktion wird definiert als

$$\textbf{vorrang}\ (a_{ij}, a_{kj}) \quad = \quad \begin{cases} 1, & \text{falls Auftrag i auf Betriebsmittel j vor Auftrag k bearbeitet wird,} \\ 0 & \text{sonst} \end{cases} \qquad (17)$$

für $i \neq k$;

a_{ij} : Bearbeitung von Auftrag i (A_i) auf Betriebsmittel j (B_j)

Maschinenbelegungspläne können anschaulich in Rangfolgenmatrizen dargestellt werden. Abb. 10 zeigt eine vollständige Rangfolgenmatrix exemplarisch für die Bearbeitung von

---

63) Um eine vorzeitige Konvergenz zu vermeiden, wenden Nakano/Yamada das Forcing nur auf das jeweils beste Element einer Population an, vgl. Nakano /Yamada (1991), p. 477.

64) Zu diesem Zweck können spezifische Vorrangbeziehungen zwischen Operationen (vgl. Kap. 5.2.1) dadurch fixiert werden, daß entsprechend gesetzte *Flags* die Unveränderlichkeit des Wertes einer Vorrangbeziehung anzeigt, vgl. Rohmann (1993), S. 37ff.

drei Aufträgen auf drei Betriebsmitteln[65]:

Betriebsmittel / Vorrang	B_1	B_2	B_3
A_1_vor_A_2	1	1	0
A_1_vor_A_3	0	1	1
A_2_vor_A_3	1	0	0

Abb. 10: Rangfolgematrix

Für die chromosomale Darstellung als Bitstring werden die Spalten der Matrix (wie in Abb. 10 durch Pfeile angedeutet) ´aneinandergehängt´. Die Rangfolgenmatrix aus Abb. 10 entspricht somit dem Chromosom [1 0 1 1 1 0 0 1 0].

5.2.2 Die lokale Harmonisierung

Die Beseitigung von Inkonsistenzen auf Betriebsmittelebene wird anhand eines Beispiels demonstriert. Gegeben sei der durch die Rangfolgematrix in Abb. 10 determinierte Plan. Die Interpretation der Rangfolgen auf Betriebsmittel 1 ergibt:

1. Auftrag 1 wird vor Auftrag 2 ausgeführt (A_1_vor_$A_2 = 1$)

   und

2. Auftrag 2 wird vor Auftrag 3 ausgeführt (A_2_vor_$A_3 = 1$)

   aber

3. Auftrag 1 wird *nicht* vor Auftrag 3 ausgeführt (A_1_vor_$A_3 = 0$)

⇒ **Widerspruch!**

Ziel der lokalen Harmonisierung ist es, die Inkonsistenz durch eine Überführung in eine zulässige Lösung mit minimaler Hamming-Distanz[66] (bzgl. der binären Darstellung) zu

---

65) In der Darstellung entspricht der Wert in Zeile ´A_i_vor_A_j´ und Spalte ´B_k´ dem Wert der Vorrangbeziehung vorrang (a_{ik}, a_{jk}).

66) Die Hamming-Distanz gibt die Anzahl der Positionen an, in denen sich zwei Bit-Vektoren der gleichen Dimension unterscheiden.

beseitigen. Zu diesem Zweck werden die Vorrangbeziehungen für jeweils ein Betriebsmittel als Matrix dargestellt, in der jede Zeile die Vorrangbeziehungen eines Auftrages zu *allen* anderen Aufträgen enthält[67]. In der Matrix wird die Summe der Vorränge eines Auftrages für das betreffende Betriebsmittel jeweils als Zeilensumme berechnet. Für die Beziehungen aus Abb. 10 ergeben sich beispielsweise für Betriebsmittel 1[68]:

	A_1	A_2	A_3	Σ
A_1	*	1	0	1
A_2	0	*	1	1
A_3	1	0	*	1

Abb. 11: Vorrangmatrix für Betriebsmittel 1

Im nächsten Schritt wird der Auftrag (hier als Zeile) mit der maximalen Vorrangssumme gewählt. Im Beispiel weisen alle Aufträge eine Vorrangssumme von 1 auf - der Konflikt wird durch die zufällige Auswahl eines Auftrages, hier des Auftrages 1, gelöst und anschließend die Vorrangssumme des gewählten Auftrages maximiert[69]. In Abb. 12 erhält Auftrag 1 durch einen entsprechenden Bitflip zu dem Vorrang vor Auftrag 2 zusätzlich auch den Vorrang vor Auftrag 3, so daß seine (maximale) Vorrangssumme nun 2 beträgt (zu beachten ist, daß in der redundanten Darstellungsweise jeweils 2 Bits geändert werden müssen).

	A_1	A_2	A_3	Σ	
A_1	*	1	1	2	⇐
A_2	0	*	1	1	
A_3	0	0	*	0	

Abb. 12: Maximierung der Vorrangsumme für Auftrag 1

67) Diese Darstellung (die so nicht implementiert wird) führt - im Gegensatz zu der Darstellung in Abb. 10 - zu einer Redundanz, da für zwei Aufträge x und y in der Matrix sowohl die Vorrangbeziehung A_x_vor_A_y als auch die Vorrangbeziehung A_y_vor_A_x abgebildet werden.

68) Für die mit einem Stern (´*´) gekennzeichneten Felder ist die Vorrangbeziehung nicht definiert.

69) Im Falle der Bearbeitung von n Aufträgen auf einem Betriebsmittel kann ein Auftrag maximal vor (n-1) Aufträgen den Vorrang erhalten.

Der Vorgang wird ggf. für die verbleibenden Aufträge so lange wiederholt, bis jedem Rang eindeutig ein Auftrag zugeordnet ist (im Beispiel ist dies der Fall)[70].

Als Ergebnis der lokalen Harmonisierung liegen - für einzelne Betriebsmittel isoliert betrachtet - konsistente Auftragsfolgen vor[71]. In Verbindung mit gegebenen Maschinenfolgen der Aufträge können sich diese Reihenfolgen jedoch bei der Integration zu Belegungsplänen gegenseitig blockieren. Derartige Blockierungen werden anschließend durch die globale Harmonisierung beseitigt.

5.2.3 Die globale Harmonisierung

In der globalen Harmonisierung wird versucht, einen Maschinenbelegungsplan gemäß der Auftragsfolgen aufzubauen, die während der lokalen Harmonisierung festgelegt wurden. Die globale Harmonisierung bedient sich dabei dreier Funktionen:

next_op_of_job (i) (18)

- liefert den nächsten Fertigungsarbeitsgang eines Auftrages i (gemäß eines gegebenen Arbeitsplanes);

mach_of_next_op_of_job (i) (19)

- liefert das Betriebsmittel, an dem der nächste Fertigungsarbeitsgang eines Auftrages i bearbeitet werden soll (gemäß der gegebenen Maschinenfolge);

next_op_of_mach (j) (20)

- liefert den nächsten Fertigungsarbeitsgang, der auf Betriebsmittel j bearbeitet werden soll (gemäß der geplanten Auftragsfolge);

Es wird eine *Einplanungsbedingung* EB angegeben, nach der ein Fertigungsarbeitsgang auf einem Betriebsmittel eingeplant werden kann, falls

- er der als nächstes auszuführende Arbeitsgang eines Auftrages i ist und
- der Auftrag i als nächster Auftrag an einem Betriebmittel j bearbeitet werden soll.

Mittels der angeführten Funktionen kann die Einplanungsbedingung EB formalisiert werden. Ein Fertigungsarbeitsgang ist einplanbar, falls für einen Auftrag i gilt:

next_op_of_job (i) = next_op_of_mach (mach_of_next_op_of_job (i)) (21)

---

70) Die Vorrangbeziehungen zu ´höherrangigen´ Aufträgen werden nicht wieder ´zurückgeändert´, so daß sich der Wert des maximal erreichbaren Ranges in jeder Runde um 1 verringert.

71) Vgl. Nakano/Yamada (1991), p. 477.

Mit Hilfe der Einplanungsbedingung wird ein einfacher Einplanungsalgorithmus formuliert, dessen Grobstruktur in Abb. 13 dargestellt ist[72].

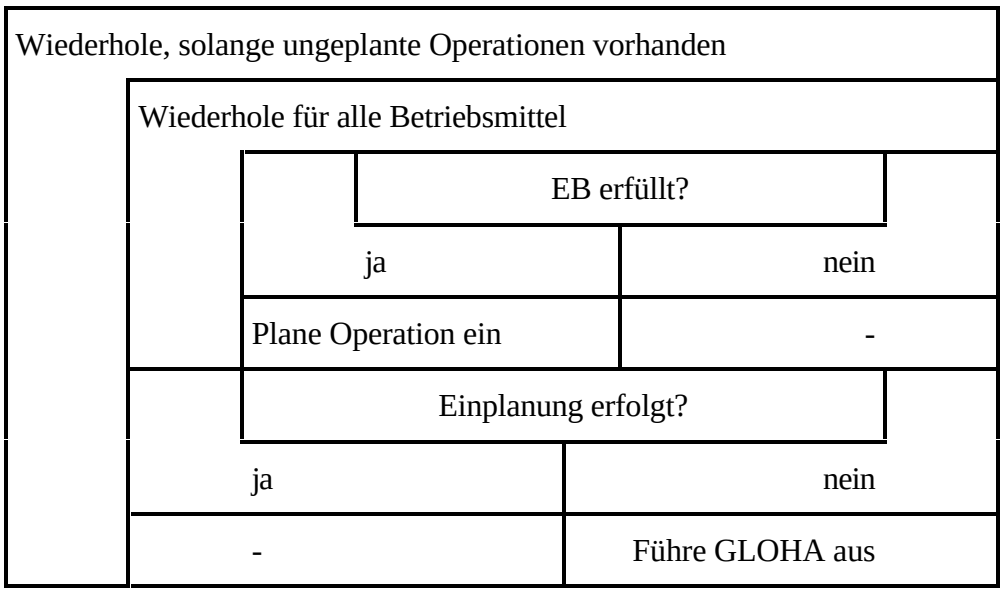

Abb. 13: Grobstruktur des Einplanungsalgorithmus

Der Algorithmus der globalen Harmonisierung basiert auf der Idee, im Falle einer Blokkierung (keine einplanbaren Fertigungsarbeitsgänge, obwohl noch ungeplante Fertigungsarbeitsgänge vorhanden sind) die Reihenfolge auf einem Betriebsmittel zu ändern, indem eine Operation vorgezogen wird. Mit einer Funktion

distance (A_i, B_j),

die für die einen Auftrag i, der noch auf einem Betriebsmittel j bearbeitet werden muß, angibt, wieviele Fertigungsarbeitsgänge anderer Aufträge auf diesem Betriebsmittel noch vorher bearbeitet werden sollen (laut geplanter Auftragsfolge des Betriebsmittels), kann die Grobstruktur der globalen Harmonisierung wie folgt angegeben werden (Abb. 14):

72) Das Akronym GLOHA steht in Abb. 13 für *globale Harmonisierung.*

Wiederhole für alle Aufträge i	
	Wähle nächstes Betriebsmittel j laut Maschinenfolge
	Berechne distance (A_i, B_j)
Bestimme $D_{min} = min_{i,j}$ (distance (A_i, B_j))	
Wähle B_{j*}, für das D_{min} ermittelt wurde[72)]	
Verschiebe die D_{min} ersten Aufträge der Auftragfolge von B_{i*} um eine Position nach hinten	
Plane A_{i*}, für den D_{min} ermittelt wurde, auf B_{j*} ein[73)]	

Abb. 14: Grobstruktur der globalen Harmonisierung

Die skizzierte Vorgehensweise wird nachfolgend anhand eines Beispieles demonstriert: Einzuplanen seien 3 Aufträge auf 3 Betriebsmitteln, die lokale Harmonisierung sei bereits durchgeführt worden. In Abb. 15 sind die gegebenen Maschinenfolgen sowie ein Lösungskandidat (Chromosom) mit den resultierenden (geplanten) Auftragsfolgen angegeben:

Chromosom: [1 1 1 0 0 1 1 0 0]

⇒

	Position		
	1.	2.	3.
A_1	B_3	B_1	B_2
A_2	B_2	B_3	B_1
A_3	B_1	B_2	B_3

	Auftragsfolge		
B_1	A_1	A_2	A_3
B_2	A_2	A_3	A_1
B_3	A_3	A_1	A_2

Abb. 15: Gegebene Maschinenfolgen und geplante Auftragsfolgen

Eine Einplanung gemäß des Algorithmus in Abb. 13 ergibt (vgl. Abb. 16):

1. Schleife (1. Spalte der Einplanung in Abb. 16 a)):
   - Auf Betriebsmittel 1 soll Auftrag 1 bearbeitet werden, Auftrag 1 muß jedoch

73) Nakano/Yamada geben keine ´*tie breaking rule*´ für den Fall an, daß es mehrere Betriebsmittel und/oder mehrere Aufträge gibt, für die eine minimale Distanz bestimmt wurde. In der realisierten Implementierung werden Konflikte durch zufällige Auswahl gelöst.

zunächst auf Betriebsmittel 3 bearbeitet werden (EB nicht erfüllt) ⇒ keine Einplanung.

- Auf Betriebsmittel 2 soll Auftrag 2 bearbeitet werden und Auftrag 2 muß auch als nächstes auf Betriebsmittel 2 bearbeitet werden (EB erfüllt) ⇒ Einplanung (und Streichung des eingeplanten Arbeitsganges aus der Maschinen- und der Auftragsfolge).
- Auf Betriebsmittel 3 soll Auftrag 3 bearbeitet werden, Auftrag 3 muß jedoch zunächst auf Betriebsmittel 1 bearbeitet werden (EB nicht erfüllt) ⇒ keine Einplanung.

a) Einplanung:

B_1	-	-
B_2	A_2	-
B_3	-	-

⇓

Deadlock!

⇒

b) Distanzen:

$A_1(B_3)$	1 ←
$A_2(B_3)$	2
$A_3(B_1)$	2

c) neue Auftragsfolge:

	Auftragsfolge		
B_1	A_1	A_2	A_3
B_2	A_2	A_3	A_1
B_3	A_1	A_3	A_2

⇒

d) neues Chromosom:

[1 1 1 0 0 1 1 1 0]

Abb. 16: Einplanungsversuch, Distanzen und neue Auftragsfolge

2. Schleife (2. Spalte der Einplanung in Abb. 16 a)):

- Auf keinem Betriebsmittel kann ein Ferigungsarbeitsgang eingeplant werden (Deadlock) ⇒ globale Harmonisierung wird ausgeführt.

Globale Harmonisierung:

- Die Distanzen distance (A_i, B_j) werden für alle Aufträge an denjenigen Betriebsmitteln berechnet, an denen die Aufträge laut Maschinenfolge jeweils als nächstes zu bearbeiten sind (A_1 auf B_3, A_2 auf B_3, A_3 auf B_1, vgl. Abb. 16 b)).
- Auftrag 1 weist den minimalen Distanzwert 1 (auf Betriebsmittel 3) auf.
- In der Auftragsfolge von Betriebsmittel 3 wird ein Auftrag (Auftrag 3) um eine Position verschoben (vgl. Abb. 16 c); das entsprechende Chromosom ist in Abb. 16 d) angegeben[74]).

Fortführung der Einplanung bei Betriebsmittel 1.

Anmerkung: Die Einplanung im Beispiel ist noch immer unvollständig und erfordert die wiederholte Anwendung der globalen Harmonisierung!

5.3 Kritik

Kritik am dargestellten klassischen genetischen Algorithmus zur Maschinenbelgungsplanung resultiert vorrangig aus der verwendeten binären Repräsentationsform. Sie weist den Nachteil auf, daß der über der Binär-Repräsentation definierte *Suchraum* erheblich größer als der *Raum zulässiger Lösungen* ist. Durch die Verwendung ´blinder´ Operatoren[75]), mit deren Hilfe der Suchraum ´ziellos´ durchforstet wird, werden während des Rekombinationsprozesses regelmäßig unzulässige Lösungen erzeugt, die durch die dargestellte Harmonisierung in zulässige Lösungen transformiert werden müssen[76]). Ein Beispiel verdeutlicht diese Problematik:

Gegeben seien n Aufträge, die auf m Betriebsmitteln zu bearbeiten sind. Für den Raum, der vom genetischen Algorithmus durchsucht wird (Suchraum) können folgende Ab-

---

74) Auch dieser Lösungskandidat ist noch nicht zulässig, die globale Harmonisierung müßte für eine erfolgreiche Einplanung wiederholt durchgeführt werden.

75) Als *blind* werden Operatoren bezeichnet, die nicht auf problemspezifisches Wissen zurückgreifen und somit - isoliert betrachtet - auch nicht zielgerichtet arbeiten. Der Effekt ihrer Anwendung wird i.d.R. erst ex post beurteilt.

76) Die Problematik ähnelt der des ersten Ansatzes auf Basis eines Hopfield-Netzwerkes (vgl. Kap. 3.3.3). Grundsätzlich besteht bei der Anwendung genetischer Algorithmen auch die zusätzliche Option des ´Aussortierens´ unzulässiger Lösungen (Zulässigkeitsfilter). Aufgrund der Multidirektionalität der Suche können unzulässige Lösungen einfach ´unter den Tisch fallen´ - zumindest solange noch (mehrere) zulässige Lösungen verbleiben, mit denen der Evaluations-/Selektions-/Rekombinations-Zyklus fortgeführt werden kann.

schätzungen angegeben werden:

- pro Auftrag und Betriebsmittel existieren n-1 Vorrangbeziehungen
- *vorrang(A,B)* determiniert *vorrang(B,A)*

$$\Rightarrow \quad \frac{n \cdot (n-1)}{2} \qquad \text{binäre Elemente pro Betriebsmittel} \qquad (22)$$

$$\Rightarrow \quad m \cdot \frac{n \cdot (n-1)}{2} \qquad \text{binäre Elemente insgesamt} \qquad (23)$$

$$\Rightarrow \quad 2^{\left(m \cdot \frac{n \cdot (n-1)}{2} \right)} \qquad \text{Lösungskandidaten im Suchraum} \qquad (24)$$

Der Raum der zulässigen Lösungen hingegen kann - ohne Berücksichtigung jeglicher Nebenbedingungen - wie folgt abgeschätzt werden:

- pro Betriebsmittel sind n! Auftragsfolgen zulässig

$$\Rightarrow \quad (n!)^m \text{ Auftragsfolgen insgesamt} \qquad (25)$$

Eine Beispielrechnung für quadratische Maschinenbelegungsprobleme (n=m) zeigt, daß der Faktor, um den der Suchraum größer als der Lösungsraum ist, bereits bei kleinen Problemumfängen ´explodiert´ (vgl. Abb. 17)[77].

m, n	2	3	4	5
realisierbare Lösungen	4	216	331 776	$2{,}49 \cdot 10^{10}$
repräsentierbare Lösungen	4	512	16 777 216	$1{,}13 \cdot 10^{15}$
Faktor	1	≈ 2	≈ 51	≈ 45 247

Abb. 17: Das Verhältnis realisierbarer zu repräsentierbaren Lösungen

77) Bereits für n = m = 10 gilt, daß der *Faktor* größer ist als die *Anzahl* zulässiger Lösungen (ohne Berücksichtigung von Nebenbedingungen!), d.h. die Größe des Suchraumes ist größer als die Quadratzahl der Größe des Lösungsraumes!

6 Implementierung und Test der Verfahren

6.1 Entwicklungs- und Einsatzumgebung

Der Ansatz auf Basis des Simulated Annealing wurde ursprünglich auf einem Transputer-System (Parsytec Multicluster/2) in C implementiert. Nach erfolgtem Test wurde eine lose Kopplung mit dem grafischen elektronischen Fertigungsleitstand L1 (unter OSF-Motif) über einen einfachen Datentranfer in einem Ethernet-LAN realisiert (automatisierter Dateitransfer über Remote-Shell-Scripts mit SQL, SQL*Loader)[78]. Die Kopplungsarchitektur veranschaulicht Abb. 18. Bei der Visualisierungskomponente in der Abbildung handelt es sich um eine grafische Benutzerschnittstelle, die mit dem Visualisierungswerkzeug DataViews erstellt wurde und der Initialisierung sowie der Auswertung der Testläufe *vor* der Anbindung an den Fertigungsleitstand diente.

In einer zweiten Stufe wurde das Modul als Feinterminierungsmodul direkt in einen objektorientierten grafischen Fertigungsleitstand[79] auf einer Workstation (NeXTCube unter Mach) eingebunden: Der Datenaustausch erfolgt über die Datenbank des Fertigungsleitstands (Sybase): Das Planungsmodul liest auf Anforderung grobgeplante Daten aus der Datenbank, plant diese zeitgenau ein und schreibt die feingeplanten Daten zurück, die anschließend in der Plantafel des Fertigungsleitstands dargestellt und weiterverarbeitet werden können. Die Schnittstellen zur Datenbank wurden dabei als C-Routinen (mit embedded SQL) implementiert und direkt in die Fertigungsleitstandfunktionalität integriert.

Da auf eine Parallelverarbeitung verzichtet wurde und die Implementierung auf einem Transputercluster somit keinerlei Vorteile bot, erfolgte die Realisierung des genetischen Feinplanungsmoduls direkt auf einer leistungsfähigeren dedizierten Workstation ebenfalls in C. Die zuvor bereits gewählten Vorgehensweise des isolierten Tests und sich daran anschließender Einbindung in den Leitstand wurde beibehalten. Da bei der Implementierung auf strikte Einhaltung des C-ANSI-Standards geachtet wurde, bereitete die Portierung der entwickelten Module auf andere Plattformen kaum Schwierigkeiten. Um Laufzeitvergleiche zu ermöglichen, wurden die Testläufe auf einem PC (80486, 25 MHz) durchgeführt, da ein Werkzeug zur analytischen Lösung der Testprobleme (s.u.) nur auf diesem System zur Verfügung stand.

---

78) Vgl. Ruppel/Siedentopf (1992), S. 559.

79) Vgl. Nietsch u.a. (1992), Kurbel u.a. (1992).

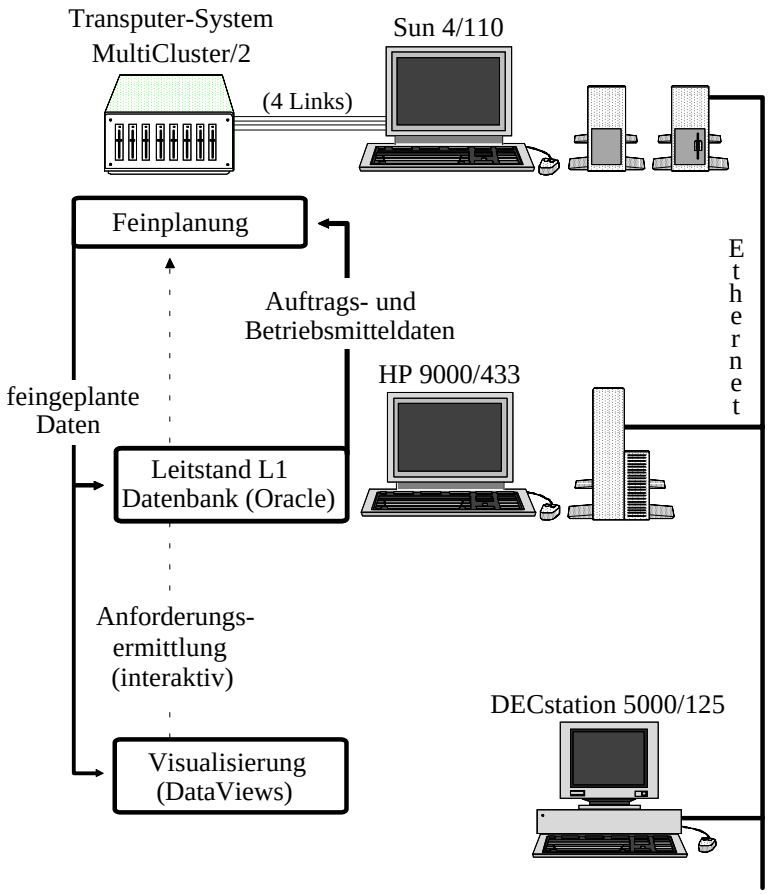

Abb. 18: Kopplungsarchitektur (lose Kopplung)

6.2 Test

Von den skizzierten Ansätzen wurden das Verfahren des Simulated Annealing sowie der genetische Algorithmus nach Nakano/Yamada (1991) implementiert und vergleichenden Tests unterzogen. Zusätzlich wurde zum Laufzeitvergleich ein ´klassisches´ Verfahren der gemischt-ganzzahligen Optimierung[80)] herangezogen, das gleichzeitig einen Zugriff auf die jeweiligen Optimallösungen ermöglichte (Kap. 6.2.2.1). Die Lösungsqualität

80) Die Berechnungen wurden mit dem Tool OMP durchgeführt. OMP ist ein unter verschiedenen Plattformen (MS-DOS, OS/2, AIX, HP-UX, VMS) verfügbares Tool zur linearen und (gemischt-) ganzzahligen Programmierung. Lineare Programme werden mittels der revidierten Simplexmethode, (gemischt-) ganzzahlige Programme durch ein Branch&Bound-Verfahren (Relaxationen wiederum durch die revidierte Simplexmethode) gelöst. OMP bietet die Möglichkeit, große Modelle mit Hilfe eines Modell-Generators kompakt zu formulieren und Lösungsstrategien über spezifische Parameter zu beinflussen (z.B. Richtung des Branching, Prioritäten für Constraints etc.). Die für die Tests zur Verfügung stehende Version Midsize 4.07 unter MS-DOS löst Probleme bis zu einer Komplexität von 5000 Variablen und 2500 Nebenbedingungen.

wurde darüber hinaus Ergebnissen gegenübergestellt, die mit (vier) Prioritätsregelverfahren erzielt wurden (Kap. 6.2.2.2). Die Darstellungen beschränken sich auf einige ausgewählte, stark aggregierte Ergebnisse.[81)]

6.2.1 Die Testdatenbasis

Die beiden ausgewählten Verfahren wurden anhand zweier unterschiedlich strukturierter Datenbestände verglichen: Zum einen handelt es sich um ein künstlich erzeugtes, symmetrisches (quadratisches) Job-Shop-Scheduling-Problem, das sich in der Literatur als Testproblem etabliert hat[82)]. Die Auftragsdaten umfassen 10 Fertigungsaufträge mit je 10 Arbeitsgängen (Operationen) und sind in Abb. 19 dargestellt.

	Auftrag																			
	1		2		3		4		5		6		7		8		9		10	
AG	BM	BZ	BM	BZ	BM	BZ	BM	BZ	BM	BZ	BM	BZ	BM	BZ	BM	BZ	BM	BZ	BM	BZ
1	1	29	1	43	2	91	2	81	3	14	3	84	2	46	3	31	1	76	2	85
2	2	78	3	90	1	85	3	95	1	6	2	2	1	37	1	86	2	69	1	13
3	3	9	5	75	4	39	1	71	2	22	6	52	4	61	2	46	4	76	3	61
4	4	36	10	11	3	74	5	99	6	61	4	95	3	13	6	74	6	51	7	7
5	5	49	4	69	9	90	7	9	4	26	9	48	7	32	5	32	3	85	9	64
6	6	11	2	28	6	10	9	52	5	69	10	72	6	21	7	88	10	11	10	76
7	7	62	7	46	8	12	8	85	9	21	1	47	10	32	9	19	7	40	6	47
8	8	56	6	46	7	89	4	98	8	49	7	65	9	89	10	48	8	89	4	52
9	9	44	8	72	10	45	10	22	10	72	5	6	8	30	8	36	5	26	5	90
10	10	21	9	30	5	33	6	43	7	53	8	25	5	55	4	79	9	74	8	45

(AG: Arbeitsgangnummer, BM: Betriebsmittelnummer, BZ: Bearbeitungszeit)

Abb. 19: Testdaten nach Fisher/Thompson

Zum anderen wurden Testdatenbestände aus einem Produktionsplanungs- und -steuerungssystem (PPS-System) verwendet, die aus einem realen Fertigungsbetrieb stammen und eine (im Vergleich zu den ´künstlichen´ Testdaten) hohe Inhomogenität bzgl. der Größe und der Ressourcenbeanspruchung einzelner Fertigungsaufträge aufweisen. Die

---

81) Detailliertere Darstellungen der Durchführung und der Ergebnisse der skizzierten Vergleichstests findet sich bei Rohmann (1993) und bei Kurbel (1994).

82) Fisher/Thompson (1963), p. 236.

Daten umfassen 10 Fertigungsaufträge, die 2 bis 29 Fertigungsarbeitsgänge enthalten und auf 18 Betriebsmitteln bearbeitet werden (asymmetrisches Scheduling-Problem, vgl. Abb. 20).

	Auftrag																			
	1		2		3		4		5		6		7		8		9		10	
AG	BM	BZ	BM	BZ	BM	BZ	BM	BZ	BM	BZ	BM	BZ	BM	BZ	BM	BZ	BM	BZ	BM	BZ
1	1	30	1	60	1	30	1	48	6	45	1	30	6	45	6	45	1	30	10	50
2	1	30	1	30	1	30	1	48	8	33	1	30	8	33	8	33	7	35	10	50
3			7	35	7	35	7	35	8	33	7	35	8	33	8	33	9	35	10	50
4			9	35	9	35	9	35	8	40	9	35	17	40	17	40	14	35	10	50
5			14	35	14	35	14	35			14	35	10	50	10	50	15	35		
6			9	35	15	35	15	35			15	35	10	50	10	50	12	40		
7					12	40	12	40			12	40	10	50	10	50	2	50		
8					12	40	12	40			12	40	10	50	10	50	3	55		
9					12	42	12	40			12	40			11	20	4	55		
10							2	50			2	50			11	20	5	55		
11							2	50			2	50			11	20	13	55		
12							3	55			3	55					16	55		
13							3	55			3	55					18	60		
14							4	55			4	55					6	45		
15							5	55			5	55					8	33		
16							13	55			13	55					17	40		
17							16	55			16	55					10	50		
18							18	60			18	30					11	20		
19											6	45								
20											8	33								
21											8	33								
22											17	40								
23											10	50								
24											10	50								
25											10	50								
26											10	50								
27											11	20								
28											11	20								
29											11	20								
∑	2		6		9		18		4		29		8		11		18		4	

(AG: Arbeitsgangnummer, BM: Betriebsmittelnummer, BZ: Bearbeitungszeit)

Abb. 20: Testdaten aus einem PPS-System

Um zum einen die Datenbasis zu ´verbreitern´ und zum anderen die Laufzeit der Testläufe zu verringern[83)], wurden die beiden gewählten Testdatenbestände in *Szenarien* ge

83) Die Komplexität der gewählten (und gemessen an praktischen Datenvolumina eher noch kleineren) Problemstellungen veranschaulicht die Tatsache, daß die Optimallösung (analytisch ermittelt, inkl.

splittet, deren Aufbau Abb. 21 und Abb. 22 verdeutlichen.

SNR	FA	SNR	FA	SNR	FA	SNR	FA	SNR	FA
1	1,2,3	11	2,3,4,5	21	2,3,4,6	31	3,4,5,8,9	41	4,5,6
2	1,2,3,4	12	3,4,5,6	22	2,4,5,6	32	1,6,7,8	42	5,6,7
3	1,2,3,4,6	13	4,5,6,7	23	2,6,7,8	33	1,6,8,9	43	6,7,8
4	1,2,3,10	14	5,6,7,8	24	2,6,9,10	34	1,6,9,10	44	7,8,9
5	1,2,9,10	15	6,7,8,9	25	1,4,7,8,10	35	1,6,7,8,10	45	8,9,10
6	3,4,8,9	16	7,8,9,10	26	5,6,7,8	36	4,5,6,7,8	46	1,3,6
7	4,8,9,10	17	3,4,6,7	27	5,7,8,9	37	5,6,7,8,9	47	3,6,7
8	9,10	18	3,4,7,9	28	1,3,8,10	38	6,7,8,9,10	48	6,7,10
9	1,2,7,8	19	1,4,5,6	29	5,8,9,10	39	2,3,4	49	1,7,9
10	6,7,8,9,10	20	1,5,8,10	30	3,4,5,6,7	40	3,4,5	50	4,6,7

(SNR: Szenarionummer, FA: Fertigungsaufträge)

Abb. 21: Die 50 Szenarien des Datenbestandes nach Fisher/Thompson

SNR	FA	FAG
1	1, 10	6
2	1, 5, 10	10
3	1, 2, 5	12
4	3, 7	17
5	1, 5, 7, 8	25
6	4, 5, 10	26
7	3, 7, 8	28
8	6, 10	33
9	7, 8, 9	37

SNR	FA	FAG
10	6, 7, 10	41
11	4, 6	47
12	1, 2, 6, 7, 10	49
13	1, 3, 6, 7, 10	52
14	3, 6, 8, 10	53
15	1, 2, 3, 6, 8	57
16	1, 2, 3, 4, 5, 7, 8, 10	62
17	3, 4, 7, 8, 9	64

(SNR: Szenarionummer, FA: Fertigungsaufträge, FAG: Anzahl Fertigungsarbeitsgänge)

Abb. 22: Die 17 Szenarien des PPS-Datenbestandes

Zusammenfassend wurden die Tests damit über 9 Szenarien à 50 Fertigungsarbeitsgänge, 27 Szenarien à 40 Fertigungsarbeitsgänge, 13 Szenarien à 30 Fertigungsarbeitsgänge und 1 Szenario mit 20 Fertigungsarbeitsgängen aus dem Testdatenbestand von Fisher/-Thompson sowie über 17 Szenarien mit 6 bis 64 Fertigungsarbeitsgängen aus dem PPS-Datenbestand durchgeführt.

Optimalitätsbeweis) für das vollständige Testproblem von Fisher/Thompson erstmals in Carlier/ Pinson (1989) dokumentiert wurde.

6.2.2 Ausgewählte Testergebnisse

6.2.2.1 Das Laufzeitverhalten

Einen Vergleich des Laufzeitverhalten der drei getesteten Verfahren für die Testprobleme von Fisher/Thompson faßt Abb. 23 zusammen. Das Akronym SA kennzeichnet die Ergebnisse für das Verfahren auf Basis des Simulated Annealing, das Akronym MIP (M*ixed* I*nteger* P*rogramming*) die Ergebnisse mit OMP sowie das Akronym GA die Ergebnisse des genetischen Algorithmus nach Nakano/Yamada (1991)[84].

Die Abbildung ´suggeriert´ für die gemischt-ganzzahlige Programmierung das erwartete exponentielle Wachstum der Laufzeit in Abhängigkeit von der Anzahl einzuplanender Arbeitsgänge (als Maß für die Problemgröße) und für den genetischen Algorithmus ein ähnliches, über dem abgebildeten Bereich bei höherem Ausgangsniveau allerdings schwächer ausgeprägtes Wachstum[85]. Für das Simulated Annealing ist ein Zusammenhang zwischen Laufzeit und Problemgröße aus der Abbildung nicht zu entnehmen.

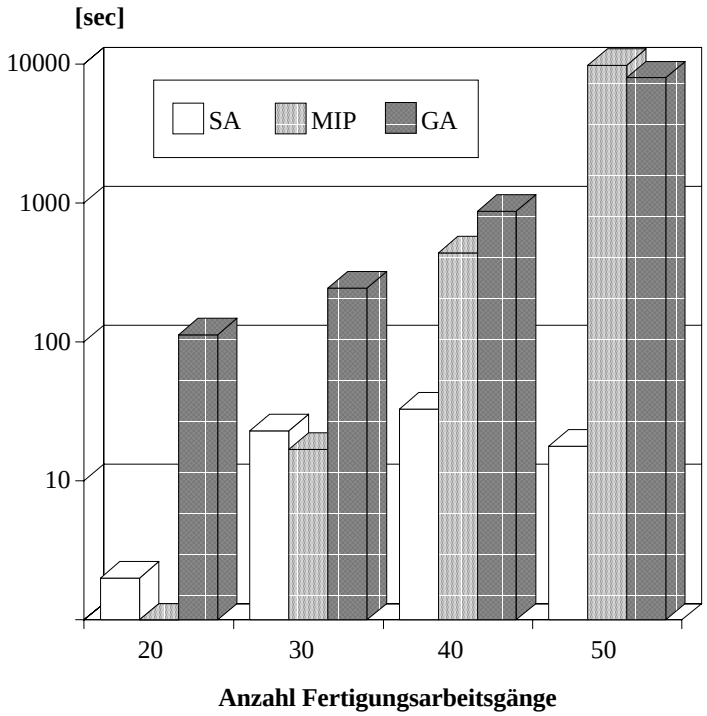

Abb. 23: Laufzeitvergleich (Daten von Fisher/Thompson)

84) Die Akronyme werden mit dieser Bedeutung auch in den folgenden Ausführungen verwendet.

85) Die Laufzeit wird als Durchschnittswert aller Testläufe über den jeweiligen Szenarien auf einer logarithmischen Skala abgebildet. Zu beachten ist, daß die Anzahl der Arbeitsgänge nur *eine* Einflußgröße der Problemkomplexität darstellt, die darüber hinaus in den unterschiedlichen Modellen unterschiedlich wirkt.

Das Laufzeitverhalten für die Testprobleme des PPS-Datenbestandes gibt Abb. 24 wieder. Bemerkenswert ist, daß sogar für das Verfahren der gemischt-ganzzahligen Programmierung eine Abhängigkeit der Laufzeit vom (in der Anzahl der einzuplanenden Arbeitsgänge gemessenen) Problemumfang nicht mehr festgestellt werden kann.

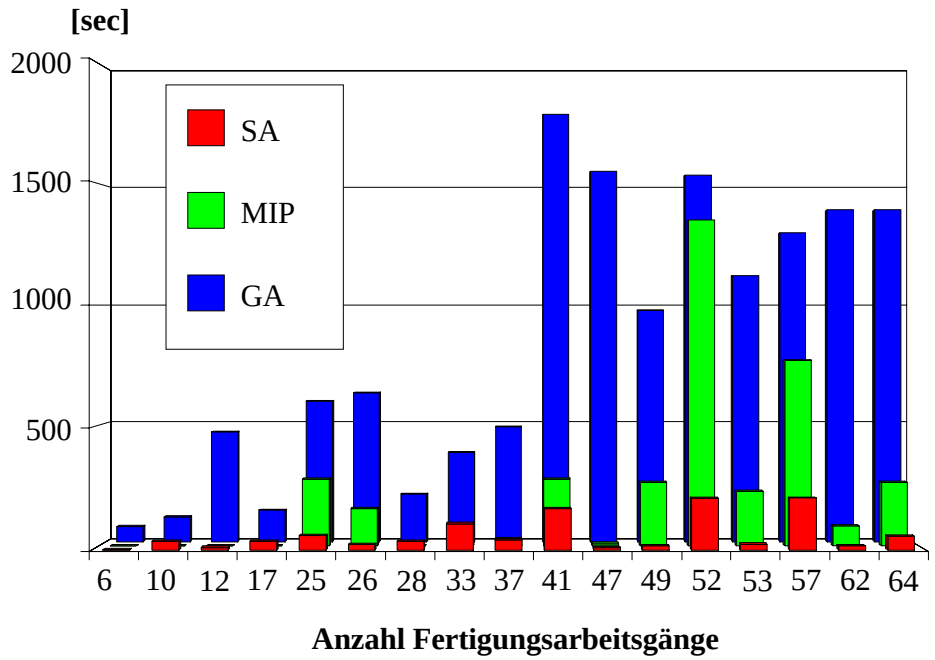

Abb. 24: Laufzeitvergleich (PPS-Datenbestand)

6.2.2.2 Die Lösungsqualität

Die Qualität der Lösungen, die vom genetischen Algorithmus und vom Simulated-Annealing-Verfahren erzeugt werden, stellt Abb. 25 gegenüber:

		Nicht-optimale Lösungen					
Anzahl Arbeitsgänge	Anzahl Szenarien	absolut		prozentual [%]		durchschnittliche Abweichung vom Optimum [%]	
		SA	GA	SA	GA	SA	GA
20	1	0	0	0	0	0,00	0,00
30	13	0	0	0	0	0,00	0,00
40	27	7	5	26	19	0,75	0,20
50	9	8	7	89	78	3,10	0,96

Abb. 25: Qualität von SA und GA (Daten von Fisher/Thompson)

Die Ergebnisse eines Vergleiches des genetischen Algorithmus mit ausgewählten Prioritätsregelverfahren enthält Abb. 26. Dargestellt sind die durchschnittlichen Abweichungen der Lösungsqualität der Prioritätsregelverfahren von den Lösungen des genetischen Algorithmus für die Szenarien mit jeweils 50 Arbeitsgängen des Datenbestands von Fisher/-Thompson[86]. Bei den Vergleichsverfahren handelt es sich um

- die kürzeste-Restbearbeitungszeit-Regel (KRB)
- die längste-Restbearbeitungszeit-Regel (GRB)
- die kürzeste-Operationszeit-Regel (KOZ)
- die längste-Operationszeit-Regel (LOZ)

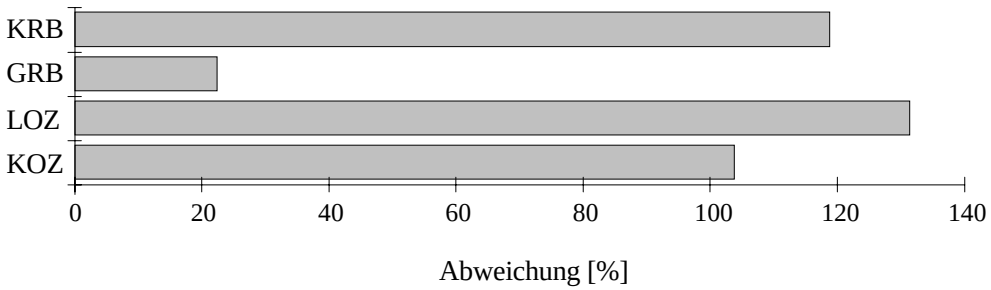

Abb. 26: Qualität von Prioritätsregelverfahren im Vergleich zum GA [100%]

7 Fazit

Aufgrund der hier grob skizzierten Ergebnisse wird abschließend ein kurzes Fazit in Form einiger Thesen gezogen, die zur Diskussion gestellt werden sollen. Im dargestellten Anwendungskontext gilt für die vorgestellten Modelle zur Maschinenbelegungsplanung:

- Hopfield-Netzwerke sind zur Lösung komplexer Scheduling Probleme nicht geeignet.

- Ansätze auf Basis des Simulated Annealing arbeiten effizienter (bzgl. der Konvergenz*geschwindigkeit*) als genetische Algorithmen.

---

86) Zusätzlich wurden auch alle sinnvollen Kombinationen der angegebenen Regeln getestet, in denen die jeweils zweitgenannte Regel nur im Konfliktfall zur Anwendung kommt, d.h., falls eine eindeutige Auswahl eines Fertigungsarbeitsganges nach der ersten Regel nicht erfolgen kann. Für die dargestellten Daten ergeben sich allerdings (aufgrund fehlender Konflikte) kaum Änderungen, so daß hier auf die Darstellung der Ergebnisse für kombinierte Regeln verzichtet wird.

- Genetische Algorithmen erzielen bzgl. der Lösungsqualität bessere Ergebnisse als Ansätze auf Basis des Simulated Annealing.

- Hinsichtlich des Einsatzes in parallelen Umgebungen bieten genetische Algorithmen gegenüber dem Simulated Annealing höhere Effizienzsteigerungspotentiale, da über dem Konstrukt der Population viele Operationen unabhängig voneinander durchgeführt werden können (Evaluierung, tw. Rekombination) und somit eine disjunkte Problemzerlegung vorgenommen werden kann (vgl. hierzu auch Kap. 4.7).

Der Trial-and-Error-Prozeß, der mit der Implementierung der dargestellten Verfahren begonnen wurde, wird fortgesetzt. Insbesondere wird dabei untersucht, wie zur Lösung realistischer Problemstellungen die Modellierungsspielräume genutzt werden können, die aus einer möglichen konzeptionellen Trennung von Lösungs*verfahren* und Problem/-Lösungs*repräsentationen* bei der Anwendung spezifischer heuristischer Verfahren erwachsen.

Literaturverzeichnis

Aarts/Korst (1989):
Aarts, E., Korst, J.: Simulated Annealing and Boltzmann Machines, A Stochastic Approach to Combinatorial Optimization and Neural Computing, Chichester et al. 1989.

Bierwirth (1993):
Bierwirth, C.: Flowshop Scheduling mit parallelen Genetischen Algorithmen, Eine problemorientierte Analyse genetischer Suchstrategien, Wiesbaden 1993.

Carlier/Pinson (1989):
Carlier, J., Pinson, E.: An algorithm for solving the job-shop problem, Management Science 35 (1989) 2, pp. 164-176.

Domschke u.a. (1993):
Domschke, W., Scholl, A., Voß, S.: Produktionsplanung, Ablauforganisatorische Aspekte, Berlin u.a. 1993.

Domschke/Drexl (1991):
Domschke, W., Drexl, A.: Einführung in Operations Research, 2. Aufl., Berlin u.a. 1991.

Dueck (1993):
Dueck, G.: New Optimization Heuristics - The Great Deluge Algorithm and the Record-to-Record Travel, Journal of Computational Physics (1993) 104, pp. 86-92.

Fisher/Thompson (1963):
Fisher, H., Thompson, G.L.: Probalistic Learning Combinations of Local Job-Shop Scheduling Rules, in: Muth, J.F., Thompson, G.L. (Eds.): Industrial Scheduling, Englewood Cliffs 1963, pp. 225-251.

Giffler/Thompson (1960):
Giffler, B., Thompson, G.L.: Algorithms for Solving Production-Scheduling Poblems, Operations Research 8 (1960), pp. 487-503.

Glover/Greenberg (1989):
Glover, F., Greenberg, H.J.: New approaches for heuristic search: A bilateral linkage with artificial Intelligence, European Journal of Operational Research (1989) 39, pp. 119-130.

Goldberg (1989):
Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning, Reading, MA 1989.

Hajek, B. (1988):
Hajek, B.: Cooling schedules for optimal annealing, Mathematics of Operations Research 13 (1988), pp. 311-229.

Hemani/Postula (1990):
Hemani, A., Postula, A.: A neural net based Self Organizing Schedule Algorithm, in: EDAC´90, Proceedings of the European Design and Automation Conference, 1990, pp. 136-140.

Holland (1975):
Holland, J.H.: Adaptation in natural and artificial systems, Ann Arbor 1975.

Hopfield (1982):
Hopfield, J.J.: Neural Networks and Physical Systems with Emergent Collective Computational Abilities; Proceedings of the National Academy of Science 79 (1982), pp. 2254-2258.

Hopfield/Tank (1985):
Hopfield, J.J., Tank, D.W.: Neural Computation of Decisions in Optimization Problems, Biological Cybernetics 52 (1985), pp. 147-152.

Horster u.a. (1993):
Horster, B., Schneider, B., Siedentopf, J.: Kriterien zur Auswahl konnektionistischer Verfahren für betriebliche Probleme, Arbeitsbericht Nr. 15 des Instituts für Wirtschaftsinformatik, Münster 1993.

Kemke (1988):
Kemke, C.: Der neuere Konnektionismus, Ein Überblick; Informatik Spektrum 11 (1988) 3, S. 143-162.

Kurbel (1993):
Kurbel, K.: Produktionsplanung und -steuerung, Methodische Grundlagen von PPS-Systemen und Erweiterungen, München, Wien 1993.

Kurbel (1994):
Kurbel, K.: Maschinenbelegungsplanung auf Basis neuronaler Netze - ein Vergleich mit konventionellen Verfahren, in: Wagner, H. (Hrsg.): Betriebswirtschaftslehre und Unternehmensforschung: Aktuelle problemorientierte Konzepte, Wiesbaden 1994, S. 53-73.

Kurbel/Meynert (1988):
Kurbel, K., Meynert, J: Flexibilität und Planungsstrategien für interaktive PPS-Systeme, HMD - Handbuch der modernen Datenverarbeitung 25 (1988), S. 60-72.

Kurbel u.a. (1992):
Kurbel, K., Nietsch, M., Rautenstrauch, C.: Objektorientierter Leitstand - effizienter Ansatz zur betriebsspezifischen Anpassung der Fertigungssteuerung, Fortschrittliche Betriebsführung und Industrial Engineering 41 (1992) 7, S. 288-293.

McCulloch/Pitts (1943):
McCulloch, W.S., Pitts, W.: A Logical Calculus of the Ideas Immanent in Nervous Activity, Bulletin of Mathematical Biophysics 5 (1943), pp. 115-133.

Metropolis et al. (1953):
Metropolis, N., Rosenbluth, A., Rosenbluth, N., Teller, A., Teller, E.: Equation of state calculations by fast computing machines, Journal of Chemical Physics 21 (1953), pp. 1087-1092.

Michalewicz (1992):
Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, Berlin et al. 1992.

Nakano/Yamada (1991):
Nakano, R., Yamada, T.: Conventional Genetic Algorithm for Job Shop Problems, in: Belew, R., Booker, L. (Eds.): Proceedings of the Fourth International Conference on Genetic Algorithms, San Mateo 1991, pp. 474-479.

Nietsch u.a. (1991):
Nietsch, M., Nietsch, T., Rautenstrauch, C., Rinschede, M., Siedentopf, J.: Anforderungen an einen elektronischen Leitstand - Ergebnisse einer Untersuchung in mittelständischen Industriebetrieben, Fortschrittliche Betriebsführung und Industrial Engineering 40 (1991) 6, S. 266-272.

Nietsch u.a. (1992):
Nietsch, M., Rinschede, M., Rautenstrauch, C.: Konzeption und ENtwicklung eines Objektmodells für einen individualisierbaren Leitstand, in: Görke, W., Rininsland, H., Syrbe, M. (Hrsg.): Information als Produktionsfaktor, Berlin u.a. 1992, S. 597-606.

Rohmann (1993):
Rohmann, T.: Reihenfolgeplanung mit Genetischen Algorithmen am Beispiel der Maschinenbelegungsplanung, Diplomarbeit im Fach Wirtschaftsinformatik an der Westfälischen Wilhelms-Universität Münster, Münster 1993.

Rojas (1993):
Rojas, R.: Theorie der neuronalen Netze, Eine systematische Einführung, Berlin u.a. 1993.

Ruppel/Siedentopf (1992):
Ruppel, A., Siedentopf, J.: Konnektionistisch motivierte Reihenfolgeplanung in Fertigungsleitständen, in: Görke, W., Rininsland, H., Syrbe, M. (Hrsg.): Information als Produktionsfaktor, Berlin u.a. 1992, S. 554-563.

Storer et al. (1992):
Storer, R.H., Wu, D.S., Vaccari, R.: Local Search in Problem and Heuristic Space for Job Shop Scheduling Genetic Algorithms, in: Fandel, G., Gulledge, T., Jones, A. (Eds.): New Directions for Operations Research in Manufacturing, Berlin et al. 1992, pp. 149-160.

Sprecher (1994):
Sprecher, A.: Resource-Constrained Project Scheduling, Exact Methods for the Multi-Mode Case, Berlin et al. 1994.

Suhl (1994):
Suhl, U.H.: MOPS - Mathematical OPtimization System, European Journal of Operational Research (1994) 72, pp. 312-322.

Suhl/Szymanski (1994):
Suhl, U.H., Szymanski, R.: Supernode Processing of Mixed-Integer Models, Arbeitsbericht des Instituts für Wirtschaftsinformatik und Operations Research der Freien Universität Berlin, Berlin, März 1994.

Yamada/Nakano (1992):
Yamada, T., Nakano, R.: A Genetic Algorithm Applicable to Large-Scale Job-Shop Problems, in: Männer, B., Manderick, B. (Eds.): Parallel Problem Solving from Nature 2, North Holland (1992), pp. 281-290.

Institut für Produktionswirtschaft und Industrielle Informationswirtschaft der Universität Leipzig

Verzeichnis der Arbeitsberichte

Nr. 1: Zelewski, Stephan: Das Konzept technologischer Theorietransformationen - eine Analyse aus produktionswirtschaftlicher Perspektive, Leipzig 1994.

Nr. 2: Siedentopf, Jukka: Anwendung und Beurteilung heuristischer Verbesserungsverfahren für die Maschinenbelegungsplanung - Ein exemplarischer Vergleich zwischen Neuronalen Netzwerken, Simulated Annealing und genetischen Algorithmen, Leipzig 1994.

Nr. 3: Zelewski, Stephan: Unternehmenskrisen und Konzepte zu ihrer Bewältigung, Leipzig 1994.

Nr. 4: Siedentopf, Jukka: Ein effizienter Scheduling-Algorithmus auf Basis des Threshold Accepting, Leipzig 1995.

Nr. 5: Zelewski, Stephan: Petrinetzbasierte Modellierung komplexer Produktionssysteme (Projekt PEMOPS), Band 1: Exposition, Leipzig 1995.

Nr. 6: Zelewski, Stephan: Petrinetzbasierte Modellierung komplexer Produktionssysteme (Projekt PEMOPS), Band 2: Bezugsrahmen, Leipzig 1995.

Nr. 7: Zelewski, Stephan: Petrinetzbasierte Modellierung komplexer Produktionssysteme (Projekt PEMOPS), Band 3: Einführung in Stelle/Transition-Netze, Leipzig 1995.

Nr. 8: Zelewski, Stephan: Petrinetzbasierte Modellierung komplexer Produktionssysteme (Projekt PEMOPS), Band 4: Verfeinerungen von Stelle/Transition-Netzen, Leipzig 1995.

Nr. 9: Zelewski, Stephan: Petrinetzbasierte Modellierung komplexer Produktionssysteme (Projekt PEMOPS), Band 5: Einführung in Synthetische Netze, Teilband 5.1: Darstellung des Kernkonzepts, Leipzig 1995.

Nr. 10: Zelewski, Stephan: Petrinetzbasierte Modellierung komplexer Produktionssysteme (Projekt PEMOPS), Band 5: Einführung in Synthetische Netze, Teilband 5.2: Auswertungsmöglichkeiten, Leipzig 1995.

Nr. 11: Zelewski, Stephan: Petrinetzbasierte Modellierung komplexer Produktionssysteme (Projekt PEMOPS), Band 6: Erweiterungen von Synthetischen Netzen, Leipzig 1995.

Nr. 12: Zelewski, Stephan: Petrinetzbasierte Modellierung komplexer Produktionssysteme (Projekt PEMOPS), Band 7: Fallstudie, Leipzig 1995.

Nr. 13: Zelewski, Stephan: Petrinetzbasierte Modellierung komplexer Produktionssysteme (Projekt PEMOPS), Band 8: Charakterisierung des Petrinetz-Konzepts, Leipzig 1995.

Nr. 14: Zelewski, Stephan: Petrinetzbasierte Modellierung komplexer Produktionssysteme (Projekt PEMOPS), Band 9: Beurteilung des Petrinetz-Konzepts, Leipzig 1995.

Nr. 15: Zelewski, Stephan: Petrinetzbasierte Modellierung komplexer Produktionssysteme (Projekt PEMOPS), Band 10: Petrinetz-Literatur, Leipzig 1995.

Nr. 16: Siedentopf, Jukka: An Efficient Scheduling Algorithm Based upon Threshold Accepting, Leipzig 1995.

Nr. 17: Siedentopf, Jukka: The Threshold Waving Algorithm for Job Shop Scheduling, Leipzig 1995.