

Fachbericht 4

Ein Genetischer Algorithmus für das Job-Shop-Scheduling-Problem

von
Ivo Rixen
und
Herbert Kopfer

25. April 1994
Version 2

Zusammenfassung: Im vorliegenden Beitrag wird ein Genetischer Algorithmus für das Job-Shop-Scheduling-Problem mit einer neuen Repräsentation vorgestellt, die sich an den technologischen Vorschriften der Problemstellung orientiert und nur zulässige Lösungen kodiert. Für diese Repräsentation wird die Wirkung unterschiedlicher Crossover-Operatoren untersucht. Außerdem wird das Lösungsverhalten des Genetischen Algorithmus durch Einbeziehung eines lokalen Suchverfahrens verbessert. Die Testergebnisse zu bekannten Benchmarkproblemen zeigen, daß der implementierte Genetische Algorithmus im Vergleich zu anderen Verfahren sehr gute Lösungen liefert.

Stichworte: Job-Shop-Scheduling-Problem, Genetische Algorithmen, technologieorientierte Repräsentation, lokale Suchverfahren

Abstract: This paper introduces a genetic algorithm for job-shop scheduling problems. The algorithm uses a new processing order oriented coding which only represents feasible solutions. For this representation the effect of different crossover operators is investigated. To improve the solution quality, a local search algorithm is employed. Applied to various difficult benchmark problems the genetic algorithm generates high quality solutions.

Keywords: job-shop scheduling problem, genetic algorithm, processing order oriented representation, local search algorithm

1 Verfahren zur Maschinenbelegungsplanung

Das hier betrachtete Problem stammt aus dem Bereich der Maschinenbelegungsplanung und wird als Job-Shop-Scheduling-Problem bezeichnet [Fre82], [DSV93]. Grob skizziert geht es bei diesem Problem um die Einplanung von Aufträgen auf Maschinen unter Einhaltung von Restriktionen und Verfolgung eines oder mehrerer Ziele. Die Aufträge bestehen aus Arbeitsvorgängen und technologischen Vorschriften. Die technologischen Vorschriften geben an, in welcher Reihenfolge und auf welchen Maschinen die Arbeitsvorgänge des jeweiligen Auftrages einzuplanen sind. Die Maschinen sind verschiedenartig, so daß keine Maschine eine andere Maschine substituieren kann.

Das Job-Shop-Scheduling-Problem ist NP -schwer [GJS76] [LKB77] und gilt als eines der schwierigsten kombinatorischen Probleme. Seit den fünfziger Jahren werden aus der Literatur die unterschiedlichsten Verfahren für dieses Problem vorgeschlagen. Hervorzuheben sind die klassischen Graphen- bzw. Baumsuchverfahren, wie z.B. Branch-and-Bound-Verfahren [CP89] und einfache Prioritätsregelverfahren [Hau89]. Neben diesen allgemeinen Verfahren wurden für das Job-Shop-Scheduling-Problem spezielle Verfahren entwickelt. Eines der besten ist das Shifting-Bottleneck-Verfahren [ABZ88]. Dieses Verfahren löst iterativ Ein-Maschinen-Probleme. Die Reihenfolge, in der die Probleme gelöst werden, richtet sich dabei nach der kritischen Maschine, der sogenannten Engpaßmaschine. In jüngster Zeit werden erfolgreich lokale Suchverfahren, wie Simulated Annealing [LAL92] und Tabu Search [Tai92], [DT93] angewendet. Hierbei handelt es sich um Verbesserungsverfahren, die eine Lösung durch eine andere aus der Nachbarschaft solange ersetzen, bis ein vorgegebenes Kriterium erfüllt wird. Eine weitere Verfahrensklasse, welche ähnlich erfolgreich auf Job-Shop-Scheduling-Probleme angewendet wird, bilden die Genetischen Algorithmen [YN92], [Pes93]. Genetische Algorithmen verwenden im Unterschied zu den lokalen Suchverfahren gleichzeitig mehrere Lösungen. Iterativ werden bessere Lösungen bevorzugt ausgesucht, um durch Kombination ihrer speziellen Eigenschaften neue Lösungen zu erzeugen, die die bisherigen Lösungen verdrängen.

In diesem Artikel stellen wir einen Genetischen Algorithmus mit einer neuen Repräsentation für das Job-Shop-Scheduling-Problem vor. Bei dieser Repräsentation werden die Arbeitsvorgänge nach den technologischen Vorschriften der Aufträge geordnet. Dadurch ist sichergestellt, daß alle im Suchraum darstellbaren Lösungen die vorgeschriebene Reihenfolge für die Durchführung von Arbeitsvorgängen einhalten. Der Beitrag ist wie folgt gegliedert. Im nachfolgenden zweiten Kapitel wird das Job-Shop-Scheduling-Problem beschrieben. Im dritten Kapitel werden die Prinzipien und Komponenten Genetischer Algorithmen erläutert. Im vierten Kapitel wird ein neuer Genetischer Algorithmus zur Maschinenbelegungsplanung vorgestellt. Hier werden u.a. die Repräsentation, die genetischen Operatoren und eine Hybridisierung mit einem lokalen Suchverfahren präsentiert. Im fünften Kapitel wird das Lösungsverhalten des Genetischen Algorithmus untersucht.

2 Das Job–Shop–Scheduling–Problem

In einem statischen und deterministischen Modell wird das Problem wie folgt beschrieben:

J ist eine Menge von n Aufträgen. Jeder *Auftrag* J_j ($j = 1, 2, \dots, n$) muß vollständig bearbeitet werden. Der Bereitstellungsstermin r_j gibt den Zeitpunkt an, zu dem der Auftrag J_j in der Planungsperiode zur Bearbeitung bereitsteht. Der Liefertermin d_j gibt den Zeitpunkt an, zu dem die Bearbeitung des Auftrags J_j beendet sein sollte.

M ist eine Menge von m Maschinen. Jede *Maschine* M_k ($k = 1, \dots, m$) ist von einem anderen Maschinentyp. Alle Maschinen stehen in der Planungsperiode zeitlich unbegrenzt und ohne Störungen zur Verfügung und können jeweils höchstens einen Auftrag J_j gleichzeitig bearbeiten.

Jeder Auftrag J_j besteht aus n_j Arbeitsvorgängen, die auf den Maschinen aus M bearbeitet werden. In der Regel wird jeder Auftrag auf jeder Maschine genau einmal bearbeitet. Es kann jedoch vorkommen, daß ein Auftrag nicht auf allen Maschinen oder mehrfach auf der gleichen Maschine bearbeitet werden muß.

O_{ijk} ist der i -te *Arbeitsvorgang* ($i = 1, \dots, n_j$) von Auftrag J_j auf der Maschine M_k . Jeder Arbeitsvorgang wird genau einmal bearbeitet und kann während der Bearbeitung nicht unterbrochen werden. Die Bearbeitungsdauer p_{ijk} ist die Zeitdauer, die für den Arbeitsvorgang O_{ijk} benötigt wird.

V_j ist die *technologische Vorschrift* des Auftrags J_j . Sie legt fest, in welcher Reihenfolge und auf welchen Maschinen die Arbeitsvorgänge des Auftrags J_j bearbeitet werden müssen ($O_{1j\sigma_j(1)}, \dots, O_{n_jj\sigma_j(n_j)}$). Hierbei bestimmt $\sigma_j(i)$ die Maschine M_k , auf der der i -te Arbeitsvorgang des Auftrags J_j gefertigt wird. Ein Arbeitsvorgang kann erst bearbeitet werden, wenn sein unmittelbarer Vorgänger beendet worden ist. Das Job–Shop–Scheduling–Problem ist dadurch gekennzeichnet, daß jeder Auftrag J_j eine individuelle technologische Vorschrift haben kann. Spezielle Job–Shop–Scheduling–Probleme sind das Open–Shop–Scheduling–Problem, bei dem keine technologischen Vorschriften existieren, und das Flow–Shop–Scheduling–Problem, bei dem alle technologische Vorschriften gleich sind.

$u_{i'j'k}^{ij}$ bezeichnet die *reihenfolgeabhängige Rüstzeit* für die Umrüstung von Arbeitsvorgang $O_{i'j'k}$ ($i' = 1, \dots, n_{j'}, j' = 1, 2, \dots, n, k = 1, \dots, m$) auf Arbeitsvorgang O_{ijk} auf der Maschine M_k . Die Rüstzeit ist abhängig von dem unmittelbar vorher eingeplanten Arbeitsvorgang $O_{i'j'k}$ auf der Maschine M_k . Ist O_{ijk} der erste Arbeitsvorgang auf der Maschine M_k , so gibt die Rüstzeit u_{ijk} die Zeitdauer an, die benötigt wird, um die Maschine für diesen Arbeitsvorgang vorzubereiten.

Tabelle 1 zeigt ein Job–Shop–Scheduling–Problem von French [Fre82, S. 2] mit vier Aufträgen, die auf vier Maschinen zu bearbeiten sind. Der Bereitstellungsstermin und die Arbeitsvorgänge eines Auftrags werden zeilenweise aufgelistet. Die Bearbeitungszeiten der Arbeitsvorgänge werden zusätzlich in Klammern angegeben. Die technologische Vorschrift eines

Auftrags ergibt sich aus der Reihenfolge, in der die Arbeitsvorgänge des Auftrags in der jeweiligen Zeile auftreten.

Aufträge	Bereitstellungs- termin r_j	technologischen Vorschriften der Aufträge (Bearbeitungszeit)						
J_1	0	$O_{111}(60)$	\rightarrow	$O_{212}(30)$	\rightarrow	$O_{313}(2)$	\rightarrow	$O_{414}(5)$
J_2	15	$O_{122}(75)$	\rightarrow	$O_{223}(3)$	\rightarrow	$O_{321}(25)$	\rightarrow	$O_{424}(10)$
J_3	15	$O_{133}(5)$	\rightarrow	$O_{232}(15)$	\rightarrow	$O_{331}(10)$	\rightarrow	$O_{434}(30)$
J_4	60	$O_{144}(90)$	\rightarrow	$O_{241}(1)$	\rightarrow	$O_{342}(1)$	\rightarrow	$O_{443}(1)$

Tabelle 1: Das Beispielproblem von French

Die Menge S bezeichne alle Lösungen des Job-Shop-Scheduling-Problems. Eine Lösung $s \in S$ kann als vollständiger Plan oder als Sequenz-Plan dargestellt werden. Ein *vollständiger Plan* legt für alle Arbeitsvorgänge O_{ijk} den jeweiligen Bearbeitungsbeginn t_{ijk} fest. Die Wartezeit w_{ijk} des Arbeitsvorgangs O_{ijk} bezeichnet das Zeitintervall von der Fertigstellung des vorhergehenden Arbeitsvorgangs in der technologischen Vorschrift V_j bis zu dem Zeitpunkt t_{ijk} . Handelt es sich um den ersten Arbeitsvorgang in der technologischen Vorschrift eines Auftrags, so entspricht w_{ijk} dem Zeitintervall von dem Bereitstellungstermin r_j des Auftrags bis zu dem Bearbeitungsbeginn t_{ijk} . Ein *Sequenz-Plan* bestimmt für jede Maschine nur die organisatorische Reihenfolge der Arbeitsvorgänge. Ein Sequenz-Plan $q = \{q_1, \dots, q_m\}$ besteht also aus einer Menge von organisatorischen Reihenfolgen q_k , wobei q_k vorschreibt, in welcher Abfolge die Arbeitsvorgänge auf der Maschine M_k bearbeitet werden sollen.

Mit dem hier vorgestellten Genetischen Algorithmus können Job-Shop-Scheduling-Probleme mit durchlaufzeitbezogenen, terminbezogenen, kapazitätsbezogenen und rüstzeitbezogenen Zielen gelöst werden. Besonders häufig verwendete Ziele sind:

Minimierung der durchschnittlichen Wartezeit \bar{W} mit

$$\bar{W} = \frac{1}{n} \sum_{j=1}^n W_j \quad \text{und} \quad W_j = \sum_{i=1}^{n_j} w_{ij\sigma_j(i)},$$

Minimierung der Zykluszeit C_{max} mit

$$C_{max} = \max\{C_1, C_2, \dots, C_n\} \quad \text{und}$$

$$C_j = r_j + \sum_{i=1}^{n_j} (w_{ij\sigma_j(i)} + p_{ij\sigma_j(i)}) \quad , j = 1, 2, \dots, n,$$

Minimierung der maximalen Terminabweichung L_{max} mit

$$L_{max} = \max\{L_1, L_2, \dots, L_n\} \quad \text{und} \quad L_j = C_j - d_j \quad , j = 1, 2, \dots, n.$$

Die drei obigen Zielfunktionen sind regulär [Fre82, S. 14], d.h. der Zielfunktionswert nimmt nicht ab, wenn mindestens eine Fertigstellungszeit C_j zunimmt. Beziehungen zwischen den Zielen der Maschinenbelegungsplanung sind u.a. in French [Fre82, S. 28-31] zu finden.

Eine Klasse von Maschinenbelegungsplänen wird durch folgende Notation gekennzeichnet:

$$n/m/T/K.$$

Dabei gibt n die Anzahl der Aufträge und m die Anzahl der Maschinen an. T steht für den Problemtyp, wobei Job-Shop-Scheduling-Probleme mit G , Open-Shop-Scheduling-Probleme mit OS und Flow-Shop-Scheduling-Probleme mit FS bezeichnet werden. Durch die Variable K wird die Zielfunktion (z.B. \bar{W} , C_{max} , L_{max}) festgelegt. Bei dem in Tabelle 1 dargestellten Beispiel von French handelt es sich um ein $4/4/G/C_{max}$ Problem.

3 Genetische Algorithmen

Genetische Algorithmen [Hol75], [Gol89], [Mic92], [SHF94] sind naturadaptive Verfahren, bei denen die Lösungen einer Problemstellung durch Individuen einer Population repräsentiert werden. Das Prinzip Genetischer Algorithmen sieht vor, daß durch Selektion und Reproduktion existierender Lösungen neue Lösungen erzeugt werden. Aufgrund des *Selektionsprozesses* werden Individuen aus der Population ausgewählt und Paare gebildet. Für die ausgewählten Paare generiert der *Reproduktionsprozeß* Nachkommen, d.h. neue Lösungen, die eine Ähnlichkeit zu ihren Vorfahren aufweisen. Es ist nützlich, das Zusammenspiel zwischen Selektion und Reproduktion um einen Akzeptanzmechanismus zu erweitern. Der *Akzeptanzmechanismus* hat eine selektierende Wirkung bei der Aufnahme von neu erzeugten Lösungen in die Population. Insgesamt werden dann durch Selektion und Reproduktion neue Lösungen erzeugt, die nach erfolgter Akzeptanz in die Population aufgenommen werden.

Ein *Individuum* der Population $P = \{I_1, \dots, I_h\}$ besteht nicht nur aus der Lösung, dem sogenannten *Phänotyp*, sondern auch aus einem Genotyp. Der *Genotyp*, der durch genetische Operatoren verändert werden kann, ist eine genetische Repräsentation des Phänotyps. Ein Genotyp kann aus einem oder mehreren Chromosomen bestehen. Ein Chromosom ist eine Zeichenkette, mit der genetische Eigenschaften eines Individuums festgelegt werden. Ein einzelnes Zeichen an einer bestimmten Position eines Chromosoms wird als Gen bezeichnet. Jedem Individuum I_x ($x = 1, \dots, h$) wird zudem eine Fitneß fit_x zugeordnet. Die *Fitneß* fit_x ist nicht einfach nur der Zielfunktionswert f_x des Individuums I_x , sondern drückt das Verhältnis der Zielfunktionswerte der Individuen einer Population zueinander aus.

Bei der Selektion dient die Fitneß als Entscheidungsgrundlage für die Auswahl von Individuen. Durch einen zufallsgesteuerten Mechanismus werden Individuen mit besserer Fitneß bevorzugt ausgewählt. Ausgehend von zwei ausgewählten Individuen erzeugt die Reproduktion in der Regel zwei neue Individuen. Dabei fügen *komplexe genetische Operatoren* [Mic92] zufallsgesteuert Informationen aus zwei Genotypen zusammen. Bei Genotypen, die aus mehreren Chromosomen bestehen, vertauscht der Rekombinations-Operator zwei ganze Chromosomen, während Crossover-Operatoren Teilinformationen aus den Chromosomen zusammenfügen. *Einfache genetische Operatoren* verändern nur einen Genotyp, ebenfalls

zufallsabhängig. Zu diesen Operatoren gehört z.B. die Mutation, die eine möglichst kleine Veränderung eines Genotyps verursachen soll. Der Akzeptanzmechanismus entscheidet aufgrund der Fitneß über das Überleben der neuen und der alten Individuen. Der Einsatz genetischer Operatoren wird durch Wahrscheinlichkeiten gesteuert, die als externe Parameter vorzugeben sind.

Der Ablauf eines Genetischen Algorithmus wird in Abbildung 1 skizziert. Zu Beginn wird eine Population $P_{g=1}$ zufällig erzeugt. Nach der Evaluierung der Individuen dieser Population tritt der Genetische Algorithmus in die Hauptschleife ein. In dieser wird iterativ für jede Generation g die Fitneß der Individuen der Population P_g berechnet, und es werden Paare für die Reproduktion ausgewählt. Neue Individuen werden erzeugt, die nach der Evaluierung entweder akzeptiert oder abgelehnt werden. Beendet wird der Genetische Algorithmus bei Eintritt einer zuvor festgelegten Abbruchbedingung. Die Abbruchbedingung kann sich nach unterschiedlichen Kriterien richten, wie z.B. maximale Laufzeit, Anzahl der erzeugten Generationen oder Niveau eines Satisfizierungsziels [SH92]. Ein weiteres Abbruchkriterium sind identische Fitneßwerte der Individuen einer Generation, da hier die Individuen für die Selektion und den Akzeptanzmechanismus nicht mehr unterscheidbar sind. Als endgültige Lösung dient das beste aller Individuen, das der Genetische Algorithmus während seines Verfahrensablaufs generiert hat.

```

Beginn
   $g := 1$ 
  initialisiere  $P_g$ 
  evaluiere Individuen aus  $P_g$ 
  wiederhole
    berechne Fitneß der Individuen aus  $P_g$ 
    selektiere Individuen aus  $P_g$  und bilde Paare
    reproduziere neue Individuen
    evaluiere neue Individuen
    akzeptiere neue Individuen in  $P_g$ 
     $g := g + 1$ 
  bis (Abbruchkriterium)
Ende

```

Abbildung 1: Ablauf eines Genetischen Algorithmus

Bei dem Entwurf eines Genetischen Algorithmus für eine konkrete Anwendung müssen seine Komponenten spezifiziert werden. Zur besseren Übersicht über die Entwurfsentscheidungen werden die Komponenten in drei Gruppen unterteilt. Zur ersten Gruppe gehören die Komponenten, die zur Repräsentation benötigt werden. Sie müssen für die jeweilige Problemklasse speziell entworfen werden. Die zweite Gruppe besteht aus Komponenten, die die Evolution beeinflussen. Sie sind von der zu lösenden Problemklasse unabhängig. Die dritte Gruppe ist die der genetischen Operatoren. Ihre Gestaltung erfolgt in Abstimmung mit den Repräsentations-Komponenten. Für diese drei Gruppen müssen folgende Entscheidungen getroffen werden:

Repräsentations–Komponenten: Die Entscheidungen bezüglich dieser Komponenten betreffen im wesentlichen die Entwicklung einer geeigneten Modellformulierung. Hierzu gehören die Kodierung und der Entwurf einer Evaluierungsfunktion, die die Genotypen interpretiert und bewertet.

Evolutions–Komponenten: Die wichtigsten Entscheidungen in dieser Gruppe betreffen die Fitneßfunktion, die Selektionsstrategie und den Akzeptanzmechanismus. Desweiteren muß entschieden werden, ob die Populationsgröße statisch oder dynamisch ist, ob die Anfangspopulation rein zufällig oder nach einer konkreten Zufallsverteilung erzeugt wird und aufgrund welcher Kriterien der Genetische Algorithmus beendet werden soll.

Genetische Operatoren: Die Entscheidungen in dieser Gruppe beziehen sich auf die Auswahl und Gestaltung der komplexen und einfachen Operatoren für die Reproduktion.

Ein Genetischer Algorithmus kann mit anderen Lösungsverfahren zu einem Verfahrensverbund kombiniert werden, um die Lösungsqualität zu verbessern. Ein solcher Verfahrensverbund wird als *hybrider Genetischer Algorithmus* bezeichnet. Es werden zwei Hybridisierungsansätze unterschieden: Der erste Ansatz verwendet zusätzliche problemspezifische Verfahren, die eine Suche auf den Phänotypen durchführen und als vierte optionale Gruppe von Komponenten angesehen werden können. Die zweite Art erweitert die vorhandenen Komponenten, z.B. die genetischen Operatoren, um problemspezifische Verfahren und/oder Informationen.

4 Entwicklung eines Genetischen Algorithmus

Zunächst stellen wir die neue Problemrepräsentation vor. Dann werden die Entscheidungen bezüglich der Evolutions–Komponenten getroffen. Anschließend wird die Auswahl genetischer Operatoren begründet. Abschließend wird ein lokales Suchverfahren zur Hybridisierung des Genetischen Algorithmus präsentiert.

4.1 Repräsentations–Komponenten

Die Repräsentations–Komponenten sind von zentraler Bedeutung. Denn durch die Kodierung wird die Größe des Suchraums beeinflusst, und die Evaluierungsfunktion, die auf jedes neue Individuum einer Population einmal angewandt werden muß, stellt den rechenintensivsten Bestandteil eines Genetischen Algorithmus dar. Aus der Literatur sind mehrere unterschiedliche Repräsentationen für Maschinenbelegungsprobleme bekannt [Dav85], [NY91], [KS91], [YN92], [SH92], [SB92], [SWV92], [DP93], [Pes93], [Bru93], [FRC93], [BKMU93]. Es wird unterschieden zwischen Repräsentationen, die Pläne darstellen, und solchen, die dazu verwendet werden, andere problemspezifische Verfahren zu dirigieren (hybride Repräsentationen). Bezüglich der verwendeten Kodierungsart wird zwischen binären Kodierungen, binärähnlichen Kodierungen und permutationsbasierten Kodierungen unterschieden.

Eine gute Repräsentation soll einen möglichst kleinen Suchraum definieren, der alle Kandidaten für eine optimale Lösung enthält, aber unbrauchbare Lösungen so weit wie möglich ausschließt. Zur Definition eines geeigneten Suchraums bietet sich die Repräsentation von

Sequenz-Plänen an. Alle anderen Daten, wie z.B. Bearbeitungszeiten und reihenfolgeabhängige Rüstzeiten, werden erst zur Evaluierung hinzugezogen. Die Schwierigkeit, Sequenz-Pläne zu kodieren, besteht darin, unzulässige Lösungen zu vermeiden. Um nur Sequenz-Pläne zu repräsentieren, die nicht gegen die technologischen Vorschriften verstoßen, kodieren wir nicht die organisatorische Reihenfolgen q_k eines Sequenzplans, sondern Reihenfolgen zwischen solchen Arbeitsvorgängen, die an gleicher Position innerhalb der technologischen Vorschriften stehen.

Kodierung: Ein Genotyp zur Repräsentation einer Lösung besteht aus $n_{\max} + 1$ Chromosomen mit $n_{\max} = \max\{n_1, n_2, \dots, n_n\}$, wobei n_j die Anzahl der zum Auftrag J_j gehörenden Arbeitsvorgänge ist. Jedes Chromosom kodiert eine Reihenfolge in Form einer Permutation. Das Chromosom Ch_i ($i = 1, \dots, n_{\max}$) repräsentiert die Reihenfolge aller Arbeitsvorgänge, die an der i -ten Stelle innerhalb einer technologischen Vorschrift stehen. Das letzte Chromosom $Ch_{n_{\max}+1}$ legt eine Rangfolge für die Aufträge der Problemstellung fest. Die Chromosomen eines Genotyps schreiben gemeinsam eine eindeutige Reihenfolge vor, nach der bei der Konstruktion des zugehörigen Phänotyps die Einplanung von Arbeitsvorgängen stattfinden soll.

Die Kodierung wird im folgenden anhand des $4/4/G/C_{max}$ Problems aus Tabelle 1 erläutert. Abbildung 2 zeigt einen Genotyp zur Repräsentation einer Lösung des Problems. Auf der linken Seite sind die Chromosomen des Genotyps und auf der rechten Seite die durch sie definierten Reihenfolgen der Arbeitsvorgänge bzw. Aufträge dargestellt. Das Chromosom Ch_i beschreibt eine Permutation der Elemente der i -ten Spalte der Tabelle mit den technologischen Vorschriften und das letzte Chromosom eine Permutation der Aufträge, d.h. der Elemente der ersten Spalte in Tabelle 1.

$$\begin{array}{ll}
 Ch_1 = (3, 2, 1, 4) & (O_{133}, O_{122}, O_{111}, O_{144}) \\
 Ch_2 = (3, 4, 1, 2) & (O_{232}, O_{241}, O_{212}, O_{223}) \\
 Ch_3 = (3, 4, 2, 1) & \Rightarrow (O_{331}, O_{342}, O_{321}, O_{313}) \\
 Ch_4 = (3, 1, 2, 4) & (O_{434}, O_{414}, O_{424}, O_{443}) \\
 Ch_5 = (3, 4, 2, 1) & (J_3, J_4, J_2, J_1)
 \end{array}$$

Abbildung 2: Ein Genotyp zur Repräsentation des $4/4/G/C_{max}$ Problem von French

Die Größe des Suchraums ergibt sich aus der Anzahl der möglichen Permutationen je Chromosom zu $c_1! \cdot \dots \cdot c_{n_{\max}}! \cdot n!$, wobei c_i die Anzahl der Arbeitsvorgänge im Chromosom Ch_i ist, und zu $(n!)^{m+1}$, wenn jeder Auftrag genau einmal von jeder Maschine bearbeitet werden muß. Dieser Wert ist im Vergleich zu anderen Repräsentationen sehr klein. Bei einer binären Kodierung [NY91], [TN92] ergibt sich die Größe des Suchraums zu $2^{m \cdot n(n-1)/2}$ und bei einer permutationsbasierten Kodierung mit Wiederholungen [BKMU93; FRC93] [BKMU93], [FRC93] zu $(m \cdot n)! / m^n$.

Evaluierungsfunktion: Um die Fitneß eines Individuums zu ermitteln, wird der Genotyp des Individuums in einen zugehörigen Phänotyp dekodiert. Die Dekodierung erfolgt durch die hintereinandergeschaltete Anwendung einer Prioritäts-Funktion und einer Plan-Funktion.

Die *Prioritäts-Funktion* erstellt eine eindeutige Reihenfolge aller Arbeitsvorgänge der Problemstellung (Prioritätsliste). Aufgrund der Prioritätsliste konstruiert die *Plan-Funktion* einen vollständigen Plan.

Die *Prioritäts-Funktion* bewertet alle Arbeitsvorgänge und sortiert sie nach aufsteigenden Bewertungen (Priorität). Ein Arbeitsvorgang, der keinen Vorgänger in der technologischen Vorschrift besitzt, wird mit der Position bewertet, die er in dem ersten Chromosom einnimmt, d.h. der erste Arbeitsvorgang des ersten Chromosoms wird mit 1 bewertet, der zweite mit 2, usw. Für alle übrigen Arbeitsvorgänge ergibt sich die Priorität aus der Summe der Position des Arbeitsvorgangs in dem entsprechenden Chromosom und der Priorität seines Vorgängers in der technologischen Vorschrift. Das letzte Chromosom wird benutzt, um die Reihenfolge von Arbeitsvorgängen mit gleicher Bewertung festzulegen. Bei gleicher Priorität ergibt sich die Reihenfolge der Arbeitsvorgänge aus der Rangfolge, die das letzte Chromosom für die zugehörigen Aufträge definiert. Die Prioritätsliste ist konsistent mit der technologischen Vorschrift, d.h. die durch die Prioritätsliste definierte Reihenfolge zwischen den Arbeitsvorgängen eines Auftrags stimmt mit der Reihenfolge, die von der technologischen Vorschrift dieses Auftrags verlangt wird, überein. Die Vorgehensweise der *Prioritäts-Funktion* wird in Abbildung 3 anhand des obigen Genotyps für das $4/4/G/C_{max}$ Problem demonstriert.

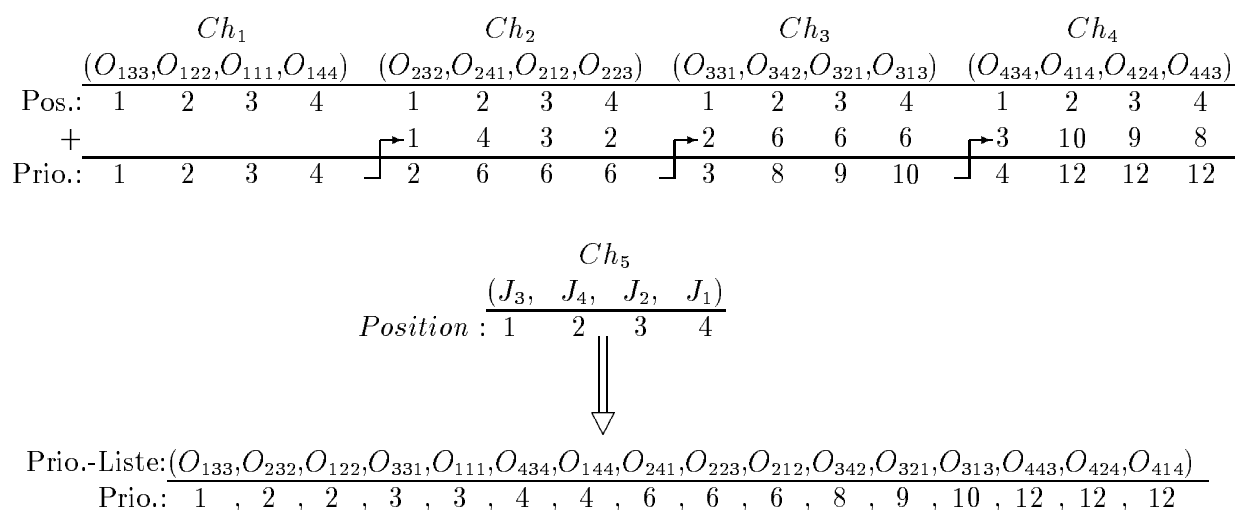


Abbildung 3: Vorgehensweise der *Prioritäts-Funktion*

Die *Plan-Funktion* erzeugt einen aktiven [DSV93, S. 369] vollständigen Plan, indem sie die Arbeitsvorgänge in der Reihenfolge der zuvor ermittelten Prioritätsliste nach den folgenden Regeln einplant:

- ein neu einzufügender Arbeitsvorgang wird so früh wie möglich eingeplant,
- ein bereits eingeplanter Arbeitsvorgang kann nicht mehr verschoben werden.

Dabei ist es möglich, daß ein neu einzufügender Arbeitsvorgang O_{ijk} auf der Maschine M_k früher terminiert als ein bereits eingeplanter Arbeitsvorgang auf der gleichen Maschine,

wenn nur ein genügend großer Zeitraum mit ungenutzter Maschinenzeit (Leerzeit) vor dem bereits eingeplanten Arbeitsvorgang existiert. Die Leerzeit ist groß genug, wenn die Summe aus Leerzeit und Rüstzeit für den bereits eingeplanten Arbeitsvorgang größer ist als die Bearbeitungsdauer p_{ijk} des einzufügenden Arbeitsvorgangs inklusive aller dann zu berücksichtigenden reihenfolgeabhängigen Rüstzeiten. Falls keine solche Leerzeit existiert, muß der neu einzufügende Arbeitsvorgang O_{ijk} an die bisher konstruierte organisatorische Reihenfolge q_k für die Maschine M_k hinten angefügt werden. Bei der Ermittlung des frühesten Zeitpunkts für die Einplanung des Arbeitsvorgangs O_{ijk} ist der Bereitstellungstermin r_j zu beachten. Ferner ist zu berücksichtigen, daß O_{ijk} frühestens zu einem Zeitpunkt starten kann, zu dem sein Vorgänger $O_{(i-1)j\sigma_j(i-1)}$ in der technologischen Vorschrift V_j beendet worden ist. Abbildung 4 zeigt das Gantt-Diagramm des vollständigen Plans, den die Planfunktion aufgrund der in Abbildung 3 dargestellten Prioritätsliste konstruiert.

Im Anschluß an die Dekodierung wird der konstruierte vollständige Plan mit der für die Problemstellung zugrundegelegten Zielfunktion bewertet.

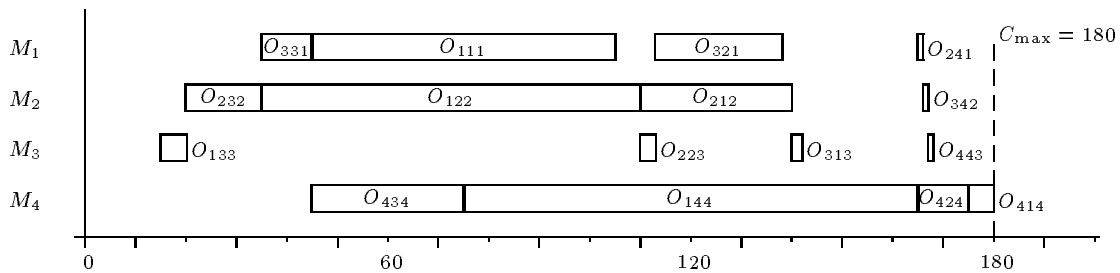


Abbildung 4: Minimaler Plan zum $4/4/G/C_{max}$ Problem von French

4.2 Evolutions-Komponenten

Die Größe der Population des Genetischen Algorithmus wird mit $h = const$ statisch festgelegt. Die Anfangspopulation wird zufällig erzeugt. Als Abbruchkriterium dient eine vorgegebene Anzahl von Generationen g .

Fitneßfunktion: Die verwendete Fitneßfunktion basiert auf dem Prinzip der proportionalen Fitneß, d.h. für Maximierungsprobleme gilt $fit_x = f_x / (\frac{1}{h} \sum_{i=1}^h f_i)$, wobei f_x den Zielfunktionswert des Individuums I_x bezeichnet. Für Minimierungsprobleme wird die Fitneß wie folgt berechnet:

$$fit_x = \frac{\sum_{i=1}^h f_i}{f_x} / \frac{\sum_{i=1}^h k_i}{h} \quad \text{mit} \quad k_i = \frac{\sum_{j=1}^h f_j}{f_i} \quad , x = 1, \dots, h.$$

Selektionsstrategie: Ein Individuum wird nach dem Prinzip stochastic sampling [Whi93] ausgewählt, d.h. die Wahrscheinlichkeit, mit der ein Individuum ausgewählt wird, ist $p_x = fit_x/h$ [Gol89].

Akzeptanzmechanismus: Die neu generierten Individuen werden nacheinander dem Akzeptanzmechanismus unterzogen. Für die Bildung einer neuen Generation wird der Fitneßwert des schlechtesten Individuums der Generation als Toleranzschwelle für den Vergleich mit den Fitneßwerten der neu generierten Individuen herangezogen. Die Population der neuen Generation wird schrittweise durch Austausch von Individuen aufgebaut. Ein neues Individuum wird in die Population aufgenommen, falls seine Fitneß besser ist als die Toleranzschwelle. In diesem Fall verdrängt es das schlechteste Individuum der aktuellen Population.

4.3 Genetische Operatoren

Bei der Bildung neuer Individuen kommen die folgenden genetischen Operatoren zum Einsatz.

Rekombinations-Operator: Dieser Operator bildet aus den Genotypen zweier Maschinenbelegungspläne zwei neue Genotypen, indem er Chromosomen, die an gleicher Position in den ursprünglichen Genotypen stehen, miteinander vertauscht. Für jede Position wird ein Tausch der zugehörigen Chromosomen in Betracht gezogen. Die Vertauschung zweier Chromosomen findet mit einer bestimmten Wahrscheinlichkeit (Rekombinationsrate) statt, die als externer Parameter vorgegeben wird.

Crossover-Operatoren: Crossover-Operatoren vertauschen Teilinformationen zwischen solchen Chromosomen, die an der gleichen Position zweier unterschiedlicher Genotypen stehen. Für jede Position der Genotypen zweier Individuen, die miteinander gepaart werden sollen, wird ein Crossover-Operator mit einer bestimmten extern vorgegebenen Wahrscheinlichkeit (Crossoverrate) ausgeführt. Crossover-Operatoren werden unmittelbar im Anschluß an die Rekombination und nur auf solche Chromosomen angewendet, die nicht miteinander vertauscht worden sind. In der Literatur werden mehrere permutationsbewahrende Operatoren für Genetische Algorithmen vorgeschlagen. Diese Operatoren [OSH87], [SMMW91], [SWMM92], [WSF89], [Sys91] unterscheiden sich hinsichtlich ihrer Fähigkeit, die in einem Chromosom kodierten unterschiedlichen Informationen zu bewahren, d.h. auf nachfolgende Individuen weitervererben zu können. Die in einem Chromosom in Form einer Permutation kodierten Informationen sind im wesentlichen die Umgebung, die Reihenfolge und die Position der Elemente der Permutation. Je nach Problemart (z.B. Tourenplanungsproblem oder Job-Shop-Scheduling-Problem) haben solche Informationen unterschiedlich starke Bedeutung. Aufgrund der Problemstruktur des Job-Shop-Scheduling-Problems haben wir uns entschieden, die Anwendung des 'position based Crossover' [Sys91] und des 'order Crossover' [OSH87] alternativ zu erproben. Der 'position based Crossover' zielt auf die Bewahrung von Positionsinformationen ab, während der 'order Crossover' Reihenfolgeinformationen bewahren will [OSH87].

Mutation: Die Mutation bewirkt kleine Veränderungen in einem Chromosom eines Genotyps und wird mit einer vorgegeben Wahrscheinlichkeit (Mutationsrate) für jedes Gen angewendet. Für die Mutation von permutationsbasierten Chromosomen bieten sich u.a. die beiden Operatoren 'order based Mutation' und 'position based Mutation' an [Sys91]. Bei dreißig Testläufen mit beiden Operatoren haben wir für fünf verschiedene Probleme festge-

stellt, daß bei Anwendung des Operators 'order based Mutation' bessere Ergebnisse erzielt werden konnten als bei Anwendung des 'position based Mutation'. Dies ist vermutlich darauf zurückzuführen, daß 'order based Mutation' die Positionsinformation stärker bewahrt als 'position based Mutation'. Im folgenden beschränken wir uns daher auf die Verwendung des erfolgversprechenderen Operators 'order based Mutation'.

4.4 Hybridisierung

Die Hybridisierung des hier vorgestellten Genetischen Algorithmus erfolgt durch die Kombination mit einem lokalen Suchverfahren. Das lokale Suchverfahren führt eine Suche auf den Phänotypen durch und dient zur Verbesserung der mit dem Genetischen Algorithmus erzeugten Individuen. Hierzu wird auf jedes neue Individuum in der Population ein heuristisches Verbesserungsverfahren der besten Verbesserung angewendet. Das Verbesserungsverfahren versucht mit Hilfe einer Nachbarschaftsstruktur, die jeder Lösung $s \in S$ eine Menge von Nachbarn $\mathcal{N}_s \subset S$ zuordnet, eine vorgegebene Bezugslösung iterativ zu verbessern. Das Verfahren beginnt mit der durch I_x vorgegebenen neuen Lösung s als Bezugslösung und sucht in der Nachbarschaft \mathcal{N}_s nach einer besseren Lösung. Gesucht wird die beste in der Nachbarschaft \mathcal{N}_s enthaltene Lösung t . Wenn diese Lösung t einen besseren Zielfunktionswert besitzt als s , dann wird t zur neuen Bezugslösung, und es wird eine erneute Suche in der Nachbarschaft von t gestartet. Das lokale Suchverfahren zur Verbesserung eines Individuums endet, wenn in der zu suchenden Nachbarschaft keine bessere Lösung als die aktuelle Bezugslösung gefunden werden kann.

Die Nachbarschaft \mathcal{N}_s einer Bezugslösung s besteht aus allen Lösungen, die durch die folgende Transformation von s erzeugt werden können [Laa88, S.69–75]. Die Transformation besteht aus einer Tauschoperation, bei der zwei Arbeitsvorgänge O_{ijk} und $O_{i'j'k}$ in der von s beschriebenen Lösung unter Berücksichtigung der folgenden Bedingungen miteinander vertauscht werden:

1. beide Tauschpartner gehören zu verschiedenen Aufträgen,
2. die zum Tausch ausgewählten Arbeitsvorgänge O_{ijk} und $O_{i'j'k}$ werden auf der gleichen Maschine M_k unmittelbar hintereinander bearbeitet,
3. beide Tauschpartner liegen auf dem kritischen Pfad, d.h. sie verhindern, daß der Arbeitsvorgang, der von allen Arbeitsvorgängen der Problemstellung als letzter beendet wird, in der aktuellen Lösung s zu einem früheren Zeitpunkt eingeplant werden kann.

Die Hybridisierung des Genetischen Algorithmus hat zur Folge, daß der Phänotyp eines jeden Individuums als ein lokales Minimum des Verbesserungsverfahrens interpretiert wird. Die Effizienzsteigerung, die durch das lokale Suchverfahren erreicht werden kann, ist am größten, wenn die Zielsetzung der Problemstellung in der Minimierung der Zykluszeit C_{max} besteht. Dies liegt an der Wahl der Nachbarschaftsstruktur des Verbesserungsverfahrens, da Tauschoperationen nur auf dem kritischen Pfad vorgenommen werden.

5 Testergebnisse

Den Genetischen Algorithmus haben wir objektorientiert in der Sprache C++ unter dem UNIX-Betriebssystem Solaris 2.3 auf einer Sun Sparc-Station 10/40 implementiert. Das Lösungsverhalten des Algorithmus haben wir an den Problemen $6/6/G/C_{max}$, $10/10/G/C_{max}$, $20/5/G/C_{max}$ Fisher und Thompson [FT63] und an dem Problem $15/10/G/C_{max}$ von Lawrence (vgl. Tabelle 7, erstes Problem in [Law84]) getestet. Diese Testprobleme bezeichnen wir im folgenden mit ft1, ft2, ft3 bzw. la21.

Es wurden drei unterschiedliche Konfigurationen erstellt, die sich bezüglich des verwendeten Crossover-Operators und hinsichtlich ihrer Hybridisierung unterscheiden. Die erste Konfiguration (OX) verwendet den 'order Crossover' und ist nicht hybridisiert; die zweite Konfiguration (PBX) verwendet den 'position based Crossover' und ist ebenfalls nicht hybridisiert; die dritte Konfiguration (PBX/H) verwendet den 'position based Crossover' und ist durch eine Kombination mit dem oben beschriebenen Verbesserungsverfahren hybridisiert worden.

Problem	Konfiguration	Parameter		t_{\emptyset}	Resultate			Opt C_{max}
		h	g		C_{max}^{\emptyset}	C_{max}^{SD}	C_{max}^{best}	
ft1	OX	32	30	2 sec.	55,26	0,68	55	55
	PBX	32	30	2 sec.	55,33	0,87	55	
	PBX/H	12	30	1 sec.	55	0	55	
ft2	OX	256	250	5,9 min.	992,6	14,2	964	930
	PBX	256	250	5,1 min.	987	15,7	948	
	PBX/H	96	250	12,3 min.	951,4	9,9	930	
ft3	OX	256	250	7,9 min.	1269,7	20	1223	1165
	PBX	256	250	6,9 min.	1238,6	14,5	1203	
	PBX/H	96	250	18,5 min.	1203,3	12,6	1185	
la21	OX	256	250	17 min.	1106,1	17,9	1079	1047
	PBX	256	250	15,8 min.	1098,4	12,4	1068	
	PBX/H	96	250	27,8 min.	1070,4	9,1	1055	

Tabelle 2: Ergebnisse für die Benchmark-Probleme

Die Populationsgröße h und die Anzahl der Generationen g beeinflussen das Lösungs- und Rechenzeitverhalten in besonderem Maße. Bei kleinen Problemen, wie z.B. dem ft1 Problem, sind für die Populationsgröße h und die Anzahl der Generationen g Werte ausreichend, die ungefähr dem Produkt $n * m$ entsprechen. Für alle anderen Probleme beschränken wir den Parameter h auf einen maximalen Wert von 256 und den Parameter g auf einen Wert von 250. Da die lokale Nachoptimierung der neuen Individuen in der Population einen großen zusätzlichen Rechenaufwand verursacht, ist das Verfahren PBX/H so parametrisiert worden, daß die Populationsgröße h möglichst klein ist, aber dennoch genügend Individuen für die Variabilität zur Verfügung stehen. Um die Population für PBX/H mit zusätzlichem genetischen Material anzureichern, werden die Individuen einer Generation mutiert, sobald die Standardabweichung der Funktionswerte der Individuen der Generation gleich Null ist. Die

Werte der übrigen Parameter sind unabhängig von der Problemgröße für alle Testprobleme, für alle Konfigurationen und alle Testläufe gleich gewählt worden. Die Rekombinations- und Crossoverrate erhalten den Wert 0,5 und die Mutationsrate 0,001.

Tabelle 2 faßt die Ergebnisse der Tests der drei Konfigurationen zusammen. Jede Konfiguration wurde für jedes Testproblem mit den angegebenen Parameterwerten für h und g dreißigmal ausgeführt. Die Spalte C_{\max}^{\emptyset} zeigt die arithmetischen Mittel der Zielfunktionswerte der dreißig Lösungen, die die jeweilige Konfiguration für das jeweilige Testproblem gefunden hat. Die Spalte C_{\max}^{SD} zeigt die Standardabweichung dieser dreißig Zielfunktionswerte. C_{\max}^{best} zeigt die Zielfunktionswerte der jeweils besten Lösung des Genetischen Algorithmus aus den dreißig durchgeführten Testläufen an. In der Spalte t_{\emptyset} wird der Rechenzeitbedarf angegeben, den eine Konfiguration bis zum erstmaligen Auffinden der endgültigen Lösung, d.h. der besten aller nach g Generationen gefundenen Lösungen, auf einer Sun Sparc-Station 10/40 durchschnittlich benötigt. In der Spalte Opt werden die besten in der Literatur bekannten Lösungen für die vier Testprobleme angegeben. Die ersten drei Lösungen der Spalte OPT sind minimal. Für das letzte Problem la21 konnte bisher nicht nachgewiesen werden, daß die beste bekannte Lösung optimal ist. Sie wurde in [Tai92] angegeben.

Tabelle 2 kann man entnehmen, daß die Konfiguration PBX bessere Testergebnisse für C_{\max}^{\emptyset} und C_{\max}^{best} generiert und ein robusteres Lösungsverhalten mit einer geringeren Standardabweichung C_{\max}^{SD} aufweist als die Konfiguration OX. Auch der Rechenzeitbedarf t_{\emptyset} ist für die Testläufe der Konfiguration PBX niedriger als für die Testläufe der Konfiguration OX. Die Lösungsqualität der Konfiguration PBX kann durch Hybridisierung des Verfahrens nochmals verbessert werden. Das hybridisierte Verfahren PBX/H benötigt allerdings pro Individuum mehr Rechenzeit, was durch die Reduzierung der Populationsgröße h nur zum Teil kompensiert werden kann.

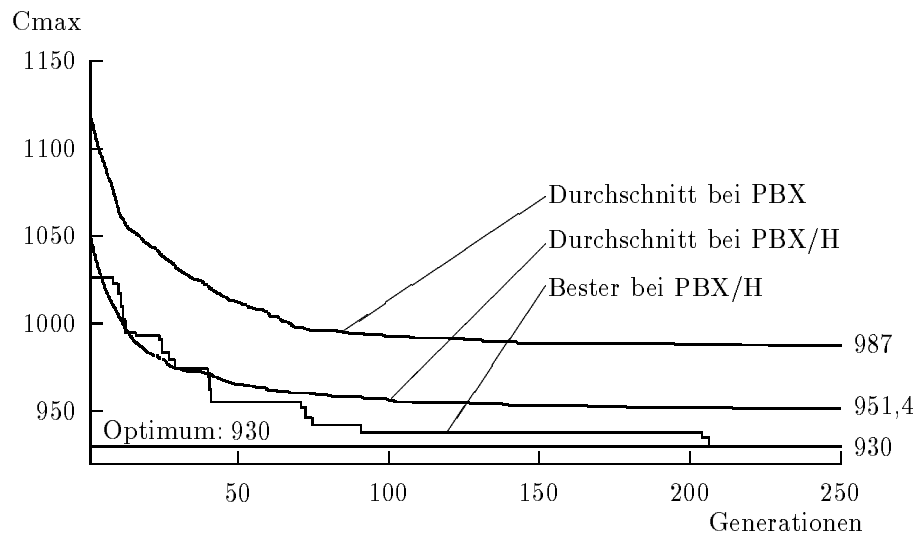


Abbildung 5: Adaptionskurven der zwei Konfigurationen für das ft2 Problem

Am Beispiel des ft2 Problems vergleicht Abbildung 5 das Lösungsverhalten der Konfigurationen PBX und PBX/H in Form von Adaptionenkurven. Eine *Adaptionenkurve* gibt an, wie sich die Qualität der vom Genetischen Algorithmus gefundenen Lösung im Verlauf der Suche, d.h. mit zunehmender Anzahl der Generationen, entwickelt. Unter der Lösung des Genetischen Algorithmus ist dabei das beste aller bis zu dieser Generation erzeugte Individuen zu verstehen. Für die Konfiguration PBX wird eine Adaptionenkurve dargestellt. Die Adaptionenkurve der Konfiguration PBX zeigt für jeden Zeitpunkt der Suche (Generation) den Durchschnittswert der Lösungen, die die Konfiguration in dreißig Testläufen ermittelt hat. Für die Konfiguration PBX/H werden zwei Adaptionenkurven dargestellt. Die erste Adaptionenkurve zeigt den Durchschnittswert der Lösungen der dreißig Testläufe, und die zweite Adaptionenkurve zeigt die beste der Lösungen, die die Konfiguration PBX/H in dreißig Testläufen ermittelt hat.

Jahr	Autoren	Verfahrensbasis (Name)	ft2	ft3	la21	Literatur
1991	Nakano, Yamada	GA	965	1215	-	[NY91]
1993	Bierwirth, Kopfer, Mattfeld, Utecht	GA (PARNET 1)	957	1191	1089	[BKMU93]
1993	Bierwirth	GA (PARNET 2)	936	1181	1076	[Bie94]
1993	Fang, Ross, Corne	GA without GVOT	960	1213	-	[FRC93]
1993	Fang, Ross, Corne	GA with GVOT	949	1189	-	[FRC93]
1992	Yamada, Nakano	GA (GA/GT)	930	1184	-	[YN92]
1993	Dorndorf, Pesch	GA	930	1165	-	[DP93]
	Rixen, Kopfer	GA (PBX/H)	930	1185	1055	
1992	Storer, Wu, Park	GA (P-GA)	954	1180	-	[SWP93]
1992	Dorndorf, Pesch	GA (Rule-GA)	960	1249	1139	[DP92]
1992	Dorndorf, Pesch	GA (SB-GA)	938	1178	1074	[DP92]
1993	Pesch	GA (1MCP-GA)	930	1165	1070	[Pes93]
1993	Pesch	GA (2JP-GA)	937	1193	1102	[Pes93]
1993	Pesch	GA (2JCP-GA)	937	1175	1074	[Pes93]
1988	Adams, Balas, Zawack	Reduktion (SB I)	1015	1290	1172	[ABZ88]
1992	Van Laarhoven, Aarts, Lenstra	SA	930	1165	1063	[LAL92]
1993	Dell'Amico, Trubian	TS	930	1165	1048	[DT93]

Tabelle 3: Lösungsverhalten verschiedener Verfahren

In der Literatur werden mehrere alternative Genetische Algorithmen vorgeschlagen, die in der Lage sind, schwierige Problemstellungen des hier betrachteten Job-Shop-Scheduling-Problems zu lösen. Für viele dieser Algorithmen sind die Lösungen, die sie für die hier betrachteten Testprobleme ermitteln, bekannt. Tabelle 3 vergleicht die Qualität dieser Lösungen mit den Lösungen, die von der Konfiguration PBX/H ermittelt wurde. Dabei sind die aufgelisteten Genetischen Algorithmen durch einen Trennstrich in zwei Gruppen gegliedert. Die obere Gruppe besteht aus Verfahren, bei denen Maschinenbelegungspläne durch Genotypen dargestellt werden; die untere Gruppe besteht aus Verfahren, deren Individuen dazu

dienen, andere problemspezifische Verfahren zu dirigieren. Um die Qualität der Lösungen der Genetischen Algorithmen besser einordnen zu können, erhält Tabelle 3 außerdem die Lösungen, die mit dem Shifting-Bottleneck-Verfahren (SB I), mit Tabu Search (TS) und Simulated Annealing (SA) erzielt werden. Sie sind durch einen doppelten Strich von den Genetischen Algorithmen getrennt.

Die zukünftige Forschungsarbeit wird sich auf die weitergehende Hybridisierung mit einem lokalen Suchverfahren und auf die Parallelisierung des Genetischen Algorithmus konzentrieren. Durch andere Nachbarschaftsstrukturen soll das Lösungsverhalten des hybriden Genetischen Algorithmus verbessert werden. Mit der Parallelisierung des Genetischen Algorithmus kann ebenfalls das Rechenzeitverhalten und die Qualität der ermittelten Lösungen positiv beeinflusst werden. Zwei vielversprechende Ansätze der Parallelisierung und eine Kombination dieser Ansätze sollen erprobt werden. Der erste Ansatz führt zu einem neuen Populations-Modell, dem Diffusions-Modell, in dem die Individuen selbst die Kontrolle über Partnerwahl und Akzeptanz übernehmen. Der zweite Ansatz teilt die Population in einzelne unabhängige Subpopulationen auf, wobei jede Subpopulation für sich einen Evolutionsprozeß simuliert. In gewissen Abständen migrieren einzelne Individuen zwischen den Subpopulationen, um neues Gen-Material bereitzustellen. Bei dem kombinierten Ansatz simuliert jede Subpopulation auf der Basis des Diffusions-Modells einen Evolutionsprozeß.

Literatur

- [ABZ88] J. Adams, E. Balas, and D. Zawack. The shifting bottleneck procedure for job shop scheduling. *Management Science*, 34:391–401, 1988.
- [Bie94] C. Bierwirth. A generalized permutation approach to jobshop scheduling with genetic algorithms. In Voss, editor, *??*, pages ??–?? Springer-Verlag, Heidelberg, 1994.
- [BKMU93] C. Bierwirth, H. Kopfer, D. Mattfeld, and T. Utecht. Genetische Algorithmen und das Problem der Maschinenbelegung. Working paper, Lehrstuhl für Logistik, Bremen, 1993.
- [Bru93] R. Bruns. Direct chromosome representation and advanced genetic operators for production scheduling. In Stephanie Forrest, editor, *Proceedings of the fifth international Conference on Genetic Algorithms*, pages 352–359, San Mateo, California, 1993. Morgan Kaufmann Publisher.
- [CP89] J. Carlier and E. Pinson. An algorithm for solving the job-shop problem. *Management Science*, 35:164–176, 1989.
- [Dav85] L. Davis. Job shop scheduling with genetic algorithms. In J.J. Grefenstette, editor, *Proceedings of an international conference on Genetic Algorithms and their Applications*, pages 136–140. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1985.

- [DP92] U. Dorndorf and E. Pesch. Evolution based learning in a job shop scheduling environment. Research memorandum 92-019, University of Limburg, Netherlands, 1992.
- [DP93] U. Dorndorf and E. Pesch. Combining genetic and local search for solving the job shop scheduling problem. In I. Maros, editor, *Symposium on Applied Mathematical Programming and Modeling – APMOD93*, pages 142–149. Budapest, 1993.
- [DSV93] W. Domschke, A. Scholl, and S. Voß. *Produktionsplanung*. Springer Verlag, Berlin Heidelberg, 1993.
- [DT93] M. Dell’Amico and Marco Trubian. Applying tabu search to the job–shop scheduling problem. *Annals of Operations Research*, 41:231–252, 1993.
- [FRC93] H.-L. Fang, P. Ross, and D. Corne. A promising genetic algorithm approach to job-shop scheduling, rescheduling and open-shop scheduling problems. In Stephanie Forrest, editor, *Proceedings of the fifth international Conference on Genetic Algorithms*, pages 375–382, San Mateo, California, 1993. Morgan Kaufmann Publisher.
- [Fre82] S. French. *Sequencing and Scheduling - An Introduction to the Mathematics of the Job-Shop*. Ellis Horwood Series. John Wiley & Sons, West-Sussex, 1982.
- [FT63] H. Fisher and G. L. Thompson. Probabilistic learning combinations of local job –shop scheduling rules. In G. L. Thompson J. F. Muth, editor, *Industrial Scheduling*, pages 225–251. Prentice – Hall, Englewood Cliffs, New Jersey, 1963.
- [GJS76] M.R. Garey, D.S. Johnson, and R.R. Sethi. The complexity of flow–shop and job–shop scheduling. *Mathematics of Operations Research*, 1:117–129, 1976.
- [Gol89] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, Reading, Mass, 1989.
- [Hau89] R. Haupt. A survey of priority rule based scheduling. *OR–Spectrum*, 11:3–16, 1989.
- [Hol75] J.H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.
- [KS91] J.J. Kanet and V. Sridharan. PROGENITOR: A genetic algorithm for production scheduling. *Wirtschaftsinformatik*, 33:332–336, 1991.
- [Laa88] P. J. M. Van Laarhoven. *Theoretical and computational aspects of simulated Annealing*. Centre for Mathematics and Computer Science, Amsterdam, 1988.
- [LAL92] P. J. M. Van Laarhoven, E. H. L. Aarts, and J. K. Lenstra. Job shop scheduling by simulated annealing. *Operations Research*, 40:113–125, 1992.

- [Law84] S. Lawrence. *Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques*. Graduate School of Industrial Administration, University Pittsburg, Carnegie Mellon, Pittsburg, 1984.
- [LKB77] J. K. Lenstra, A.H.G. Rinnooy Kan, and P. Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:343–362, 1977.
- [Mic92] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer–Verlag, Berlin u.a., 1992.
- [NY91] R. Nakano and T. Yamada. Conventional genetic algorithm for job shop scheduling. In R.K. Belew and L.B. Booker, editors, *Proceedings of the fourth international Conference on Genetic Algorithms*, pages 474–479, San Mateo, California, 1991. Morgan Kaufmann Publisher.
- [OSH87] I. Oliver, D. Smith, and J. Holland. A study of permutation crossover operators on the traveling salesman problem. In J.J. Grefenstette, editor, *Genetic Algorithms and their Applications: Proceedings of the second International Conference on Genetic Algorithms*, pages 224–230. L. Erlbaum Associates, Hillsdale, New Jersey, 1987.
- [Pes93] E. Pesch. Machine learning by schedule decomposition. Working paper, Faculty of Economics and Business Administration, University of Limburg, Netherlands, 1993.
- [SB92] S. Stöppler and C. Bierwirth. The application of a parallel genetic algorithm to the $n/m/p/c_{max}$ flowshop problem. In G. Fandel, Th. Gullledge, and A. Jones, editors, *New Directions for Operations Research in Manufacturing*, pages 161–178. Springer–Verlag, Heidelberg, 1992.
- [SH92] E. Schöneburg and F. Heinzmann. PERPLEX: Produktionsplanung nach dem Vorbild der Evolution. *Wirtschaftsinformatik*, 34:224–232, 1992.
- [SHF94] E. Schöneburg, F. Heinzmann, and S. Feddersen. *Genetische Algorithmen und Evolutionsstrategie*. Addison–Wesley, Bonn Paris Reading Mass., 1994.
- [SMMW91] T. Starkweather, S. McDaniel, K. Mathias, and D. Whitley. A comparison of genetic sequencing operators. In R.K. Belew and L.B. Booker, editors, *Proceedings of the fourth international Conference on Genetic Algorithms*, pages 69–76, San Mateo, California, 1991. Morgan Kaufmann Publisher.
- [SWMM92] T. Starweather, D. Whitley, K. Mathias, and S. McDaniel. Sequence scheduling with genetic algorithms. In G. Fandel, Th. Gullledge, and A. Jones, editors, *New Directions for Operations Research in Manufacturing*, pages 129–148. Springer–Verlag, Heidelberg, 1992.

- [SWP93] R. H. Storer, S. D. Wu, and I. Park. Genetic algorithms in problem space for sequencing problems. In G. Fandel, Th. Gullledge, and A. Jones, editors, *Operations Research in Production Planning and Control*, pages 584–597. Springer-Verlag, Heidelberg, 1993.
- [SWV92] R. H. Storer, S. D. Wu, and R. Vaccari. Local search in problem and heuristic space for job-shop scheduling genetic algorithm. In G. Fandel, Th. Gullledge, and A. Jones, editors, *New Directions for Operations Research in Manufacturing*, pages 149–160. Springer-Verlag, Heidelberg, 1992.
- [Sys91] G. Syswerda. Scheduling optimization using genetic algorithms. In L. Davis, editor, *Handbook of Genetic Algorithms*, pages 332–349. Van Nostrand Reinhold, New York, 1991.
- [Tai92] E. Taillard. Parallel taboo search technique for the jobshop scheduling problem. Internal report orwp 89/11, Département de Mathématiques, Ecole Polytechnique Fédérale de Lausanne, Lausanne, 1992.
- [TN92] H. Tamaki and Y. Nishikawa. A paralleled genetic algorithm based on a neighborhood model and its application to the jobshop scheduling. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, pages 573–582. Elsevier Science Publishers B. V., North-Holland, 1992.
- [Whi93] D. Whitley. A genetic algorithm tutorial. Technical report cs-93-103, Department of Computer Science, Colorado State University, USA, 1993.
- [WSF89] D. Whitley, T. Starkweather, and D. Fuquay. Scheduling problems and traveling salesmen : The genetic edge recombination operator. In D. Schaffer, editor, *Proceedings of the 3rd International Conference on Genetic Algorithms and their Applications*, pages 133–140. Morgan Kaufmann Publisher, San Mateo, 1989.
- [YN92] T. Yamada and R. Nakano. A genetic algorithm applicable to large-scale job-shop problems. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, pages 281–290. Elsevier Science Publishers B. V., North-Holland, 1992.