UNIVERSITÄT LEIPZIG

Institut für Produktionswirtschaft
und Industrielle Informationswirtschaft

Marschnerstr. 31, 04109 Leipzig
Tel.: 0341 / 97-3355-0, Fax: -9

Arbeitsbericht Nr. 20

Feinterminierung unter restriktiven Laufzeitanforderungen - Ein exemplarischer Vergleich lokaler Suchverfahren (Teil I)

Jukka Siedentopf

⟨siedentopf@wifa.uni-leipzig.de⟩

Juli 1996

Alle Rechte vorbehalten.

Inhalt

1. Problemstellung **1**

2. Lokale Suchstrategien **2**

3. Die Basisstrategie **3**

3.1 Einplanung 3

3.2 Modifikation 4

3.3 Abschätzung der Lösungsqualität 4

3.4 Grobstruktur der Basis-LSS 5

4. Die Parametrisierung **6**

4.1 Simulated Annealing 6

4.2 Threshold Accepting 7

4.3 Great Deluge Algorithm 8

4.4 Threshold Waving 8

4.5 Fading Acceptance Probabilty 9

5. Ergebnisse **9**

5.1 Einfache Prioritätsregeln 10

5.2 Hillclimber 11

5.3 Wahrscheinlichkeits- und schwellenwertbasierte LSS 12

6. Fazit **14**

Literatur **15**

Zusammenfassung:

Gegenstand des Beitrags ist die Anwendung einiger ausgewählter lokaler Suchstrategien zur näherungsweisen Lösung eines kombinatorischen Optimierungsproblems, des allgemeinen Job-Shop-Scheduling-Problems, dessen (idealisierte) Struktur dem Problem der Feinterminierung in der Werkstattfertigung entspricht. Ziel ist es, die Leistungsfähigkeit der Strategien unter restriktiven Laufzeitanforderungen zu beobachten, wie sie etwa in interaktiven Planungsumgebungen gestellt werden. Als Basis dient eine generische lokale Suchstrategie, die gemäß fünf verschiedener Strategievarianten instanziiert wird. Die resultierenden Verfahren werden anhand einiger bekannter Testdatenbestände evaluiert, so daß auch Vergleiche mit den Ergebnissen anderer Autoren ermöglicht werden. Die dargestellten Tests besitzen sowohl aufgrund der eingeschränkten Größe der untersuchten Problemstellungen als auch hinsichtlich der Beschränkung auf (willkürlich gewählte) einzelne Parametrisierungen und lediglich ein Laufzeitniveau nur eine beschränkte Aussagekraft. Die Ergebnisse motivierten jedoch eine Fortsetzung der Tests, in der diesen Beschränkungen Rechnung getragen wird.

1. Problemstellung

Im Rahmen der Produktionsplanung und -steuerung stellen Probleme der Feinterminierung besondere Anforderungen an die Leistungsfähigkeit verfügbarer Planungsinstrumente, da Problemstellungen hoher Komplexität - deren optimale Lösung oft selbst unter Vernachlässigung sämtlicher Restriktionen hinsichtlich der verfügbaren Planungszeit nicht möglich ist - i.d.R. in einem „nervösen“ Umfeld ständig wechselnder Planungsnebenbedingungen gelöst werden müssen. Im folgenden wird eine idealisierte, besonders anspruchsvolle Variante der Feinterminierung betrachtet: Das Problem des Job-Shop-Scheduling (JSS), das einer allgemeinen Formulierung des Maschinenbelegungsproblems in der Werkstattfertigung entspricht und wie folgt charakterisiert werden kann:

- n Aufträge $\{A_1, A_2, \ldots, A_n\}$ sind auf m Maschinen $\{M_1, M_2, \ldots, M_m\}$ zu bearbeiten, die Bearbeitung eines Auftrags an einer Maschine wird auch als Operation oder Arbeitsgang bezeichnet. Für die hier untersuchten Probleme gilt, daß jeder Auftrag höchstens einmal an einer Maschine bearbeitet wird.
- Die (technologisch bedingten) Maschinenfolgen - die Reihenfolgen, in denen die einzelnen Aufträge die Maschinen durchlaufen müssen - sind linear und gegeben. Unterschiedliche Maschinenfolgen verschiedener Aufträge sind zulässig.
- Die Auftragsfolgen der Maschinen - die Reihenfolgen, in denen die Aufträge an den einzelnen Maschinen bearbeitet werden - sind dispositiv.
- Die Ausführungszeiten der Operationen sind gegeben und auftragsfolgenunabhängig. Es werden weder Rüst- noch Transportzeiten berücksichtigt, und eine Unterbrechung der Bearbeitung der Operationen ist nicht zulässig.
- Ziel ist die Festlegung der Auftragsfolgen, so daß die gegebenen Maschinenfolgen eingehalten werden und eine gegebene Zielfunktion optimiert wird. Als Zielsetzung wird im folgenden die Minimierung der Bearbeitungszeit des gesamten Auftragsbestands (Zykluszeit, Makespan) angestrebt.

Im weiteren werden zunächst die hier untersuchten lokalen Suchstrategien skizziert (Abschnitt 2). Nachfolgend wird eine generische Suchstrategie für das dargestellte JSS-Problem formuliert (Abschnitt 3) und anschließend deren Parametrisierung gemäß der zuvor skizzierten Varianten der lokalen Suche erläutert (Abschnitt 4). In Abschnitt 5 werden die Ergebnisse dargestellt, die mit den Verfahren in einem Test unter restriktiven Laufzeitanforderungen anhand einiger ausgewählter Testdatenbestände erzielt wurden.

2. Lokale Suchstrategien

Lokale Suchstrategien (LSS) dienen zur Steuerung iterativer Verbesserungsverfahren bei der näherungsweisen Lösung kombinatorischer Optimierungsprobleme. Iterative Verbesserungsverfahren basieren auf dem Prinzip, durch die Modifikation oder Mutation vorhandener Problemlösungskandidaten (LK) neue Lösungskandidaten zu erzeugen und anschließend eine Selektion derjenigen Kandidaten vorzunehmen, die weiterhin im Suchprozeß verbleiben und in einer folgenden Iteration als Grundlage der Generierung neuer LK dienen. Eine Lokalität der Suche wird dabei installiert, indem Modifikationen nur innerhalb einer - jeweils problemspezifisch zu definierenden - „Nachbarschaft“ eines LK zugelassen werden, und dient der Begrenzung des Suchraumes, da eine vollständige Enumeration aller LK i.a. aus Komplexitätsgründen ausscheidet.

Im folgenden werden ausschließlich unidirektionale (Single-Point-) Verfahren berücksichtigt, die jeweils genau einen aktuellen LK sowie genau einen aus diesem LK abgeleiteten (modifizierten) LK betrachten. Die Selektion reduziert sich in diesem Falle auf eine Entscheidung darüber, ob der modifizierte LK oder aber der jeweilige Ausgangs-LK als Basis der Generierung neuer LK dienen soll, also auf eine Entscheidung über die Akzeptanz (oder Ablehnung) einer Modifikation. Rein qualitätsabhängige Akzeptanzentscheidungen (Lösungsqualität als Selektionskriterium) führen i.d.R. zu einem vorzeitigen Terminieren der Suche in lokalen Optima, d.h. in Lösungen, in deren Nachbarschaft sich keine „besseren“ LK befinden. Um ein Verlassen lokaler Optima zu ermöglichen, werden daher insbesondere Suchstrategien vorgeschlagen, in denen die Akzeptanz neuer LK so gesteuert wird, daß partiell auch Verschlechterungen einer aktuellen Lösungsqualität zugelassen werden.

Es werden zum einen Strategien betrachtet, in denen die Akzeptanz modifizierter LK geringerer Lösungsqualität probabilistisch gesteuert wird, also mit einer bestimmten Wahrscheinlichkeit erfolgt: Beim *Simulated Annealing* [1,15] hängt die Wahrscheinlichkeit der Akzeptanz eines neuen LK sowohl von der Differenz seiner Lösungsqualität zu der des Ausgangs-LK als auch von einem externen Parameter („Temperatur“) ab, dessen Einfluß im Laufe des Verfahrens kontinuierlich abnimmt. Alternativ wird hier ein Verfahren vorgeschlagen, in dem die Wahrscheinlichkeit der Akzeptanz eines LK geringerer Qualität unabhängig von dieser Qualität ist und, ausgehend von einem vorgegebenen Startwert, im Laufe des Verfahrens kontinuierlich bis zu einem ebenfalls vorgegebenen Endwert reduziert wird (*Fading Acceptance Probability*).

Zum anderen wurden 3 Verfahren implementiert, in denen die Akzeptanz eines modifizierten LK durch dessen Lösungsqualität sowie einen Schwellenwert determiniert, also nicht probabilistisch ermittelt wird: Beim *Threshold Accepting* [9] wird die Differenz der Lösungsqualität eines LK zu der des Ausgangs-LK mit einem aktuellen Schwellenwert (Threshold) verglichen, und der modifizierte LK wird als neue Ausgangslösung akzeptiert, falls seine Verschlechterung gegenüber der Ausgangslösung den Threshold nicht übersteigt. Der Wert des Threshold sinkt

dabei im Laufe des Verfahrens. Das *Threshold Waving* [20] stellt eine Erweiterung des Threshold Accepting dar, bei dem der Schwellenwert während des Verfahrens „wellenförmig“ ab- und zunimmt, wobei die „Höhe“ (Amplitude) der Wellen im Laufe des Verfahrens reduziert wird. Im Unterschied zu diesen beiden differenzorientierten Varianten wird beim *Great Deluge Algorithm* (Sintflutalgorithmus) [8] ausschließlich der Absolutbetrag der Lösungsqualität des modifizierten LK mit einem aktuellen Threshold verglichen. Ein LK wird akzeptiert, falls seine Qualität unterhalb (Minimierungsproblem) oder oberhalb (Maximierungsproblem) des Thresholds liegt, dessen Wert im Laufe des Verfahrens einer Optimallösung angenähert wird (sinkender Threshold für Minimierungsprobleme, steigender Threshold für Maximierungsprobleme).

3. Die Basisstrategie

Die im vorangehenden Abschnitt angeführten LSS unterscheiden sich lediglich durch den jeweiligen Akzeptanzoperator sowie die zur Verfahrenssteuerung benötigte Parametrisierung. Es wurde daher eine Basis-LSS i.S. eines generischen Moduls implementiert, aus der durch eine entsprechende Initialisierung die jeweils benötigte LSS erzeugt wird. Als problemspezifischen Input erhält eine LSS Daten über die Problemgröße (Anzahl n der Jobs, Anzahl m der Maschinen) sowie Daten über die Problemstruktur in Form der Maschinenfolgen der Aufträge sowie der Bearbeitungszeiten der Operationen.

Ein Lösungskandidat wird durch einen Plan repräsentiert, der für jede Maschine eine vollständige Operationsfolge enthält. Die Evaluierung eines Plans erfolgt durch die zeitgenaue Einplanung aller Operationen gemäß der vorgeschlagenen Operationsfolgen unter Berücksichtigung der vorgeschriebenen Maschinenfolgen und resultiert in einem Schedule, der eine um die Start- und Endzeitpunkte der Ausführungen aller Operationen erweiterte Kopie eines Plans darstellt.

3.1 Einplanung

Zum Zwecke der Einplanung wird zum einen eine Liste geführt, die zu jeder Maschine den Auftrag enthält, der gemäß der vorgeschlagenen Reihenfolge als nächstes an dieser Maschine eingeplant werden soll. Zum anderen wird auf Basis der vorgeschriebenen Maschinenfolgen eine Liste verwaltet, die zu jedem Auftrag die Maschine angibt, auf der der Auftrag gemäß seiner Maschinenfolgen als nächstes bearbeitet werden muß. Die Einplanung eines Auftrages an einer Maschine erfolgt, wenn das entsprechende Auftrags-/Maschinen-Tupel in beiden Listen geführt wird. Anschließend werden beide Listen aktualisiert und die Planung fortgesetzt. Eine Einplanung erfolgt jeweils frühestmöglich, d.h. entweder direkt im Anschluß an die Ankunft des Auftrages an der Maschine oder - falls diese noch eine andere Operation bearbeitet - direkt im Anschluß an das Bearbeitungsende dieser Vorgängeroperation an der betreffenden Maschine. Es resultiert ein semiaktiver Schedule, in dem keine Operation unter Beibehaltung der vorge-

schlagenen Reihenfolgen zeitlich vorgezogen werden kann. Die Zulässigkeit eines Plan ist nicht grundsätzlich gewährleistet, d.h., eine Einplanung gemäß der vorgesehenen Auftragsfolgen unter Einhaltung der vorgeschriebenen Maschinenfolgen ist nicht zwingend möglich. Der Einplanungsalgorithmus enthält daher eine Repair-Komponente, die die Zulässigkeit durch möglichst geringe Änderungen des Plans wiederherstellt. Da durch die Anwendung eines speziellen Modifikationsoperators (s.u.) jedoch sichergestellt wird, daß ausschließlich zulässige Pläne erzeugt werden, wird auf eine Darstellung verzichtet (vgl. [19]).

3.2 Modifikation

Eine Modifikation wird als Austausch zweier Operationen in der Auftragsfolge einer Maschine realisiert. Aus der Menge potentieller Vertauschungen wird eine (kleine) Teilmenge herausgefiltert, für die gezeigt werden kann, daß sie sowohl ausschließlich Vertauschungen enthält, durch die zulässige Pläne generiert werden, als auch alle Vertauschungen, die zu einer Verbesserung des Zielfunktionswertes (Zykluszeit) führen. Ausgangspunkt ist eine zulässige, feingeplante Lösung (Schedule). Innerhalb des Schedules werden zunächst alle kritischen Operationen bestimmt, d.h. diejenigen Operationen, die nicht verzögert werden können, ohne die Zykluszeit des Schedules zu verlängern. Auf den einzelnen Maschinen werden (falls vorhanden) zeitlich direkt aufeinanderfolgende kritische Operationen zu kritischen Blöcken zusammengefaßt. Als Modifikation wird ausschließlich der Tausch zweier Operationen innerhalb eines kritischen Blocks zugelassen, von denen mindestens eine am Rand des kritischen Blocks liegt und mit einer nicht-kritischen Operation benachbart ist [6,18,21].

3.3 Abschätzung der Lösungsqualität

Sowohl die Evaluierung (Einplanung) als auch die Bestimmung des kritischen Pfades eines Plans gehören zu den „teuren“ (i.S. von laufzeitintensiven) Komponenten des iterativen Suchprozesses. Besonders nachteilig für die Effizienz der Suche wirken dabei „überflüssige“ Berechnungen für diejenigen LK, die wieder verworfen, also nicht akzeptiert werden. Im dargestellten Verfahren erfolgt daher eine Akzeptanzentscheidung im Anschluß an eine Modifikation nicht auf Basis einer exakt berechneten Lösungsqualität, sondern auf Basis eines Schätzwertes für eine untere Grenze der Zykluszeit [21]. Zur Berechnung dieses Schätzwertes werden als zusätzliche Informationen zum einen die frühestmöglichen Bearbeitungsstartzeitpunkte der Operationen (Heads) benötigt. Diese hängen ggfs. wiederum von den jeweiligen Bearbeitungsendzeitpunkten der Vorgängeroperation auf der betrachteten Maschine und der Vorgängeroperation in der Maschinenfolge des betrachteten Auftrags ab. Daneben werden die Zeitspannen benötigt, die zwischen den Bearbeitungsendzeitpunkten der Operationen und den Fertigstellungszeitpunkten der jeweils betreffenden Aufträge, also den Bearbeitungsendzeitpunkten der letzten Operationen der zugehörigen Aufträge, verbleiben (Tails). Aus den Heads und Tails einer gegebenen Lösung können die Heads und Tails berechnet werden, die zwei auf einer Ma-

schine direkt aufeinanderfolgende kritische Operationen nach einem Tausch aufweisen. Aus diesen neuen Heads und Tails kann wiederum eine untere Grenze für die Zykluszeit der modifizierten Lösung abgeleitet werden, die exakt ist, falls eine der getauschten Operationen auch nach dem Tausch kritisch ist. Eine exakte Ermittlung der Lösungsqualität sowie des kritischen Pfades eines Plans erfolgt jeweils nur, falls eine Modifikation auf Basis des resultierenden geschätzten Zielfunktionswertes akzeptiert wurde.

3.4 Grobstruktur der Basis-LSS

Der Ablauf der in Abb. 1 dargestellten Basis-LSS sieht im Anschluß an die verfahrensindividuelle Initialisierung die Erzeugung einer Startlösung durch die Anwendung einer Prioritätsregel, hier der Kürzesten-Operationszeit- oder Shortest-Processing-Time-Regel (KOZ oder SPT, Schritt (2) in Abb. 1), vor, die als temporäres Optimum gespeichert wird (3). Der kritische Pfad (4) sowie die Heads und Tails (5) dieser Lösung werden berechnet, und anschließend wird der iterative Suchprozeß gestartet, der bis zum Erreichen eines Terminierungskriteriums (13) wiederholt wird, das zwar grundsätzlich verfahrensindividuell ausgestaltet werden kann, in den betrachteteten Implementierungen jedoch einheitlich durch das Überschreiten einer maximalen Iterationszahl erfüllt wird.

Im einem Durchgang des iterativen Suchprozesses wird zunächst eine Modifikation des aktuellen LK *candidate* durchgeführt (6) sowie ein Schätzwert für die resultierende Lösungsqualität berechnet (7). Auf Basis dieses Wertes wird mittels des jeweiligen strategieindividuellen Akzeptanzoperators entschieden, ob der modifizierte LK als neuer Ausgangs-LK akzeptiert wird (8). Im Falle der Akzeptanz wird der exakte Wert der Lösungsqualität durch Einplanung ermittelt (9), wiederum der zur Durchführung folgender Modifikationen benötigte kritische Pfad ermittelt (10) sowie die zur Ermittlung entsprechender Zielfunktionsschätzwerte benötigten Heads und Tails berechnet (11). Ein neues temporäres Optimum der Lösungsqualität wird gegebenenfalls gespeichert (12), das Endekriterium geprüft (13) und entweder ein neuer Zyklus gestartet oder das temporäre Optimum als Lösung an das aufrufende System übermittelt (14).

Während einer Modifikation erfolgt die Auswahl der zu tauschenden Operationen weitgehend zufällig. Zugunsten der Diversifizierung der Suche wurde jedoch eine in Abb. 1 nicht dargestellte Besonderheit eingeführt: Die Wahl einer Maschine, auf der nach einer zulässigen Modifikation gesucht wird, erfolgt gemäß einer zufällig ermittelten Reihenfolge der Maschinen. Sobald diese Reihenfolge vollständig abgearbeitet wurde, wird sie - wiederum zufällig - neu festgelegt. Diese Vorgehensweise verhindert insbesondere kurze Zyklen weitgehend, in denen z.B. identische Modifikationen mehrmals hintereinander getestet oder aber auch gerade akzeptierte Modifikationen direkt wieder invertiert werden.

```
LOCAL_SEARCH_BASE_FOR_JSS
/* input:  # jobs, # machines, operation times, machine sequences of jobs,
           acceptance operator, strategy dependent initialization data */
/* output: schedule, makespan */
initialize                                          /* strategy dependent */   (1)
compute candidate, schedule(candidate), makespan(candidate)   /* SPT-Rule */   (2)
temp_optimum ← (schedule(candidate), makespan(candidate))                       (3)
compute critical_path(schedule(candidate))                                      (4)
compute heads&tails(schedule(candidate))                                        (5)
loop  new_candidate ← mutation(candidate)                                       (6)
      compute estimated_makespan(new_candidate)                                 (7)
      if accept(new_candidate) then                 /* strategy dependent */    (8)
            candidate ← new_candidate
            compute schedule(candidate), makespan(candidate)                    (9)
            compute critical_path(schedule(candidate))                         (10)
            compute heads&tails(schedule(candidate))                           (11)
            if makespan(candidate) < temp_optimum.makespan then
                  temp_optimum ← (schedule(candidate), makespan(candidate))    (12)
until termination_criterion                                                    (13)
return solution ← temp_optimum                                                 (14)
```

Abb. 1: Die Basis-LSS

4. Die Parametrisierung

Die Parametrisierung der getesteten Suchstrategien erfolgt unter der Prämisse, daß die zur Verfügung stehende Laufzeit durch Angabe der durchzuführenden Anzahl von Iterationen I begrenzt wird. Für den folgenden Vergleich wurde ein einheitlicher Wert von 40.000 Iterationen festgelegt, für den verschiedene Verfahrensvarianten in einigen ersten Tests ein (willkürlich) als akzeptabel bewertetes Antwortzeitverhalten von ca. 30 Sekunden zeigten. Die implementierten Verfahren wurden so parametrisiert, daß die resultierenden Temperatur-, Wahrscheinlichkeits- oder Schwellenwerte über diesem Intervall von 40.000 Iterationen ein jeweils durch verfahrensspezifische Höchst- und Mindestwerte beschriebenes Werteintervall vollständig durchlaufen.

4.1 Simulated Annealing

Das Simulated Annealing erhält als Parameter neben der Iterationszahl I die Werte für die Start- und für die Endtemperatur. Die Temperatur wird - im Gegensatz zu einer Reihe anderer Implementierungen - nach jeder Iteration gesenkt. Für eine Temperatur T_i nach der i-ten Iteration gilt $T_i = T_{(i-1)} \cdot f$, i = 1,...,I, und für die Endtemperatur gilt $T_{end} = T_{start} \cdot f^{\,I}$, wobei T_{start} den Wert der Starttemperatur, T_{end} den Wert der Endtemperatur und f den Faktor bezeichnet, mit dem die Temperatur nach jeder Iteration aktualisiert wird, also $f = \exp[(\ln(T_{end}) - \ln(T_{start})) / I]$. Die Akzeptanzwahrscheinlichkeit P berechnet sich aus der jeweils aktuellen Temperatur T so-

wie der „Energiedifferenz“ ΔE zwischen den Zielfunktionswerten C_{alt} der Ausgangslösung und C_{neu} der modifizierten Lösung als

$$P = \begin{cases} 1, \text{ falls } \Delta E \geq 0, \\ \exp(\Delta E / T) \text{ sonst.} \end{cases}$$

Die Bestimmung des Startwertes der Temperatur wurde auf Basis eines einfachen Tests vorgenommen: Für ein Testproblem wurde eine Initialisierungslösung gemäß der implementierten KOZ-Regel erzeugt, und anschließend wurden 10.000 zufällige Modifikationen dieser Lösung generiert. Der Startwert der Temperatur wurde für eine erste Variante SA1 anschließend so festgesetzt, daß gemäß obiger Aktzeptanzregel 95% aller generierten Modifikationen akzeptiert worden wären. Es resultierte ein (gerundeter) Temperaturwert von 75. Für eine zweite Variante SA2 wurde der Startwert der Temperatur hingegen so bestimmt, daß 95% derjenigen Modifikationen akzeptiert worden wären, die einen schlechteren Zielfunktionswert als die Ausgangslösung aufwiesen. Der so ermittelte Temperaturwert beträgt 580. Für T_{end} wurde jeweils ein Wert von 0,1 vorgegeben. Abb. 2 zeigt den Verlauf der Temperaturkurven für SA1 und SA2 (T_{SA1}, T_{SA2}) sowie die resultierenden Akzeptanzwahrscheinlichkeiten (P_{SA1}, P_{SA2}) für eine Modifikation, die einen um 10 Einheiten schlechteren Zielfunktionswert als ihre Ausgangslösung (auf beliebigem Qualitätsniveau) aufweist (ΔE = -10).

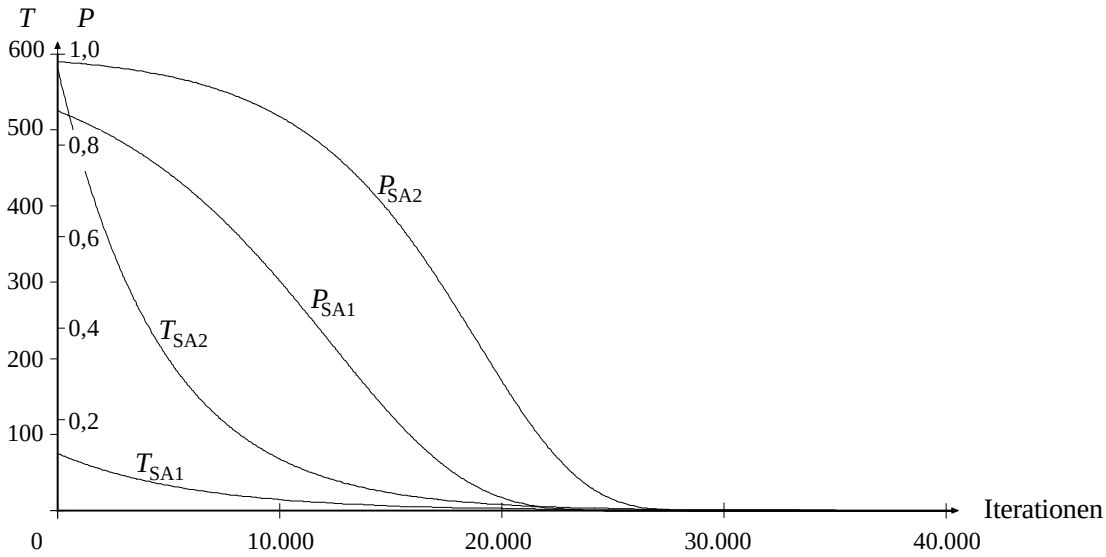

Abb. 2: Temperaturen (T) und Akzeptanzwahrscheinlichkeiten (P) für SA1 und SA2

4.2 Threshold Accepting

Das Threshold Accepting (TA) wird zusätzlich zur Iterationszahl I durch den Startwert des Thresholds TH_{start} initialisiert. Die Dekrementierung des Thresholds erfolgt jeweils um 1 Einheit bis auf einen Wert von 0. Die Anzahl *change_all* der Iterationen, nach denen jeweils eine

Dekrementierung erfolgt, wurde mittels der Gauß-Funktion DIV aus I und TH_{start} als *change_all* = I DIV TH_{start} + 1 berechnet. Der Wert für TH_{start} wurde auf 105 festgesetzt (*change_all* = 381).

4.3 Great Deluge Algorithm

Der Great Deluge Algorithm fällt eine Akzeptanzentscheidung durch einen Vergleich der Qualität einer modifizierten Lösung mit einer absoluten aktuellen „Qualitätsschranke". Um einerseits eine einheitliche Parametrisierung des Verfahrens für unterschiedliche Problemstellungen zu ermöglichen, andererseits jedoch zu verhindern, daß eine gleichförmige Dekrementierung der Schranke bis in Bereiche unterhalb der jeweils optimalen Zielfunktionswerte erfolgt, wird der Startwert der Qualitätsschranke mittels einer Konstanten *start_above* berechnet, die zu einer problemindividuellen minimalen Qualitätsschranke addiert wird. Die Qualitätsschranke wird im Laufe des Verfahrens bis auf diese minimale Qualitätsschranke dekrementiert, so daß der durchmessene Wertebereich für alle getesteten Probleme identisch ist.

Der Wert der minimalen Qualitätsschranke kann z.B. durch Berechnung einer unteren Schranke (Lower Bound) des Zielfunktionswertes des jeweiligen Problems, i.d.R. durch Lösung einer Relaxation des Problems, ermittelt werden (vgl. zur Berechnung von Lower Bounds z.B. [4]); um diesen Wert in realisierbare Bereiche abzubilden, kann er durch einen relativen oder absoluten Aufschlag korrigiert werden. In der implementierten Version wurde als minimale Qualitätsschranke vereinfachend das jeweils bekannte Optimum angesetzt. Der Wert *change_all* der Iterationen, nach denen eine Dekrementierung der Qualitätsschranke erfolgt, berechnet sich demgemäß als *change_all* = I DIV *start_above* + 1. GDA wurde zunächst in einer Variante GDA1 mit einem *start_above*-Wert von 320 (*change_all* = 125) sowie anschließend aufgrund der resultierenden Ergebnisse (vgl. Abschnitt 5) zusätzlich in einer Variante GDA2 mit einem *start_above*-Wert von 500 (*change_all* = 80) getestet.

4.4 Threshold Waving

Bei Anwendung des Threshold Waving (TW) wird der Schwellenwert - entsprechend der Ursprungsvariante des Threshold Accepting - ebenfalls nach einer festgelegten Anzahl von Iterationen *change_all* um eine Einheit modifiziert. Die Schwelle oszilliert dabei jeweils zwischen einer Obergrenze *th_upper_limit* und einer Untergrenze, die auf 0 gesetzt wird. Die Variable *th_upper_limit* wird zunächst mit dem Startwert des Thresholds TH_{start} initialisiert und jeweils um eine Einheit dekrementiert, wenn der Schwellenwert das Niveau 0 erreicht hat. Ziel der Parametrisierung ist es, daß das Verfahren terminiert, wenn *th_upper_limit* den Wert 0 annimmt. Die Anzahl A der Iterationen, die ein Lauf des TW benötigt, um die Oszillation zu beenden (*th_upper_limit* = 0), kann berechnet werden als $A = change\_all \cdot (TH_{start}^2 + 1)$. In der implementierten Version ($A = I$ = 40.000) wurde der Wert von TH_{start} auf 100 festgesetzt und ein

Wert von *change_all* als *change_all* = *I* DIV (TH^2_{start} + 1) + 1 berechnet, für die gegebenen Werte gilt *change_all* = 4. Den resultierenden Schwellenwertverlauf skizziert Abb. 3.

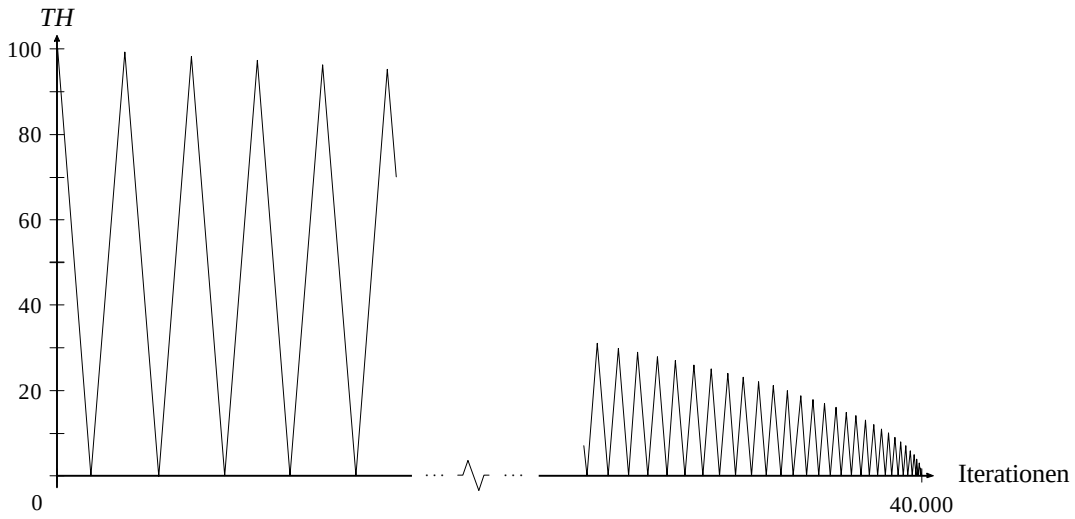

Abb. 3: Schwellenwertverlauf für TW

4.5 Fading Acceptance Probabilty

Das Verfahren Fading Acceptance Probabilty (FAP) wird durch Angabe einer Start- und einer Endwahrscheinlichkeit (*start_prob*, *end_prob*) für die Akzeptanz modifizierter Lösungskandidaten, die einen gegenüber ihrer Ausgangslösung verschlechterten Zielfunktionswert aufweisen, initialisiert. Die Akzeptanzwahrscheinlichkeit wird über das verfügbare Zeit-/Iterationsintervall *I* linear von *start_prob* auf *end_prob* dekrementiert, so daß sich das Dekrement *lower_step*, um das die Akzeptanzwahrscheinlichkeit nach jeder Iteration reduziert wird, berechnet als *lower_step* = (*start_prob* - *end_prob*) / *I*. Für die nachfolgenden Tests wurden die Werte 33% (*start_prob* = 0,33) sowie 1% (*end_prob* = 0,01) gewählt.

5. Ergebnisse

Als Testprobleme dienen zehn bekannte 10x10-Probleme (10 Aufträge, 10 Maschinen): Das Problem von Fisher und Thompson (FT3, [11]), die Probleme 5 und 6 von Adams, Balas und Zawack (ABZ5, ABZ6, [3]), die Probleme 19 und 20 von Lawrence (LAW19, LAW20, [16]) sowie die Probleme ORB1,...,ORB5 von Applegate und Cook [4]. Die Testverfahren wurden in Pascal codiert und alle Tests auf einem Pentium-PC/90 MHz durchgeführt. Die im folgenden angegebenen Resultate basieren auf Testserien von jeweils 20 Läufen pro Verfahren und Problem.

5.1 Einfache Prioritätsregeln

Tab. 1 vermittelt zunächst einen Eindruck der Lösungsqualität, die bei Anwendung einfacher prioriätsregelgesteuerter Heuristiken erzielt wird: Als Einplanungsalgorithmus dient das Verfahren von Giffler und Thompson [12], gegebenenfalls auftretende Einplanungskonflikte werden alternativ durch die Kürzeste-Operationszeit-Regel (KOZ), die Längste-Operationszeit-Regel (LOZ), die Kürzeste-Restbearbeitungszeit-Regel (KRBZ), die Längste-Restbearbeitungszeit-Regel (LRBZ) oder durch eine Modifikation der KOZ-Regel gelöst, bei der in Konfliktfällen an einer Entscheidung gemäß der KOZ-Regel gegebenenfalls nur solche Operationen teilnehmen, deren Wartezeiten vor der jeweiligen Maschine eine vorgegebene Schranke übersteigen (KOZ/WZ); die angegebenen Resultate wurden für eine Wartezeitschranke von 100 Zeiteinheiten ermittelt. Zusätzlich wurde das Verfahren in einer Version implementiert, in der Konflikte immer zufällig gelöst werden (RANDOM). Angegeben werden jeweils die durchschnittliche Lösungsqualität (Zykluszeit) sowie deren Abweichung von der optimalen Lösung des jeweiligen Problems. Ausgewiesene Nachkommastellen der durchschnittlichen Lösungsqualität deuten darauf hin, daß die Anwendung einer Regel keine eindeutigen Resultate liefert, sondern wiederum Konflikte auftreten, die dann durch eine Zufallsauswahl gelöst werden. Die ermittelten Lösungsqualitäten, die zwischen ca. 10% (KOZ/WZ-Regel für ABZ6) und über 52% (LOZ-Regel für ORB2) vom jeweiligen Optimum abweichen, zeigen große Verbesserungspotentiale auf.

		RANDOM	KOZ	LOZ	KRBZ	LRBZ	KOZ/WZ
ABZ5	C_{av}	1592,83	1393	1560,12	1554,17	1658,47	1486
(*1234)	$\Delta_{(av\text{-}opt)}$ [%]	29,08	12,88	26,43	25,95	34,40	20,42
ABZ6	C_{av}	1097,89	1039	1293	1126,59	1162,03	1039
(*943)	$\Delta_{(av\text{-}opt)}$ [%]	16,43	10,18	37,12	19,47	23,23	10,18
LAW19	C_{av}	1049,52	991	1145	1091,98	1126,64	991
(*842)	$\Delta_{(av\text{-}opt)}$ [%]	24,65	17,70	35,99	29,69	33,81	17,70
LAW20	C_{av}	1132,37	1182	1209	1203,25	1233,99	1049
(*902)	$\Delta_{(av\text{-}opt)}$ [%]	25,54	31,04	34,04	33,40	36,81	16,30
FT3	C_{av}	1319,35	1089	1356,95	1291,17	1272,71	1184,56
(*930)	$\Delta_{(av\text{-}opt)}$ [%]	41,87	17,10	45,91	38,84	36,85	27,37
ORB1	C_{av}	1413,08	1522,00	1458	1454,61	1460,33	1326,26
(*1059)	$\Delta_{(av\text{-}opt)}$ [%]	33,44	43,72	37,68	37,36	37,90	25,24
ORB2	C_{av}	1149,55	1047	1351	1246,74	1258,22	1139
(*888)	$\Delta_{(av\text{-}opt)}$ [%]	29,45	17,91	52,14	40,40	41,69	28,27
ORB3	C_{av}	1392,18	1178	1510	1370,85	1378,45	1296,01
(*1005)	$\Delta_{(av\text{-}opt)}$ [%]	38,53	17,21	50,25	36,40	37,16	28,96
ORB4	C_{av}	1405,58	1407	1392,40	1409,19	1518,50	1338,76
(*1005)	$\Delta_{(av\text{-}opt)}$ [%]	39,86	40,00	38,55	40,22	51,09	33,21
ORB5	C_{av}	1160,14	1078,20	1184,15	1188,28	1215,66	1091
(*887)	$\Delta_{(av\text{-}opt)}$ [%]	30,79	21,56	33,50	33,97	37,05	23,00

Legende: C_{av} : durchschnittliche Zykluszeit
* : optimale Lösung (minimale Zykluszeit)
$\Delta_{(av\text{-}opt)}$ [%] : durchschnittliche Abweichung vom Optimum in Prozent

Tab. 1: Lösungsqualitäten des Giffler/Thompson-Algorithmus

5.2 Hillclimber

Eine deutliche Verbesserung der Lösungsqualitäten kann bereits mit elementaren iterativen Verbesserungsverfahren erzielt werden, die keinerlei Techniken zur Überwindung lokaler Optima nutzen, sondern ausschließlich verbesserungsorientiert arbeiten. Tab. 2 enthält Ergebnisse für 4 Varianten derartiger Hillclimbing-Verfahren: Eine Initialisierung erfolgt dabei entweder durch eine zufällig erzeugte Lösung (Subskript random) oder durch eine mit der KOZ-Regel erzeugte Lösung (Subskript KOZ). Die Variante HC/BEST wählt aus der Nachbarschaft eines LK - also der Menge aller zulässigen Modifikationen - immer diejenige Modifikation, die den besten Zielfunktionswert liefert; das Verfahren terminiert, wenn der Zielfunktionswert eines LK nicht verbessert werden kann. Die Variante HC/NEXT akzeptiert aus der Nachbarschaft eines LK hingegen jeweils die erste Modifikation, die einen besseren Zielfunktionswert liefert - unabhängig davon, ob andere Modifikationen zu einem noch besseren Zielfunktionswert führen würden; das Verfahren terminiert ebenfalls, wenn keine Verbesserung des Zielfunktionswertes möglich ist.

Während die Laufzeiten für die gewählten - gegenüber praxisrelevanten Größenordnungen ausserordentlich kleinen - Testprobleme bei Anwendung des Giffler/Thompson-Algorithmus unterhalb von 10 Millisekunden liegen, steigen sie für die elementaren lokalen Suchverfahren bis auf ca. 20-120 Millisekunden an, wobei die Laufzeit wesentlich von der Anzahl der akzeptierten Modifikationen abhängt.

		$HC/BEST_{random}$	$HC/NEXT_{random}$	$HC/BEST_{KOZ}$	$HC/NEXT_{KOZ}$
ABZ5	C_{av}	1392,43	1388,31	1342	1326,21
(*1234)	$\Delta_{(av-opt)}$ [%]	12,84	12,50	8,75	7,47
ABZ6	C_{av}	1047,24	1044,13	984	985,33
(*943)	$\Delta_{(av-opt)}$ [%]	11,05	10,72	4,35	4,49
LAW19	C_{av}	981,32	979,68	936	925,23
(*842)	$\Delta_{(av-opt)}$ [%]	16,55	16,35	11,16	9,88
LAW20	C_{av}	1090,73	1113,06	1064	1056,81
(*902)	$\Delta_{(av-opt)}$ [%]	20,92	23,40	17,96	17,16
FT3	C_{av}	1179,47	1176,36	998	1020,01
(*930)	$\Delta_{(av-opt)}$ [%]	26,82	26,49	7,31	9,68
ORB1	C_{av}	1314,63	1318,04	1332	1332
(*1059)	$\Delta_{(av-opt)}$ [%]	24,14	24,46	25,78	25,78
ORB2	C_{av}	1044,44	1062,43	1035	1035
(*888)	$\Delta_{(av-opt)}$ [%]	17,62	19,64	16,55	16,55
ORB3	C_{av}	1321,87	1318,33	1146	1146
(*1005)	$\Delta_{(av-opt)}$ [%]	31,53	31,18	14,03	14,03
ORB4	C_{av}	1242,47	1236,24	1317	1286,05
(*1005)	$\Delta_{(av-opt)}$ [%]	23,63	23,01	31,04	27,97
ORB5	C_{av}	1071,70	1072,76	1024,51	1029,82
(*887)	$\Delta_{(av-opt)}$ [%]	20,82	20,94	15,50	16,10

Legende: C_{av} : durchschnittliche Zykluszeit
* : optimale Lösung (minimale Zykluszeit)
$\Delta_{(av-opt)}$ [%] : durchschnittliche Abweichung vom Optimum in Prozent

Tab. 2: Lösungsqualitäten der Hillclimbing-Verfahren

5.3 Wahrscheinlichkeits- und schwellenwertbasierte LSS

Die in Tab. 3 angeführten Ergebnisse für die in Abschnitt 2 und 3 skizzierten Varianten lokaler Suchstrategien demonstrieren, daß durch deren Anwendung wiederum eine signifikante Verbesserung der erzielten Lösungsqualitäten erreicht wird. Ein Vergleich der angeführten Resultate zeigt, daß das Verfahren FAP dabei insofern deutlich dominiert, als es für alle Testprobleme die besten durchschnittlichen Lösungswerte (C_{av}) liefert und bezüglich der besten jeweils erreichten Einzelergebnisse (C_{best}) lediglich einmal übertroffen wird (SA1 für ABZ6). Eine Rangfolge gemäß der erreichten durchschnittlichen Lösungsqualität würde - über alle Probleme gemittelt - die weiteren Verfahren in der Reihenfolge SA2, TW, SA1, GDA2 sowie (gemeinsam) TA und GDA1 reihen. Allerdings erscheint es angesichts der divergenten Einzelergebnisse zumindest fragwürdig, eine derartige Rangfolgenbildung vorzunehmen. Deutlich herausgestellt werden kann demgegenüber, daß alle Verfahren in der Lage sind, die zuvor erzielten Ergebnisse deutlich zu verbessern. Eine Ausnahme bilden zwei Ergebnisse des Verfahrens GDA1, die ein Problem schwellenwertorientierter Verfahren aufdecken: Für ORB1 und ORB4 kann das Verfahren GDA1 die Lösungsqualitäten gegenüber der Startlösung nicht (ORB4) oder nur in einzelnen Fällen geringfügig (ORB1) verbessern und schneidet sogar erheblich schlechter als die elementaren Hillclimber ab. Da die Ergebnisse der Hillclimber zeigen, daß sich in den Nachbarschaften der Ausgangslösungen Kandidaten besserer Qualität befinden, folgt, daß GDA1 diese Lösungen nicht entdeckt. Eine mutmaßliche Erklärung ist, daß das Verfahren regelmäßig zunächst schlechtere Lösungen akzeptiert und dabei lokale Optima erreicht, die aufgrund eines zu geringen Schwellenwertes nicht verlassen werden können. Die - zunächst widersprüchlich anmutende - Möglichkeit, durch Akzeptanz eines LK geringerer Qualität ein lokales Optimum zu erreichen, ist dabei auf den verwendeten nicht-symmetrischen Modifikationsoperator zurückzuführen, der die Umkehrung einer Modifikation nicht notwendigerweise gestattet. Durch eine Erhöhung der Startschwelle um 180 Einheiten in Version GDA2 wird die dargestellte „Pathologie“ (für die Testprobleme) zwar beseitigt, jedoch deuten die deutlich höheren Laufzeiten, die einen erhöhten Anteil akzeptierter Modifikationen signalisieren, bei gleichzeitig nur geringfügig verbesserter durchschnittlicher Lösungsqualität an, daß das Konvergenzverhalten des Verfahrens GDA gegenüber den anderen angeführten Verfahren als schwach zu bewerten ist.

		SA1	SA2	TA	GDA1	GDA2	TW	FAP
ABZ5	C_{av}	1245,95	1245,70	1250,95	1251,15	1249,55	1247,95	1242,35
	C_{best}	1238	1238	1238	1238	1242	1238	1238
(*1234)	C_{worst}	1255	1255	1266	1268	1263	1285	1252
	$\Delta_{(av\text{-}opt)}$ [%]	0,97	0,95	1,37	1,39	1,26	1,13	0,68
	σ	6,01	5,08	6,14	5,51	4,13	10,92	4,89
	t_{av}	24,31	28,50	30,82	28,31	43,30	27,93	30,39
ABZ6	C_{av}	953,40	951,25	960,35	956,40	953,10	957,60	949,35
	C_{best}	945	947	947	947	946	947	947
(*943)	C_{worst}	974	966	975	972	960	979	966
	$\Delta_{(av\text{-}opt)}$ [%]	1,10	0,87	1,84	1,42	1,07	1,55	0,67
	σ	7,76	5,98	7,25	5,89	5,15	9,35	5,05
	t_{av}	17,49	26,13	28,34	31,52	46,16	23,82	27,19
LAW19	C_{av}	855,45	859,55	860,00	862,00	863,05	853,65	848,95
	C_{best}	*842	850	850	*842	854	*842	*842
(*842)	C_{worst}	872	872	873	879	873	866	860
	$\Delta_{(av\text{-}opt)}$ [%]	1,60	2,08	2,14	2,38	2,50	1,38	0,83
	σ	7,50	6,79	5,77	9,26	5,39	6,84	6,01
	t_{av}	16,01	25,69	35,11	32,28	47,31	28,39	27,71
LAW20	C_{av}	915,25	908,45	915,95	917,30	916,80	913,70	905,45
	C_{best}	907	*902	*902	909	907	*902	*902
(*902)	C_{worst}	966	920	940	946	925	959	912
	$\Delta_{(av\text{-}opt)}$ [%]	1,47	0,72	1,55	1,70	1,64	1,30	0,38
	σ	12,12	5,45	8,10	8,43	4,73	11,06	3,71
	t_{av}	16,75	26,11	32,29	32,26	45,34	25,61	27,09
FT3	C_{av}	967,70	964,75	974,65	983,95	976,30	968,90	956,20
	C_{best}	949	939	954	952	955	952	*930
(*930)	C_{worst}	992	1010	992	1009	992	999	971
	$\Delta_{(av\text{-}opt)}$ [%]	4,05	3,74	4,80	5,80	4,98	4,18	2,82
	σ	10,09	15,13	11,71	14,57	10,61	11,30	9,62
	t_{av}	19,61	27,14	38,90	28,48	44,01	29,58	28,87
ORB1	C_{av}	1103,25	1101,40	1109,70	1504,20	1112,95	1113,80	1087,90
	C_{best}	1073	1071	1079	1479	1078	1087	1067
(*1059)	C_{worst}	1154	1123	1146	1563	1149	1203	1103
	$\Delta_{(av\text{-}opt)}$ [%]	4,18	4,00	4,79	42,04	5,09	5,17	2,73
	σ	18,05	13,14	19,29	38,49	14,96	26,40	10,28
	t_{av}	17,52	26,53	35,76	2,06	42,41	28,82	29,78
ORB2	C_{av}	910,20	909,00	918,30	915,95	917,70	907,25	902,60
	C_{best}	896	892	904	901	907	893	891
(*888)	C_{worst}	924	927	934	935	935	933	919
	$\Delta_{(av\text{-}opt)}$ [%]	2,50	2,36	3,41	3,15	3,34	2,17	1,64
	σ	7,61	9,46	7,93	9,89	7,82	9,97	7,54
	t_{av}	18,38	27,14	33,61	32,45	45,33	30,47	28,82
ORB3	C_{av}	1052,45	1061,10	1076,45	1063,00	1073,85	1058,80	1041,10
	C_{best}	1029	1023	1045	1032	1047	1040	*1005
(*1005)	C_{worst}	1076	1117	1108	1101	1095	1081	1071
	$\Delta_{(av\text{-}opt)}$ [%]	4,72	5,58	7,11	5,77	6,85	5,35	3,59
	σ	11,43	21,53	12,93	15,93	13,07	9,90	19,92
	t_{av}	17,86	26,44	35,27	25,41	41,26	30,75	29,24
ORB4	C_{av}	1053,55	1040,00	1039,30	1407	1038,15	1035,55	1028,00
	C_{best}	1019	1012	1013	1407	1015	1012	1012
(*1005)	C_{worst}	1089	1068	1073	1407	1070	1064	1055
	$\Delta_{(av\text{-}opt)}$ [%]	4,83	3,48	3,41	40,00	3,30	3,04	2,29
	σ	18,80	16,07	13,05	0,00	13,17	14,14	9,86
	t_{av}	18,51	27,36	34,62	1,69	47,14	29,52	29,72
ORB5	C_{av}	945,35	919,35	924,80	926,05	911,70	914,50	903,85
	C_{best}	908	895	902	894	895	898	891
(*887)	C_{worst}	978	948	943	999	934	940	941
	$\Delta_{(av\text{-}opt)}$ [%]	6,58	3,65	4,26	4,40	2,78	3,10	1,90
	σ	19,39	14,09	13,01	22,62	9,88	14,53	11,94
	t_{av}	17,78	25,38	38,06	28,92	46,42	30,79	27,04

Legende: C_{av} : durchschnittliche Zykluszeit; C_{best} : beste ermittelte Zykluszeit;
* : optimale Lösung (minimale Zykluszeit); C_{worst} : schlechteste ermittelte Zykluszeit;
t_{av} : durchschnittliche Laufzeit in Sekunden; σ : Standardabweichung der Zykluszeit;
$\Delta_{(av\text{-}opt)}$ [%] : durchschnittliche Abweichung vom Optimum in Prozent;

Tab. 3: Resultate für die lokalen Suchstrategien

6. Fazit

Die in den dargestellten Tests erzielten Ergebnisse motivieren eine Anwendung lokaler Suchstrategien zur näherungsweisen Lösung betrieblicher kombinatorischer Optimierungsprobleme, da mit ihrer Hilfe die Lösungsqualität einfacher, gut eingeführter Heuristiken signifikant gesteigert werden kann. Der Nachteil fehlender allgemeingültiger Evaluierungstechniken insbesondere für eine Anwendung unter „harten“ Zeitconstraints verliert bei einer Hybridisierung bereits eingesetzter Verfahren mit einfachen Suchstrategien an Bedeutung, da eine Beibehaltung des Lösungsniveaus dieser Verfahren gesichert ist. Darüber hinaus bietet der Einsatz der Suchstrategien den großen Vorteil, ohne Mindestanforderungen an verfügbare Laufzeiten auszukommen und somit annähernd beliebige verfügbare Rechenzeiten zur Verbesserung einer Problemlösung nutzen zu können. Aus Anwenderperspektive ist es dabei erfreulich, daß ein besonders einfach zu parametrisierendes Verfahren (FAP) dominierte; dieses Ergebnis wurde inzwischen in einigen weiteren Studien für Laufzeiten bis zu ca. 10 Minuten bestätigt.

Die dargestellten Tests besitzen sowohl aufgrund der eingeschränkten Größe der untersuchten Problemstellungen als auch hinsichtlich der Limitierung auf jeweils maximal zwei Parametrisierungsvarianten und genau ein Laufzeitniveau nur eine beschränkte Aussagekraft. Sie dienten insbesondere der Vorauswahl derjenigen Verfahren, die weiterführenden Test unterzogen werden, um den Einfluß der verfahrensindividuellen Parameter auf die Leistungsfähigkeit der Verfahren unter unterschiedlichen Laufzeitanforderungen und für unterschiedliche Problemgrößen zu untersuchen. Auch die Dokumentation der Tests wird dementsprechend fortgeführt.

Abschließend sei auf zwei lokale Suchstrategien hingewiesen, die im vorliegenden Test nicht berücksichtigt wurden: Ausgeschlossen wurden zum einen multidirektionale Verfahren wie z.B. genetische Algorithmen, in denen Populationen (i.S. mehrelementiger Mengen) von Lösungskandidaten verwaltet werden. Auf Grundlage von Populationen können neue Modifikationsoperatoren formuliert werden, die neue Lösungskandidaten aus den „Bausteinen“ bereits vorhandener Kandidaten generieren (Rekombinationen). Obgleich mit multidirektionalen Verfahren bei der Behandlung des JSS gute Ergebnisse hinsichtlich der erzielbaren Lösungsqualität erreicht werden [7,10], leidet das Laufzeitverhalten i.d.R. unter der teuren Vervielfachung potentieller Problemlösungskandidaten. Ein Nachweis, daß der Nachteil dieser Vervielfachung durch einen (Lösungs-) Qualitätsgewinn durch Anwendung der Rekombination „wettgemacht“ wird, steht bisher aus. Ein unbestrittener Vorteil multidirektionaler Verfahren ist jedoch das inhärente Parallelisierungspotential durch die weitgehend einheitliche und unabhängige Bearbeitung einzelner Populationsmitglieder (Individuen), das ihren Einsatz in parallelen DV-Umgebungen nahelegt [5].

Unberücksichtigt blieb ebenso das - wie die getesteten Verfahren unidirektional arbeitende - Verfahren des *Tabu Search* [13,14], obwohl aktuelle Forschungsarbeiten dieses Verfahren als die z.Z. leistungsfähigste lokale Suchstrategie sowohl hinsichtlich der erzielbaren Lösungs-

qualität als auch insbesondere hinsichtlich der benötigten Laufzeiten ausweisen [6,18,21]. Grund für die Vernachlässigung im dargestellten Test war, daß einige Tests mit einem Tabu-Search-Verfahren auf eine außerordentliche Sensibilität des Verfahrens bzgl. der Parametrisierung hindeuteten (vgl. dazu z.B. auch [18]). Tabu-Search-Verfahren durchsuchen darüber hinaus die Nachbarschaft einer Ausgangslösung i.d.R. vollständig, so daß ihre Effizienz weitaus enger mit der Größe der Nachbarschaft eines Lösungskandidaten korreliert als die der hier getesteten Verfahren. Da bei einer Adaption lokaler Suchstrategien für praktische Scheduling-Probleme gerade die Übertragbarkeit der auf „idealen“ Strukturen beruhenden Nachbarschaftskonstrukte sehr fraglich ist - dies gilt auch für das im vorliegenden Test verwendete Nachbarschaftskonstrukt - wird das Laufzeitverhalten einer Tabu-Search-Variante für unterschiedliche Nachbarschaftskonstrukte z.Z. untersucht, so daß auf eine Berücksichtigung im dargestellten Test verzichtet wurde.

Für einen weiterführenden Vergleich der im vorliegenden Beitrag vorgestellten Verfahren(svarianten) mit den Ergebnissen anderer Autoren sei z.B. auf die Arbeiten [2,3,4,6,7,10, 17,18,22] verwiesen, in denen ebenfalls heuristische Verfahren für das JSS u.a. anhand der hier gewählten Problemstellungen evaluiert werden.

Literatur

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

[2] Aarts, E.H.L., Van Laarhoven, P.J.M., Lenstra, J.K., Ulder, N.L.J.: A Computational Study of Local Search Algorithms for Job Shop Scheduling, ORSA Journal on Computing 6 (1994) 2, pp. 118-124.

[3] Adams, J., Balas, E., Zawack, D.: The Shifting Bottleneck Procedure for Job Shop Scheduling, Management Science 34 (1988) 3, pp. 391-401.

[4] Applegate, D., Cook, W.: A Computational Study of the Job-Shop Scheduling Problem, ORSA Journal on Computing 3 (1991) 2, pp. 149-156.

[5] Bierwirth, C.: Flowshop Scheduling mit parallelen Genetischen Algorithmen - Eine problemorientierte Analyse genetischer Suchstrategien, Wiesbaden 1993.

[6] Dell’Amico, M., Trubian, M.: Applying tabu search to the job-shop scheduling problem, Annals of Operations Research (1993) 41, pp. 231-252.

[7] Dorndorf, U., Pesch, E.: Evolution Based Learning in a Job Shop Scheduling Environment, Research Memorandum 92-019, University of Limburg, Limburg 1992.

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

[9] Dueck, G., Scheuer, T.: Threshold Accepting: A General Purpose Optimization Algorithm Superior to Simulated Annealing, Journal of Computational Physics (1990) 90, pp. 161-175.

[10] Fang, H.-L., Ross, P., Corne, D.: A Promising Genetic Algorithm to Job-Shop Scheduling, Rescheduling, and Open-Shop Scheduling Problems, in: Forrest, S. (ed.): Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo 1993, pp. 375-382.

[11] Fisher, H., Thompson, G.L.: Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules, in: Muth, J.F., Thompson, G.L. (eds.): Industrial Scheduling, Englewood Cliffs 1963, pp. 225-251.

[12] Giffler, B., Thompson, G.L.: Algorithms for Solving Production-Scheduling Problems, Operations Research 8 (1960), pp. 487-503.

[13] Glover, F.: Tabu Search - Part I, ORSA Journal on Computing 1 (1989) 3, pp. 190-206.

[14] Glover, F.: Tabu Search - Part II, ORSA Journal on Computing 2 (1990) 1, pp. 4-33.

[15] Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Opimization by Simulated Annealing, Science 220 (1983) 4598, pp. 671-680.

[16] Lawrence, S.: Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh 1984.

[17] Matsuo, H., Suh, C.J., Sullivan, R.S.: A Controlled Search Simulated Annealing Method for the General Jobshop Scheduling Problem, Working Paper 03-04-88, Departement of Management, Graduate School of Business, The University of Texas at Austin, Austin 1988.

[18] Nowicki, E., Smutnicki, C.: A Fast Taboo Search Algorithm for the Job Shop Problem, Preprint No. 8/93, Technical University of Wrocław, Institute of Engineering Cybernetics, Wrocław 1993.

[19] Siedentopf, J.: Ein effizienter Scheduling-Algorithmus auf Basis des Threshold Accepting, Arbeitsbericht Nr. 4 des Instituts für Produktionswirtschaft und Industrielle Informationswirtschaft der Universität Leipzig, Leipzig 1995.

[20] Siedentopf, J.: An Efficient Unidirectional Search Approach for Job Shop Scheduling, in: Kleinschmidt, P., Bachem, A., Derigs, U., Fischer, D., Leopold-Wildburger, U., Möhring, R. (eds.): Operations Research Proceedings 1995, Berlin, Heidelberg et al. 1996, pp. 167-172.

[21] Taillard, E.D.: Parallel Taboo Search Techniques for the Job Shop Scheduling Problem, ORSA Journal on Computing 6 (1994) 2, pp. 108-1117.

[22] Van Laarhoven, P.J.M., Aarts, E.H.L., Lenstra, J.K.: Job Shop Scheduling by Simulated Annealing, Operations Research 40 (1992) 1, pp. 113-125.

UNIVERSITÄT LEIPZIG

Institut für Produktionswirtschaft und Industrielle Informationswirtschaft

- 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.

Nr. 18: Zelewski, Stephan: Diskussionspapier zum Text "Zur wirtschaftlichen und sozialen Lage in Deutschland" einer evangelisch-katholischen Arbeitsgruppe, Leipzig 1995.

Nr. 19: Schimmel, Katrin; Zelewski, Stephan: Untersuchung alternativer Auktionsformen hinsichtlich ihrer Eignung zur Koordination verteilter Agenten auf Elektronischen Märkten, Leipzig 1996.

Nr. 20: Siedentopf, Jukka: Feinterminierung unter restriktiven Laufzeitanforderungen - Ein exemplarischer Vergleich lokaler Suchverfahren (Teil I), Leipzig 1996.