Short Listing für multikriterielle Job-Shop Scheduling-Probleme

Dr. André Henning , r.z.w.-cimdata AG,
Zum Hospitalgraben 2, 99425 Weimar, andre.henning@rzw.de

1. Multikriterielle Job-Shop Scheduling-Probleme

Das Job-Shop Scheduling-Problem, im Folgenden als JSP bezeichnet, ist formal wie folgt definiert: Das JSP besteht aus einer Menge von Jobs $J = \{J_j\}_{j=1}^{n}$ die auf einer Menge von Maschinen $M = \{M_i\}_{i=1}^{m}$ bearbeitet werden müssen. Jeder Job J_j besteht aus einer Menge von m_j Vorgängen $T = \{T_{j1}, T_{j2}, ..., T_{jm_j}\}$, die auch als Tasks oder Operationen bezeichnet werden. Die zu einem Job gehörenden Tasks müssen in einer vorgegebenen festen Reihenfolge auf den Maschinen bearbeitet werden. Es gibt insgesamt N Vorgänge, $|T| = N = \sum_{j=1}^{n} m_j$. Der Vorgang T_{ji} gehört zu Job J_j und muss auf Maschine M_i für eine ununterbrochene Dauer p_{ji} bearbeitet werden. Jeder Job hat seine eigene, von den anderen Jobs unabhängige Maschinenreihenfolge und durchläuft jede Maschine höchstens einmal. Jede Maschine kann nur einen Vorgang gleichzeitig bearbeiten. Zwei Vorgänge des gleichen Jobs können nicht simultan bearbeitet werden. Ein zulässiger *Maschinenbelegungsplan (engl. Schedule)* ist durch Startzeiten $s_{ji} \geq 0$ für alle Vorgänge T_{ji} definiert, so dass alle obigen Nebenbedingungen erfüllt sind. Gesucht ist ein Schedule, der eine gegebene Zielfunktion minimiert.
Dieses bekannte Modell ist für praktische Anwendungen zu unflexibel. Deshalb werden hier folgende Erweiterungen betrachtet:

1. Verallgemeinerte Reihenfolgebeziehungen
   In der Praxis ist die Ordnung der Vorgänge innerhalb eines Jobs nicht immer linear vorgegeben. Das bedeutet, die Vorgänge in einem Job sind nur teilweise geordnet. Diese Scheduling-Probleme werden als Mixed-Job-Probleme bezeichnet. Vorgänge eines Jobs, zwischen denen keine Vorrangbeziehungen bestehen, werden als parallele Tasks bezeichnet.
   Weitere Reihenfolgebeziehungen ergeben sich aus Montageaufträgen. Komplexe Endprodukte bestehen in der Praxis meist aus mehreren Baugruppen. Jede Baugruppe benötigt zur Produktion Ressourcen und hat eine vorgegebene technologische Reihenfolge. Die Produktion einer Baugruppe kann deshalb als Job im Sinne eines Mixed-Job-Problems betrachtet werden. Die Montagebeziehungen werden durch Reihenfolgebeziehungen zwischen den Jobs modelliert.
2. Verfügbarkeitsintervalle auf Maschinen und Fälligkeitstermine
   Maschinen haben in realen Produktionsumgebungen deterministische Stillstandzeiten (z.B. Schichtpausen). Die Bearbeitung der Vorgänge wird während der Stillstandszeiten gestoppt und zu Beginn des nächsten Verfügbarkeitsintervalls weitergeführt. Das bedeutet, dass die Dauer eines Vorgangs von der Startzeit abhängig ist.
3. Erneuerbare diskrete Ressourcen
   Oft benötigen Vorgänge weitere Ressourcen zur Bearbeitung. Das können Werkzeuge, Paletten oder Arbeitskräfte sein. Diese Ressourcen heißen erneuerbar, da sie nach Beendigung eines Vorgangs wieder zur Verfügung stehen und nicht (wie z.B. Materialien) verbraucht werden. Der Menge $R = \{R_k\}_{k=1}^{r}$ der r verschiedenen Ressourcenarten im Scheduling-Problem wird der Vektor $ra = (ra_1, ..., ra_r), ra_k \in \mathrm{N}$ zugeordnet, der für jede Ressource die verfügbare Menge angibt. Jeder Vorgang kann jede Ressource innerhalb ihrer vorhandenen Menge zur Bearbeitung benötigen.

Ein weiteres Merkmal praktischer Scheduling-Probleme ist das Vorhandensein mehrerer Zielfunktionen. Hier wurden der Makespan $C_{\max}$, die Lateness $L_{\max}$, die totale Tardiness $\sum T_j$ und die

Summe der Bearbeitungszeiten $\sum C_j$ sowohl einzeln als auch in multikriteriellen Scheduling-Problemen betrachtet.
Die so eingeführten verallgemeinerten Job-Shop-Probleme werden im Weiteren als *praktische Job-Shop Scheduling-Probleme* PJSP bezeichnet. Diese beinhalten eine oder mehrere Verallgemeinerungen sowie verschiedene oder mehrere Zielfunktionen. Für die Notation multikriterieller Scheduling-Probleme wird die in [12] vorgeschlagene Schreibweise für das γ-Feld in der α|β|γ-Notation genutzt. Die Suche nach der Menge der nichtdominierten Lösungen bei m Zielfunktionen wird mit $\#(f_1,...,f_m), f_i \in \{C_{\max}, L_{\max}, \sum T_j, \sum C_j\}, m \in \{1,2,3,4\}$ im γ-Feld angegeben.

2. Algorithmen zur Optimierung einer Zielfunktion und zur Approximation der Pareto-Menge

Da die Zielfunktionen sehr verschieden sind und somit auch die Struktur des Problems, wurde eine genetische lokale Suche GLS entwickelt, um die Strukturen schon gefundener Lösungen zu nutzen. Als Lösungsrepräsentation wurde die topologische Ordnung oder aufsteigende Nummerierung der Vorgänge im zugehörigen disjunktiven Graphen verwendet. In der lokalen Suche wurde die Shift-Shift-Nachbarschaft implementiert. Dabei sind alle Lösungen benachbart, bei der die Vertauschung zweier Vorgänge in der topologischen Ordnung wieder zu einer zulässigen Lösung führt. Als Variante der lokalen Suche wurde ein deterministischer Schwellwertalgorithmus verwendet. Hierbei werden alle gemäß Shift-Shift-Nachbarschaft benachbarten Lösungen in zufälliger Reihenfolge durchlaufen. Falls ein Nachbar mit gleichem oder besserem Zielfunktionswert gefunden wird, wird zu diesem übergegangen. Der Übergang zu einem Nachbarn mit gleichem Funktionswert wird als Sidestep bezeichnet. Die Suche bricht ab, falls in der Nachbarschaft keine Lösung mit gleichem oder besserem Funktionswert existiert oder nach einer vorgegebenen Anzahl von Nachbarschaftsschritten ohne Verbesserung.
Als Rekombinationsoperator in der GLS wurde die in [11] eingeführte *Mittelwertbildung* der Anfangszeiten der Vorgänge in den Elterlösungen verwendet.

Definition 1: (*Mittelwertbildung*)
Sei $\Pi = \{\pi^1, \pi^2, ..., \pi^k\}$, $k \geq 2$ die Menge der zulässigen Elternlösungen in ihrer Darstellung als aufsteigende Nummerierung und N die Anzahl der Vorgänge der Lösungen. s_i^j, $i = 1,..,N$, $j = 1,...,k$ sei die Startzeit desVorgangs i in der Lösung j. Setze $\bar{s}_i = \sum_{j=1}^{k} s_i^j$ für jedes $i = 1,...,N$ und sortiere die Vorgänge aufsteigend nach $\bar{s}_i$. Die resultierende Permutation $\bar{\pi}$ der Vorgänge ist die durch Mittelwertbildung erzeugte aufsteigende Nummerierung der Rekombinationslösung.

Die Mittelwertlösung ist zulässig und der korrespondierende gerichtete Graph kreisfrei. Aus der aufsteigenden Nummerierung wurde der semi-aktive Schedule und damit die Startzeiten der Vorgänge ermittelt.
Algorithmus 1 beschreibt die GLS bei Optimierung mit einer Zielfunktion in Pseudocode. Die Startlösungen werden mit einem Prioritätsregelverfahren bei zufälliger Auswahl erzeugt.

Algorithmus 1: *Genetische lokale Suche für eine Zielfunktion*
Sei n die maximale Anzahl der Lösungen im Pool P.
1. Für $i = 1,...,n$
(a) Generiere zufällige Startlösung y
(b) Finde mit Sidestep Algorithmus beginnend mit y lokales Optimum x und füge x in P ein.
2. Wähle zufällig gemäß Gleichverteilung k Lösungen $\{x_1,...x_k\}$ aus P mit $k \leq n$.

3. Bilde Mittelwertlösung x aus den k Lösungen.
4. Finde mit Sidestep Algorithmus beginnend mit x lokales Optimum y und füge y in P ein.
5. Entferne die gemäß Zielfunktion schlechteste Lösung aus P.
6. Falls Abbruchkriterium erfüllt, STOPP. Sonst gehe zu 2.

Die durch den Sidestep Algorithmus erzeugten lokalen Optima werden nur in den Pool eingefügt, falls im Pool keine Lösung mit dem gleichen kritischen Pfad bezüglich $C_{\max}$ existiert, um zu schnelle Konvergenz gegen schlechte lokale Optima zu verhindern. Das Abbruchkriterium der GLS ist eine Zeitschranke oder eine vorgegebene Anzahl von Generationen.
Die GLS wurde auf einer Menge von 242 Standard-Benchmark-Instanzen für $J \| C_{\max}$ und an 60 Instanzen für $J \| L_{\max}$ getestet. [1,6,10] Von diesen gut untersuchten Instanzen wurden bei 78 die oberen Schranken für den Makespan verbessert und bei weiteren 141 Instanzen Lösungen für die bekannten oberen Schranken gefunden. Bei den restlichen 23 Benchmark-Instanzen lag die Abweichung zur oberen Schranke bei unter einem Prozent. Bei den 60 $J \| L_{\max}$-Instanzen wurden alle bekannten oberen Schranken verbessert. Mit einer Zeitbeschränkung von 500 Sekunden auf einem 1GHz Pentium III Rechner erreichte die GLS Lösungen, die im Durchschnitt maximal 5% über der besten bekannten oberen Schranke lagen. [5]
Die verwendete Nachbarschaft und der Rekombinationsoperator waren bei diesen Untersuchungen unabhängig von der zu optimierenden Zielfunktion. Auf dieser Basis wurde die GLS zur Approximation der Menge der Pareto-optimalen Lösungen verallgemeinert. Zum Vergleich der Lösungen in der Nachbarschaft in der lokalen Suche wird eine *Ranking-Funktion* **F** benötigt.

Definition 2: (*Ranking-Funktion* **F**)
Sei $S = \{s_j\}_{j=1,\dots,n}$ eine Menge von Lösungen eines Scheduling-Problems, $F = \{f_i\}_{i=1,\dots,m}$ die Menge der relevanten Zielfunktionen und $W = \{w_i\}_{i=1,\dots,m}$ eine Menge von Gewichten für die Zielfunktionen aus F mit $\sum_{i=1}^{m} w_i = 1$. Sei $\min_{f_i} = \min_{j=1,\dots,n} f_i(s_j)$ das Minimum aller Lösungen aus S und $\max_{f_i} = \max_{j=1,\dots,n} f_i(s_j)$ das Maximum, dann ist die Ranking-Funktion **F** wie folgt definiert:

$$\mathbf{F}(s_j) = \sum_{i=1}^{m} w_i \frac{f_i(s_j) - \min_{f_i}}{\max_{f_i} - \min_{f_i}}$$

für $\max_{f_i} \neq \min_{f_i}$. Bei Gleichheit wird der Quotient gleich 0 gesetzt.

Der Pool P der Startlösungen der multikriteriellen genetischen lokalen Suche (MGLS) wird mit zufälligen Startlösungen oder mit lokalen Optima (bezüglich der einzelnen Zielfunktionen aus $F = \{f_i\}_{i=1,\dots,m}$) gefüllt. Mit Hilfe der Ranking-Funktion und der vom Benutzer vorgegebenen Gewichtung, können die Lösungen im Pool geordnet werden. In der multikriteriellen Version des Sidestep Algorithmus wird ein Nachbarschaftsschritt durchgeführt, falls der Nachbar einen besseren Ranking-Funktionswert hat oder bezüglich des Pools der MGLS den gleichen Rang besitzt. Letzteres wird hier als Sidestep bezeichnet. Der multikriterielle Sidestep Algorithmus führt eine Liste mit allen während des Durchlaufs gefundenen Pareto-optimalen Lösungen mit. Diese werden nach dem Abbruch des Sidestep Algorithmus in den Pool eingefügt. Aus dem Pool werden dann die dominierten Lösungen entfernt, die den schlechtesten Ranking-Funktionswert besitzen. Dies wird fortgesetzt, bis die vorgegebene Poolgröße erreicht ist. Falls nur noch Pareto-optimale Lösungen im Pool existieren, werden ebenfalls die mit dem schlechtesten Ranking-Funktionswert entfernt. Zur Selektion aus dem Pool der MGLS wird die vom Benutzer vorgegebene Gewichtung verwendet. Für jeden Lauf des multikriterielle Sidestep Algorithmus wird eine neue Gewichtung zufällig gleichverteilt erzeugt.

3. Short Listing

Mit der multikriteriellen genetischen lokalen Suche lässt sich eine Approximation der Menge der Pareto-optimalen Lösungen für praktische Scheduling Probleme erzeugen. Die Aufgabe des Entscheidungsträgers besteht darin, aus dieser Menge von Lösungen diejenige auszuwählen, die für die Anwendung übernommen werden soll. Ranking-Funktionen können hierbei als multikriterielle Bewertungsmethoden verwendet werden. Schwierigkeiten bestehen in der subjektiven Gewichtung der Zielfunktionen, Änderungen der Reihefolge im Ranking durch hinzufügen oder herausnehmen einer Lösung aus dem Pool und in der Auswahl der Ranking-Funktion selbst. Die große Anzahl der Lösungen im Pool und die Datenmenge einer Lösung hingegen erschwert die Visualisierung und somit den Vergleich aller Lösungen. Es wird eine Methode zur Entscheidungsunterstützung benötigt, die die Menge der Lösungen im Pool auf eine überschaubare Menge (3 bis 6 Lösungen) reduziert. Diese reduzierte Menge kann dem Entscheidungsträger vorgelegt und visualisiert werden. Wichtig ist hierbei, dass durch die Reduzierung möglichst strukturell verschiedene Lösungen ausgewählt werden, um dem Entscheider echte Alternativen zu bieten.
Aussagen über die strukturelle Verschiedenheit von Lösungen können durch Abstandsmaße im Lösungsraum getroffen werden. Als Methode zur Reduktion der Anzahl der Lösungen bietet sich die Clusterung der Lösungen nach diesen Abstandsmaßen an. Aus den Clustern werden Lösungen ausgewählt, und die so entstandene Liste überschaubar weniger Lösungen wird dem Entscheider zur Auswahl vorgelegt. Der Vorgang der Reduktion vieler Vorschläge auf wenige Kandidaten wird als Short Listing bezeichnet.
Die hier entwickelte GLS ist eine Heuristik zur Reihenfolgeoptimierung der Vorgänge auf den Maschinen. Die Menge der Vorgänge $\{T_{ji}\}$, die auf der gleichen Maschine M_i bearbeitet werden müssen, ist für alle Lösungen einer Instanz gleich. Der Abstand $A(\pi,\sigma)$ zwischen zwei Lösungen π und σ kann aus diesem Grund wie folgt definiert werden:

$$A(\pi,\sigma)=\sum_{i=1}^{m} A_i(\pi_i,\sigma_i).$$

π_i und σ_i sind hierbei Permutationen der Tasks aus $\{T_{ji}\}$, die auf der Maschine M_i bearbeitet werden müssen. Weiterhin gilt durch die Definition von Abstandmaßen das Folgende. Sei f eine Abbildung der Menge $\{T_{ji}\}$ auf die Menge $\{1,...,n\}, n=|\{T_{ji}\}|$, dann gilt $A_i(\pi_i,\sigma_i)=A_i(\alpha,\beta)$, wobei $\alpha=f\circ\pi_i$ und $\beta=f\circ\sigma_i$ Permutationen auf der Menge $\{1,...,n\}$ sind. Wir können uns somit auf die Untersuchung von Abstandsmaßen A, die auf der Menge $\{1,...,n\}$ definiert sind, beschränken. Aus Vereinfachungsgründen wird im Weiteren der Index i weggelassen.
Abstandsmaße auf Permutationen sind aus der Literatur bekannt. Eine Übersicht findet man in [2]. Hier werden folgende Distanzmaße verwendet:

$D(\alpha,\beta)$ $=\sum|\alpha(i)-\beta(i)|$ (Footrule) und

$T(\alpha,\beta)$ =minimale Anzahl von Transpositionen, um α in β zu überführen (Cayley-Abstand).

Beide Maße lassen sich in linearer Zeit in der Länge der Permutation berechnen.
Basierend auf diesen Abstandsmaßen kann die Menge der potentiell Pareto-optimalen Lösungen geclustert werden. Als Heterogenitätmaße zwischen zwei Clustern C_i, C_j wurde der minimale Abstand zweier Lösungen im Cluster v_s (single linkage), der maximale Abstand zweier Lösungen im Cluster v_c (complete linkage) und der durchschnittliche Abstand der Lösungen beider Cluster v_a (average linkage) verglichen.

Zur Klassifikation wurde ein hierarchisches und ein nicht-hierarchischesVerfahren implementiert. Im hierarchischen Verfahren wurden kleinere Cluster zu größeren generalisiert (bottom up). Begonnen wurde mit Clustern mit jeweils einer Lösung. Die Cluster mit der geringsten Distanz

wurden zusammengefasst. Die Anzahl k der Cluster bei der gestoppt wurde, wurde vom Entscheider bestimmt. Das nicht-hierarchische Verfahren ist ein lokale Suche Algorithmus. Zu Beginn werden zufällig gemäß Gleichverteilung k Cluster erzeugt. Danach wurden die Lösungen in zufälliger Reihenfolge durchlaufen. Die jeweils ausgewählte Lösung wurde aus ihrem Cluster entfernt und dem Cluster mit dem geringsten Abstand zugeordnet. Gestoppt wurde, wenn in einem Durchlauf keine Lösung ihren Cluster gewechselt hat.
Zum Vergleich der Klassifikationsverfahren wird noch ein Gütemaß für eine Klassifikation $\mathbf{C}$ benötigt. Sei das Homogenitätsmaß h_a der durchschnittliche Abstand der Lösungen in einem Cluster C_i. Das hier verwendete Gütemaß $g(\mathbf{C})$ wird folgendermaßen definiert:

$$g(\mathbf{C}) = \frac{(|\mathbf{C}|-1)\sum_{C_i \in \mathbf{C}} h_a(C_i)}{\sum_{\substack{C_i, C_j \in \mathbf{C} \\ C_i \neq C_j}} v_a(C_i, C_j)}.$$

Für den Vergleich der Klassifikationsalgorithmen wurde eine Teilmenge von 60 Benchmark-Instanzen mit verschiedenen Dimensionen verwendet. Zu jeder Zielfunktionskombination wurde mit der MGLS für jedes Benchmark ein Pool von 50 Lösungen erzeugt, die dann mit beiden Verfahren geclustert wurden. [5]
In den Versuchen zeigte sich, dass die durch das hierarchische Clusterverfahren ermittelten Klassifikationen bei allen Zielfunktionskombinationen, zwei Abstandsmaßen und drei Heterogenitätsmaßen bessere Werte für das Gütemaß als das nicht-hierarchische Verfahren erreichen. Die Wahl des Abstandsmaßes hat nur einen geringen Einfluss auf die Güte der Klassifikation. Bei den drei Heterogenitätsmaßen sind die Unterschiede in der Güte der Klassifikationen deutlicher. Das Maß v_s erreicht beim hierarchischen Clusterverfahren die beste Güte und beim nicht-hierarchischen die schlechteste. Beim Vergleich der Maße v_c und v_a erreichen die Verfahren mit v_a die etwas besseren Ergebnisse.
Neben der Güte der Klassifikation sind weitere Eigenschaften der Verfahren von Bedeutung. Zum einen die Laufzeit der Algorithmen, da die Auswahl der Lösung aus dem vorhandenen Pool durch den Entscheider interaktiv erfolgt. Hier hat das nicht-hierarchische Verfahren Vorteile, da die Antwortzeiten deutlich geringer sind. Bei einer Anzahl von 600 Vorgängen in der Instanz und einer Poolgröße von 50 Lösungen, ist das nicht-hierarchische Verfahren um etwa den Faktor 4 schneller. Zum anderen ist die Größe der Cluster von Bedeutung, falls der Entscheider die präferierte Lösung mit anderen ähnlich strukturierten Lösungen vergleichen möchte. In dem Fall sind Verfahren, die mehrere Cluster mit einer Lösung und einen Cluster mit den restlichen Lösungen liefern, ungünstig. Aus diesem Grund wurden die Clustergrößen der durch die 12 Varianten erstellten Klassifikationen ermittelt. Dabei war ersichtlich, dass das hierarchische Clusterverfahren zur Bildung kleiner Cluster neigt.
Ziel der Clusterung der Lösungsmenge ist die Reduktion der zur Auswahl stehenden Lösungen. Dazu wurde aus jedem Cluster eine Lösung ausgewählt. Es wurden die folgenden zwei Methoden der Auswahl untersucht:

1. In jedem Cluster wurden die Lösungen nach der Ranking-Funktion sortiert und jeweils die Lösung auf dem ersten Rang selektiert. Dabei hatten alle Zielfunktionen die gleiche Gewichtung.
2. Aus jedem Cluster wird die Medianlösung selektiert, d.h. die Lösung die den minimalen durchschnittlichen Abstand zu den anderen Lösungen im Cluster hat.

Zur Untersuchung dieser beiden Varianten wurden die oben ermittelten Klassifikationen verwendet. Damit wurde aus jeder Klassifikation drei Lösungen ausgewählt. Es wurde der durchschnittliche Abstand zwischen den Lösungen und der minimale Abstand zwischen je zwei der drei Lösungen ermittelt.
Die in den Versuchen ermittelten Daten lassen folgende Beobachtungen zu:

1. Die durch das hierarchische Clusterverfahren erzeugten Klassifikationen liefern größere Abstände zwischen den ausgewählten Lösungen als die durch das nicht-hierarchische Verfahren erzeugten Klassifikationen. Dies gilt für beide Auswahlverfahren.
2. Die Abstände zwischen den Lösungen sind bei Auswahl der jeweils ersten Lösung im Cluster größer als bei der Auswahl der Medianlösung. Dies gilt wiederum für beide Klassifikationsalgorithmen.
3. Die Wahl des Abstandsmaßes hat nur einen geringen Einfluss auf die relativen Abstände zwischen den ausgewählten Lösungen.
4. Das Heterogenitätsmaß v_s liefert unter Verwendung des nicht-hierarchischen Verfahrens die schlechtesten Ergebnisse.

Nach diesen Resultaten ist es bei beiden Clusteralgorithmen am günstigsten, die Lösungen auf dem ersten Rang aus jedem Cluster zu wählen. Dabei wird im Vergleich mit der zweiten Auswahlmethode die größte Diversität zwischen den ausgewählten Lösungen erreicht. Weiterhin werden dadurch die Präferenzen des Entscheider berücksichtigt.

Es bleibt festzuhalten, dass die Vorteile des hierarchischen Verfahrens in der besseren Güte der Klassifikationen und in der größeren Diversität der ausgewählten Lösungen liegen. Die Vorteile des nicht-hierarchischen Algorithmus liegen in der kürzeren Laufzeit, der ungefähr gleichen Größe der Cluster und der besseren Berücksichtigung der Entscheiderpräferenzen. Bei beiden Algorithmen erreichten die Heterogenitätsmaße v_c und v_a sowie die Auswahlmethode 1 die besten Ergebnisse.

Literatur

[1] Benchmark-Instances of Demirkol et al.
http://gilbreth.ecn.purdue.edu/_uzsoy2/benchmark/problems.html

[2] Diaconis, P.: *Group Representations in Probability und Statistics.* Lecture Notes - Monograph Series, Vol. 11, Institute of Mathematical Statistics, Harvard University (1988).

[3] Esquivel S.; Ferrero S.; Gallard R.; Salto C.; Alfonso H.; Schütz M.: *Enhanced evolutionary algorithms for single and multiobjective optimization in the job shop scheduling problem.* Knowledge-Based Systems 15, 13-25 (2002).

[4] Hansen, M. P.; Jaszkiewicz, A.: *Evaluating the quality of approximations to the non-dominated set.* Technical report 07/98, Institute of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark (1998).

[5] Henning, A.: *Praktische Job-Shop Scheduling-Probleme.* Dissertation, Fakultät für Mathematik und Informatik, Friedrich-Schiller-Universität Jena, Jena (2002).

[6] Taillard, É: Homepage :
http://www.eivd.ch/ina/Collaborateurs/etd/problemes.dir/ordonnancement.dir/ordonnancement.html

[7] Jain, A. S.: *A multi-level hybrid framework for the deterministic job-shop scheduling problem.* Ph. D. Thesis, Department of Applied Physics and Electronic and Mechanical Engineering, University of Dundee (1998).

[8] Neumann, K.; Schwindt, C.; Zimmermann, J.: *Project scheduling with time windows and scarce resources.* Lecture Notes in Economics and Mathematical Systems, Vol. 508, Springer, Berlin Heidelberg New York (2001).

[9] Nowicki, E.; Smutnicki, C.: *New ideas in TS for job shop scheduling.* Preprint nr 50/2001, Institute of Engineering Cybernetics, Technical University of Wroclaw, Wroclaw, Poland (2001).

[10] OR-Library: http://www.ms.ic.ac.uk/jeb/pub/jobshop1.txt

[11] Rose, C.*: Mehrheitsbildung in der Kombinatorischen Optimierung.* Dissertation, Fakultät für Mathematik und Informatik, Friedrich-Schiller-Universität Jena, Jena (2001).

[12] T'kindt, V; Billaut, J.-C.: *Multicriteria scheduling problems: a survey.* RAIRO Oper. Res. 35, 143-163 (2001).