Ausarbeitung zum Vortrag

Direct Chromosome Representation and Advanced Genetic Operators for Production Scheduling

Autor Alexander Dragosch

1 Einleitung 3

2 Problembeschreibung 3
2.1 Allgemeines 3
2.2 Formelle Beschreibung 3

3 GA-Basierte Verfahren 6
3.1 Überblick 6
3.2 Ergebnis 7

4 Wissensbasierter GA 7
4.1 Wissensbasierte Initialisierung: 9
4.2 Wissensbasierter Crossover: 9
4.3 Wissensbasierter Mutationsoperator: 10
4.4 Zusammenfassung 10
4.5 Versuche / empirische Resultate 11

5 Zusammenfassung 14

6 Diskussion: Ist dieser Algorithmus für das Problem der Fahrplanoptimierung geeignet ? 14

7 Literatur 16

1 Einleitung

Bekannte Standardprobleme der Klasse der Ablaufplanungsprobleme (scheduling problems) sind das JSP-Problem (job shop sheduling) und die industrielle Maschinenbelegungsplanug. Diese Abhandlung behandelt das zweite Problem, das wesentlich komplexer ist als das erstere, da es einen realen Produktionsprozeß abzubilden und zu lösen versucht.

Bei diesem Problem handelt es sich um ein kombinatorisches Problem, das zu der Klasse der NP - harten Probleme gehört und mit deterministischen Algorithmen nicht in polynomieller Zeit zu lösen ist. Aus diesem Grund und weil dieses Problem für die Wirtschaft von sehr großer Bedeutung ist, gab es schon ziemlich früh Versuche dieses Problem mit Hilfe von GAs zu lösen.

2 Problembeschreibung

2.1 Allgemeines

Was versteht man unter Ablaufplanung (in der Literatur auch als „scheduling problem" bekannt) ?

Unter Ablaufplanung versteht man den Prozeß der zeitlichen Zuordnung einer Menge gegebener Aufträge zur Verrichtung von Tätigkeiten (z.B. Herstellung eines Produkts) auf eine Menge von limitierten Ressourcen (z.B. Personal, Maschinen). Diese Planung ist gewissen Restriktionen (Nebenbedingungen) unterworfen, die vor allem aus zeitlichen Beziehungen zwischen den Tätigkeiten oder weiteren problemspezifischen Beschränkungen. Das Ergebnis dieses Planungsprozesses ist ein Ablaufplan (schedule), in dem jeder Tätigkeit ein Zeitintervall auf derjenigen Ressource zugeordnet ist, auf der die Tätigkeit durchgeführt werden soll. Ein Ablaufplan, der allen Nebenbedingungen genügt heißt zulässig. Das Ziel der Ablaufplanung ist das Finden eines zulässigen Ablaufplans, der hinsichtlich bestimmter Kriterien wie Kosten, Termineinhaltung oder Durchsatz möglichst optimal ist.

2.2 Formelle Beschreibung

Dieses Problem kann man formell folgendermaßen darstellen:

Gegeben ist ein Tupel (R, P, O, C), wobei R die Menge der Ressourcen, P die Menge der Produkte, O die Menge der Aufträge und C die Menge der Constraints (Nebenbedingungen) ist.

$R = \{R_1, ..., R_r\}$

$P = \{P_1, ..., P_p\}$

$O = \{O_1, ..., O_n\}$

$C = \{C_1, ..., C_c\}$

Für jeden Auftrag ist das herzustellende Produkt (P_j), der früheste Starttermin (r_j) und der gewünschte Fertigstellungstermin (Liefertermin) (d_j) gegeben.

$O_j=(P_j,r_j,d_j)$

Jedes Produkt kann auf mehrere verschiedene Arten (Produktionsvarianten / Rezepturen) hergestellt werden. Jeder dieser Produktionsvarianten ist wiederum in eine bestimmte Anzahl von Operationen (Herstellungsschritte) untergliedert.

$P_i =\{V_{i1}, ..., V_{iv}\}$	Produktionsvarianten für das Produkt P_i
$V_{ij}=\{Op_{ij1}, ..., Op_{ijy}\}$	Operationen für die Produktionsvariation V_{ij}

Zu jeder Operation ist die Menge der alternativen Maschinen gegeben, auf denen die Operation ausgeführt werden kann. Die Ausführungszeit einer Operation ist gleich der Länge des Zeitintervalls, das notwendig ist, um die Operation zu produzieren.

$Op_{ijk}=\{(R_{ijk1},e_{ijk1}), ..., (R_{ijkd},e_{ijkd})\}$

Eine Operation wird also durch einen Tupel (Maschine, Ausführungszeit) beschrieben.

Ralf Bruns verwendet in seinem Artikel folgende Constraints:

C1 - Alle Aufträge müssen vollständig ausgeführt werden

C2 - Jeder Auftrag wird durch die Ausführung genau einer Produktionsvariante des entsprechenden Produktes erfüllt

C3 - Die Ausführung einer Variante besteht aus der Ausführung aller ihrer Operationen

C4 - Eine Operation kann nur auf einer ihrer alternativen Maschinen ausgeführt werden.

C5 - Die Reihenfolge, in der die Operationen eines Auftrags ausgeführt werden müssen, ist durch die Präzedenzrelation festvorgegeben. Die Ausführung zweier Operationen des gleichen Auftrags darf sich hierbei nicht überlappen.

C6 - Die Starzeit einer Operation darf nicht kleiner sein als der früheste Starttermin des zugehörigen Auftrags

Um das Problem zu lösen wird noch ein Optimierungskriterium benötigt. Für die Praxis sind folgende Kriterien interessant:

- Minimierung der Durchlaufzeit

- Minimierung der Termineinhaltung

- Maximierung der Maschinenauslastung

- Minimierung des Energieverbrauchs

- Minimierung der Kosten

In seinem Artikel und in den darin beschriebenen Versuche wird nach dem zweiten Kriterium optimiert. Dafür verwendet Ralf Bruns folgende Bewertungsfunktion:

$$F(f_1,...,f_n) = \sum_{i=1}^{n} (\max(0, f_i - d_i))^2$$

Die Funktion beschreibt die Summe der quadrierten Terminverzüge, wobei

- d_i für den Liefertermin, d.h. den Fertigstellungszeitpunkt, und
- f_i für die tatsächlich benötigte Zeit für die Erfüllung des Auftrags O_i steht

Interessant wäre die Möglichkeit nach mehreren Kriterien zu optimieren, v.a. wenn diese Kriterien konkurrierend sind. Dies kann in der Praxis relativ oft vorkommen (z.B. Termintreue und minimaler Lagerbestand). (dieses Thema wird in seinem Artikel nicht behandelt).

3 GA-Basierte Verfahren

3.1 Überblick

Da das Maschinenbelegungsproblem eine wirtschaftlich sehr wichtige Problemstellung ist, hat man relativ früh versucht dieses Problem mit Hilfe von genetischen Algorithmen zu lösen.

An dieser Stelle möchte ich einen kurzen Überblick der vorhandenen GA geben, die zur Lösung dieses Problems verwendet wurden.

Gemäß ihrer zugrundeliegenden Lösungsrepräsentation kann man nach [Bagchi 91] zwei Klassen von genetischen Algorithmen definieren: die indirekte und die direkte Lösungsrepräsentation.

Bei der indirekten Repräsentation kann man darüberhinaus zwischen problemunabhängigen und problemspezifischen Formen der Lösungsdarstellung unterscheiden.

Die überwiegende Mehrzahl der GA für Ablaufplanungsprobleme basiert auf einer indirekten Repräsentation der Lösungskandidaten. Die Algorithmen arbeiten in diesem Fall (wie bei GA allgemein üblich) auf einer Population von codierten Lösungsalternativen (Genotypen).

In der problemunabhängigen indirekten Repräsentation ist keinerlei Wissen über das zu lösende Problem enthalten. Vorteil dieser Repräsentation: sie kann für verschiedene Problemstellungen eingesetzt werden (ohne Modifikation der Operatoren, da nur konventionelle Operatoren verwendet werden) Außer der Evaluierungskomponente (Decodierung und Fitneßfunktion) wird kein problemspezifisches Wissen verwendet.

Nun folgt die Liste der bisher eingesetzten Algorithmen:

- indirekte Repräsentation
  - problemunabhängig
    - a) binäre Repräsentation [Cleveland 19989], [Nakano 91]
    - b) Liste der Aufträge
  - problemspezifisch
    - c) Liste von (Auftrag,Varianten)-Tupeln [Bagchi 91]
    - d) Liste von (Auftrag,Varianten, Maschinen)-Tupel [Bagchi 91]
    - e) Präferenzlisten [Davis 85]
- direkte Repräsentation
    - f) Liste von (Auftrag, Maschine, Startzeit)-Tupel [Kanet 91]

Der bislang am häufigsten eingesetzte Repräsentationsformalismus repräsentiert eine Lösung durch die Liste der zu verplanenden Aufträge (c). Ein Individuum wird durch eine Liste von Aufträge repräsentiert. Die Reihenfolge der Aufträge in der Liste legt die Priorität ihrer Verplanung fest, so daß das Planungsproblem auf ein Reihenfolgeproblem (sequencing problem) reduziert wird (vergleichbar dem TSP – Traveling Salesman Problem). Für ein Individuum erzeugt die Decodierung den dazugehörenden Ablaufplan, indem die Aufträge entsprechend ihrer Reihenfolge verplant werden (alle Operationen des ersten Auftrags werden verplant, danach die Operationen des zweiten Auftrags usw.)

3.2 Ergebnis

GA wurden bis jetzt nur für Ablaufplanungsprobleme verwendet, die eine relativ einfache Problemstruktur haben und denen wenige Nebenbedingungen zugrunde liegen.(z.B. Flow-Shop- oder Job-Shop-Scheduling Probleme, 1-Maschinen-Probleme)

Empirische Vergleiche (von Bagchi) belegen, daß problemunabhängige GAs besonders gut für Ablaufplannungprobleme mit wenigen Nebenbedingungen geeignet sind. Er macht die Beobachtung, daß je mehr Wissen in der Repräsentation des GAs integriert wird, desto besser sind auch die Ergebnisse.

Außerdem führen die binäre Lösungsrepräsentation und die unstrukturierten Standard - Operatoren bei Problemen mit nichttrivialen Nebenbedingungen i.allg. zu neuen Lösungen, die bzgl. der Nebenbedingungen unzulässig sind. Dadurch wird der Suchraum vergrößert und die Effektivität nimmt dadurch ab.

4 Wissensbasierter GA

Der wissensbasierte GA funktioniert eigentlich wie ein Standard GA:

Zunächst generiert die wissensbasierte Initialisierung zufällig eine Anfangspopulation von zulässigen Lösungskandidaten. Die Individuen der Anfangspopulation werden bewertet, und jedem Individuum wird ein entsprechender Fitneßwert zugeordnet.

Die Selektion wählt stochastisch Individuen für die Erzeugung von Nachkommen aus.

Der wissensbasierte Crossover-Operator tauscht die genetische Information zwischen den selektierten Individuen aus und generiert so neue , zulässige Nachkommenlösungen.

Diese neuen Lösungen werden anschließend durch den wissensbasierten Mutations-operator leicht modifiziert, wobei jedoch deren Zulässigkeit gewahrt bleibt.

Die Güte der neuen Lösungen wird bestimmt und jedem neuen Individuum wird ein Fitneßwert zugeordnet.

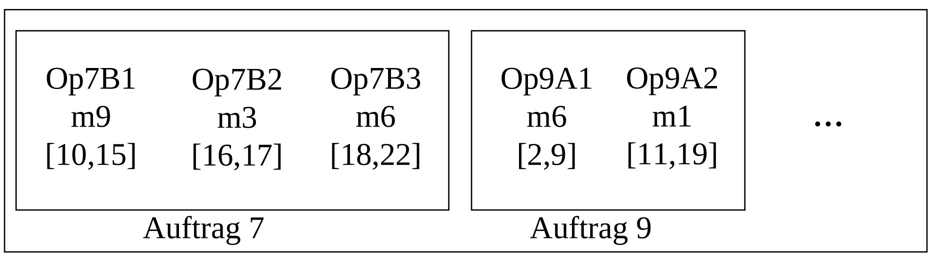
Dieser Vorgang wird solange iteriert, bis ein vorgegebenes Terminierungskriterium erfüllt wird.

Wichtig ist, daß während des gesamten Optimierungsprozesses ausschließlich zulässige Lösungen generiert werden. Dadurch, daß man immer in einem Suchraum von gültigen Lösungen operiert ist es denkbar, daß dieser Algorithmus effizienter ist.

Der Selektionsoperator braucht nicht geändert werden, da er nur die Fitneßwerte der Individuen benötigt.

```
initialization;
while not terminated do
  selection;
  advanced crossover;
  advanced mutation;
od;
```

Direkte Repräsentation eines Ablaufplans



4.1 Wissensbasierte Initialisierung:

Ziel der wissensbasierten Initialisierung ist die Erzeugung einer heterogenen Startpopulation, die vollständige und konsistente Ablaufpläne enthält. Dies kann durch konventionelle Algorithmen erfolgen (z.B. „Order-based“).

Um eine Anfangspopulation zu erzeugen werden zufällige Permutationen der Auftragsliste erzeugt. Daraus kann man konsistente und gültige Ablaufpläne erzeugen.

Diese Methode erzeugt zufällige und verschiedene Populationen von Ablaufpläne, da auch nur geringfügig verschiedene Auftragslisten zu komplett verschiedenen Ablaufpläne führen können.

Dabei wird aus der Menge der Aufträge ein Auftrag genommen und alle seine Operationen in den Ablaufplan hinzugefügt, dann wird der nächste Auftrag bearbeitet, usw..

4.2 Wissensbasierter Crossover:

Die Aufgabe des Crossover-Operators ist die Erzeugung einer neuen Nachkommenlösung durch die zufallsgesteuerte Kombination charakteristischer Strukturen der Elternlösungen. Dabei werden die guten Eigenschaften der Eltern übernommen. Teillösungen (Teilstrukturen) werden aus den Eltern extrahiert und zu einer neuen Lösung passend zusammengefügt. Die Auswahl bestimmter Teillösungen kann über heuristisches Wissen beeinflußt werden (z.B. um die Vererbung bestimmter Strukturen zu betonen)

Der verwendete Algorithmus in Pseudocode lautet:

```
Wähle zwei Ablaufpläne aus;

1. Wähle vom ersten Elternteil eine zufällige
   Anzahl an nicht verspäteten Aufträge aus;
   Diese Aufträge werden weitervererbt

2. Die fehlenden Aufträge werden vom zweiten
   Elternteil genommen und im neuen Ablaufplan
   eingefügt;

   a)zuerst werden, die nicht verspäteten Aufträge
    (mit aufsteigender Startzeit)eingefügt

   b)dann werden die verspäteten Aufträge eingefügt
```

Wenn eine vererbte Operation nicht zu dem spezifizierten Zeitpunkt eingefügt werden kann, dann wird sie in die Zukunft verschoben, bis ein freier Zeitintervall vorhanden wird. Die nachfolgenden Operationen werden natürlich auch verschoben.

4.3 Wissensbasierter Mutationsoperator:

Der wissensbasierte Mutationsoperator modifiziert zufallsgesteuert die genetische Information in einem Individuum. Hierbei soll der Operator in der Lage sein, einerseits alle repräsentierten Informationen zu ändern, andererseits aber auch alle überhaupt zulässigen Informationen vom neuen zu erzeugen (wichtig für die Wiedergewinnung von genetischem Material, welches im Laufe der Generationen verlorengegangen ist).

Dafür wurden drei Operatoren entwickelt die unabhängig voneinander sind, d.h. bevor die Mutation ausgeführt wird, entscheidet man zufällig welche dieser drei Operatoren verwendet wird.

In der Dissertation von Bruns steht ein etwas komplexerer Algorithmus, der zuerst eine Menge von Variablen aus dem Ablaufplan zufällig bestimmt und diese Variablen dann mutiert.

```
Wähle einen Ablaufplan aus;
Wähle einen zufälligen Auftrag aus;

 a)verändere den Produktionsplan für den
   ausgewählten Auftrag;

 b)wähle eine zufällige Operation und wähle eine
   alternative Maschine für diese Operation
  (falls diese Maschine frei ist)

 c)wähle eine zufällige Operation;
   verändere die Startzeit dieser Operation indem
   man sie auf die früheste zulässige Startzeit
   verschiebt.
```

4.4 Zusammenfassung

Der Hauptunterschiede zum GA ist die phänotypische Lösungsrepräsentation der Individuen. Die Individuen sind keine kodierten Ablaufpläne sondern sie beschreiben direkt den Ablaufplan. Da direkt auf den Lösungen gearbeitet wird, lassen sich die Lösungen einfacher und effizienter berechnen (Codierung und Decodierung nicht mehr notwendig, was häufig mit großem Aufwand verbunden ist). Außerdem benötigt man komplexere, problemspezifische Operatoren (Initialisierung, Crossover und Mutation). Der Selektionsopertor muß nicht geändert werden, da er weiterhin auf die Fitneßwerte der Individuen arbeitet.

Gearbeitet wird ausschließlich auf dem Raum der zulässigen Lösungen. Dadurch wird der zusätzliche Aufwand unzulässiger Lösungen vermieden, d.h. die Problematik der Bewertung unzulässiger Lösungen oder deren Reparatur entfällt.

Constraints und problemspezifische Heuristiken können bei dem wissensbasierten GA aktiv zur Steuerung der genetischen Suche ausgenutzt werden. In alle anderen Ansätze werden die Nebenbedingungen (d.h. das Problemwissen) i.allg. nur passiv verwendet, um die Zulässigkeit bereits generierter Lösungskandidaten zu überprüfen. Dieser Algorithmus verwendet das Wissen aktiv, d.h. beim Erzeugen der Lösungen (Crossover, Mutation). Dabei ist zu beachten, daß das eingesetzte heuristische Wissen einen entscheidenden Einfluß auf die Effizienz des Systems hat. Dieses Wissen kann das Systemverhalten im positiven aber auch im negativen Sinne beeinflussen). Daß die Leistung des GAs ist von heuristischem Wissen anhängig ist, könnte man als Nachteil betrachten.

4.5 Versuche / empirische Resultate

Die Versuche die R. Bruns gemacht hat, wurden mit Daten aus der Industrie gemacht, man hat also ein reales, existierendes Problem gehabt.

Für die Versuche gelten folgende Eckdaten:

- hergestellt werden sollen 53 Produkte
- für jedes Produkt existieren min. 3 Produktionsalternativen
- Pro Produktionsvariante existieren bis zu 19 Operationen
- Es stehen höchstens 12 Maschinenalternativen für eine Operation zur Verfügung

Für die ersten Versuche wurde eine Populationsgröße von 30 Individuen gewählt und eine proportionale Selektion verwendet. Die verwendete Mutationswahrscheinlichkeit war ungewöhnlich hoch (p_m=1).

Die Versuche wurden mit zwei verschiedenen Auftragsmenge (42 und 54 Aufträge) durchgeführt. Für jede Auftragsmenge wurden 50 Versuche gemacht, deren Ergebnisse gemittelt wurden.

Folgender Verlauf wurde beobachtet:

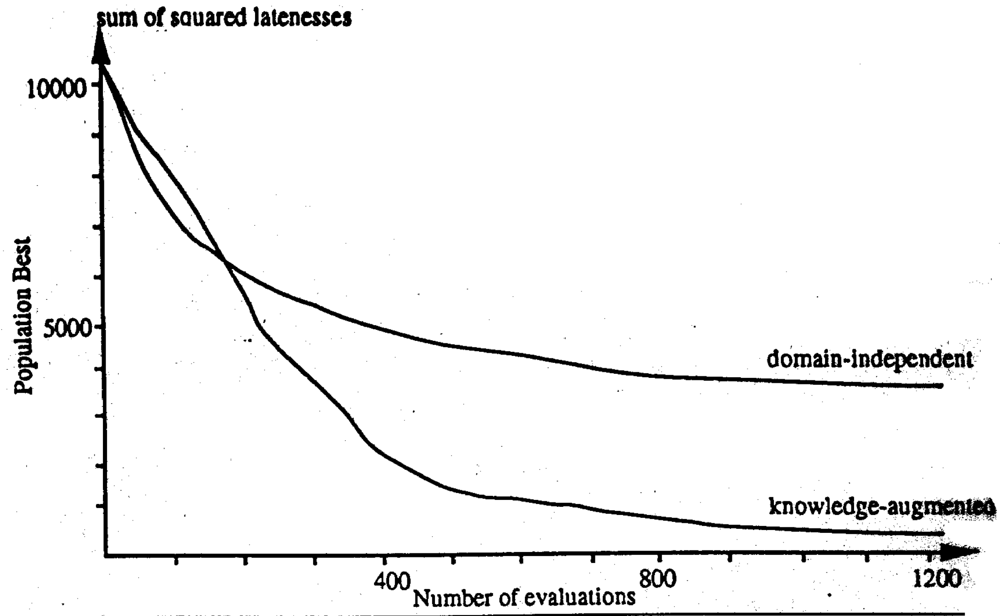


Figure 4: Typical Performance of Knowledge-Augmented and Domain-Independent Algorithm

Nachträglich wurden Versuche mit folgenden Variationen durchgeführt:

- verschiedene Populationsgrößen (20 - 100)

- verschiedene Selektionsmethoden (proportionale, rangbasierte)

- verschiedene Crossover und Mutationswahrscheinlichkeiten

- verschiedene Fitneßfunktionen (Summe der Terminverzüge, Summe der quadrierten Terminverzüge)

- verschiedene Initialisierungsalgorithmen (load-balancing, random, first-fit)

Die Ergebnisse dieser Versuche entsprachen den oben dargestellten Kurven.

Bruns macht außerdem folgende Beobachtung, daß wenn der Crossover- oder der Mutationsoperator weggelassen wird, so erhält man nur unbefriedigende Ergebnisse. Beide Operatoren sind also notwendig um qualitativ hochwertige Ablaufpläne erzeugen zu können.

Aus dem Buch „DISKI“ von Ralf Bruns stammen folgende zwei Diagramme:

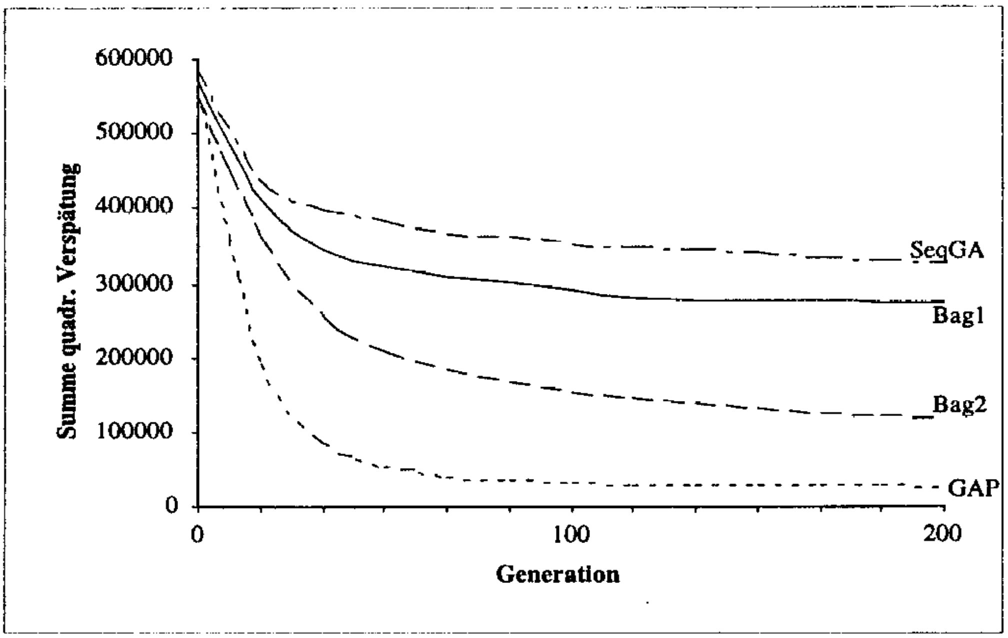


Abb. 5.9: Vergleich der Optimierungsverläufe der GA-Varianten

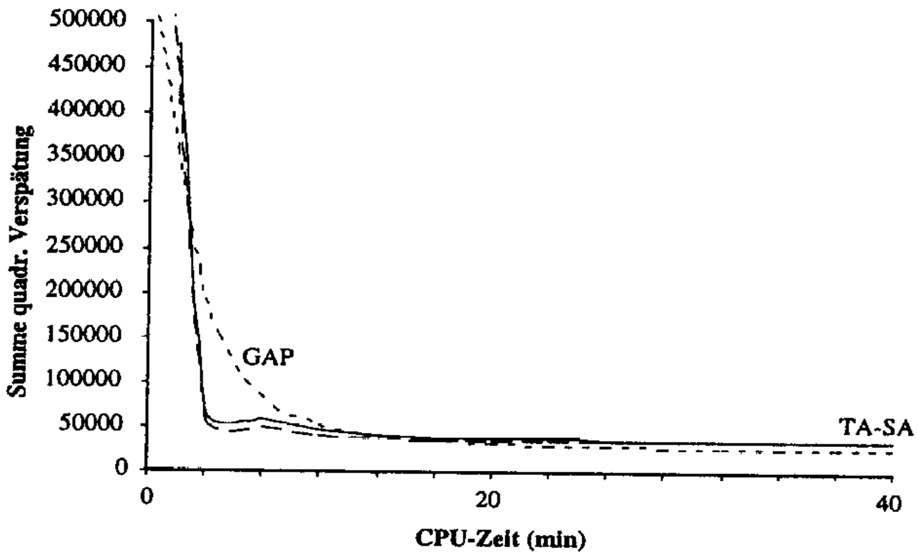


Abb. 5.10: Vergleich der Optimierungsverläufe von GAP - SA - TA

In Abbildung 5.9 ist ein ähnlicher Verlauf der Kurven zu beobachten. Der GAP (Genetic Algorithm + Constraint Programming) konvergiert schon nach wenigen Generationen und erreicht besseren Lösungen als die standard GAs.

SeqGA ist ein Reihenfolge-GA, (ähnlich dem Beispiel GA aus Kapitel 3). Bei Bag1 und Bag2 handelt es sich um problemspezifische GAs, die von [Bagchi 91] entwickelt wurden.

Die Abbildung 5.10 zeigt die benötigte CPU-Zeit. Der wissensbasierte Algorithmus (GAP) wird mit einem SA (Simulated Annealing) und einem TA (Threshold Accepting) verglichen. Beide Algorithmen konvergieren viel schneller als der GAP, erreichen aber nicht so gute Ergebnisse.

5 Zusammenfassung

Die wesentlichen Vorteile der wissensbasierten Repräsentation liegen in der realistischeren Abbildung der Problemstellung sowie in der Steuerung der Lösungsfindung auf Basis des repräsentierten Wissens. Sie sind dadurch anwendbar auf realistische Ablaufplanungsprobleme mit realen Mengengerüsten und spezifischen Nebenbedingungen. Diese Probleme konnten bisher nicht mit den standard GAs gelöst werden. Außerdem liefert der wissensbasierte GA für diese Problemstellung qualitativ bessere Ergebnisse durch die Integration von problemspezifischen Wissens in die Lösungsrepräsentation.

6 Diskussion: Ist dieser Algorithmus für das Problem der Fahrplanoptimierung geeignet ?

Um das Problem der Fahrplanoptimierung mit diesem Ansatz lösen zu können wird folgendes festgelegt:

Die Menge R der Ressourcen entspricht der Menge der Haltestellen bzw. der Menge der Haltepunkte. (was gewählt wird ist davon abhängig, wie Linien definiert wurden, d.h. wurde eine Linie als Liste von Haltestellen definiert, so entspricht die Resourcenmenge der Menge der Haltestellen). Für diese Ressourcen gelten ähnlich wie bei R. Bruns folgendes Constraint: eine Ressource, darf nicht von zwei Aufträge belegt gleichzeitig belegt werden.

Aufträge entsprechen den einzelnen Linienfahrpläne. Ein Linienfahrplan ist durch die Taktzeit, Startzeit an dem einen Ende der Linie und Startzeit an dem anderen Ende der Linie festgelegt. Falls das System auch die Linienverläufe optimieren soll, dann müssen die entsprechenden Haltepunkte, die die Linie beschreiben mit aufgenommen werden.

Produktionsvariationen existieren hier nicht, da für eine Linie nur ein Verlauf möglich ist. Denkbar ist, daß man für eine Linie verschieden Verläufe definiert, und diese im Sinne der Produktvariationen verwendet. Dies ist nur der Fall, wenn das Optimieren der Linienverläufe gewünscht wird.

Benötigt wird eine Bewertungsfunktion, die sowohl einen kompletten Fahrplan als auch den Fahrplan einer einzigen Linie bewerten kann. Die Fitneßfunktion, die von Stefan Lewandowski für das Studienprojekt angegeben wurde, könnte sich dazu eignen.

Der Algorithmus für den Crossover-Operator könnte wie folgt vorgehen:

```
Wähle zwei Fahrpläne aus;

1. Wähle vom ersten Elternteil eine zufällige
   Anzahl an guten Linienfahrpläne aus; (Die
   Bewertungsfunktion müßte angegeben, welche
   Linienfahrpläne „gut“ sind)
   Diese Fahrpläne werden weitervererbt

2. Die fehlenden Linienfahrpläne werden vom zweiten
   Elternteil genommen und im neuen Fahrplan
   eingefügt;

   a)zuerst werden, die guten Linienfahrpläne
    (mit aufsteigender Güte)eingefügt

   b)dann werden die schlechten Linienfahrpläne
     eingefügt
```

Dabei müssen die Schwellenwerte für einen guten oder einen schlechten Linienfahrplan experimentell ermittelt werden.

Der Algorithmus für den Mutationsoperator ist ähnlich wie bei R.Bruns und besteht aus vier unabhängigen Teile und könnte wie folgt ablaufen:

```
Wähle einen Fahrplan aus;
Wähle einen zufälligen Linienfahrplan aus;

 a)verändere alle Werte des Linienfahrplan;

 b)verändere die Taktzeit des Linienfahrplans;

 c)verändere die 1.Startzeit des Linienfahrplans;

 d)verändere die 2.Startzeit des Linienfahrplans;
```

7 Literatur

- Gleichnamiger Artikel von Ralf Bruns
- Solving Scheduling Problems via Evolutionary Methods
  Igor.P.Norenkov, Erik D. Goodman
- Genetic and Evolutionary Algorithms in the Real World
  David E. Goldberg
- Investigating Parallel Genetic Algorithms on Job Shop Scheduling
  Shyh-Chang Li, Erik D. Goodman, William F. Punch
- A Genetic Algorithm Approach To Dynamic Job Shop Scheduling
  Shyh-Chang Li, Erik D. Goodman, William F. Punch