Steuerung einer Heuristik zur Losgrößenplanung unter Kapazitätsrestriktionen mit Hilfe eines parallelen genetischen Algorithmus.

J. Branke, U. Kohlmorgen, H. Schmeck, H. Veith
Institut für Angewandte Informatik und Formale Beschreibungsverfahren
Universität Karlsruhe, 76128 Karlsruhe
E-mail: {branke, kohlmorgen, schmeck, veith}@aifb.uni-karlsruhe.de

Abstract

In dieser Arbeit wird das Problem der kostenminimalen Losgrößenbestimmung betrachtet, wenn der Rüstzustand der Maschine über eine Periodengrenze hinweg erhalten werden kann.

Ausgangspunkt ist eine Heuristik, die abhängig von einem Steuerungsparameter die Losgrößen, angefangen von der letzten Planungsperiode bis zurück zur ersten Periode, bestimmt. Der Parameter gibt die Gewichtung von Rüst- gegenüber Lagerkosten vor und beeinflußt so die Entscheidungen in den jeweiligen Perioden.

In dieser Arbeit wird die Heuristik derart erweitert, daß der Steuerungsparameter für verschiedene Perioden verschiedene Werte annehmen darf. Statt eines einzigen Parameters benutzen wir also ein Tupel von Parametern, entsprechend der Anzahl der Planungsperioden. Die Optimierung dieses Tupels erfolgt durch einen genetischen Algorithmus.

Der genetische Algorithmus ist auf einem massiv parallelen SIMD-Rechner implementiert und nutzt die lokale Verbindungsstruktur der Prozessoren untereinander.

Grundlagen und Modellbeschreibung

Beim kapazitierten Losgrößenplanungsproblem geht es darum, die Losgrößen für J Produkte über einen Zeithorizont von T Perioden auf einer Maschine so zu wählen, daß die Summe aus Rüst- und Lagerkosten minimiert wird. Dabei ist der Planungshorizont in T Zeitperioden unterteilt, die Kapazität der Maschine in jeder Periode beschränkt, und der Bedarf in jeder Periode muß vollständig gedeckt werden, es sind also keine Rückstände erlaubt.

In der Literatur wird außerdem meist davon ausgegangen, daß für jedes Gut, das in einer Periode produziert wird, Rüstkosten anfallen.

Dann läßt sich das Problem formal folgendermaßen beschreiben:

$$\min \sum_{j=1}^{J} \sum_{t=1}^{T} (s_j x_{jt} + h_j I_{jt}) \qquad (1)$$

$u.d.N.$

$$I_{j,t-1} + q_{jt} - I_{jt} = d_{jt} \qquad (j = 1,...,J; t = 1,...,T) \qquad (2)$$

$$\sum_{j=1}^{J} p_j q_{jt} \leq C_t \qquad (t = 1,...,T) \qquad (3)$$

$$C_t x_{jt} - p_j q_{jt} \geq 0 \qquad (j = 1,...,J; t = 1,...,T) \qquad (4)$$

$$I_{jt}, q_{jt} \geq 0 \qquad (j = 1,...,J; t = 1,...,T) \qquad (5)$$

$$x_{jt} \in \{0,1\} \qquad (j = 1,...,J; t = 1,...,T) \qquad (6)$$

wobei

J: Anzahl der verschiedenen Produkte, die gefertigt werden.

T: Anzahl an Planungsperioden

q_{jt}: Produzierte Menge von Produkt j in Periode t.

d_{jt} Nachfrage nach Produkt j in Periode t.

C_t: Kapazität der Maschine in Periode t.

s_j: Rüstkosten für Produkt j.

h_j: Lagerhaltungskosten für eine Einheit von Proukt j in einer Periode.

I_{jt}: Lagerbestand von Produkt j am Ende von Periode t

x_{jt}: eine binäre Variable die angibt, ob Produkt j in Periode t gefertigt wird (x=1) oder nicht (x=0)

p_j: Kapazitätsbedarf zur Herstellung einer Einheit des Produkts j.

o.B.d.A. gilt: $I_{j0} = 0$ für alle j = 1,...,J.

Gleichungen (2) und (5) stellen sicher, daß der gesamte Bedarf auch tatsächlich befriedigt wird. Nebenbedingung (3) stellt sicher, daß die gesamte Produktion in Periode t nicht die Kapazität überschreitet. Gleichung (4) erzwingt, daß für jede Periode t, in der j produziert wird ($q_{jt} > 0$), auch die Rüstkosten angesetzt werden ($x_{jt} = 1$).

Dieses Problem ist NP-schwer, wie Florian et al [FLK80] gezeigt haben. Zur Lösung gibt es eine ganze Reihe von Heuristiken, vgl. zum Beispiel [DS81], [LV79], [CMW90] oder [KK94].

Die Annahme, daß Rüstkosten in jeder Periode für jedes produzierte Gut anfallen, ist in der Praxis nicht immer gegeben. Häufig kann nämlich der Rüstzustand einer Maschine über die Periodengrenze hinweg beibehalten werden. Dann läßt sich durch geschickte Ablaufplanung in den einzelnen Perioden mitunter ein erheblicher Teil der Rüstkosten einsparen.

Ein einfaches Beispiel (aus [Ha94a]) soll dies verdeutlichen:

Seien J=2, T=3, Rüstkosten $s_1=s_2=100$, die Lagerhaltungskosten $h_1=h_2=1$ und die Kapazität der Maschine in allen Perioden gleich 10. Der Bedarf für die Produkte sei aus Tabelle 1 zu entnehmen:

Tabelle1

t =	1	2	3
d_{1t}	5		6
d_{2t}		3	

Tabelle2

t =	1	2	3
q_{1t}	5∪	6	
q_{2t}		3	

Die optimale Lösung für das Problem ohne periodenübergreifende Lose ergibt sich mit einem Zielfunktionswert von 300 mit einem Produktionsplan entsprechend den jeweiligen Bedarfen, da für Produkt 1 wegen der Kapazitätsrestriktion immer zweimal Rüstkosten anfallen. Erlaubt man das Fortbestehen eines Rüstzustandes über eine Periodengrenze hinweg, so kann man, wie in Tabelle 2 dargestellt, Rüstkosten sparen, indem man den Bedarf von Produkt 1 aus Periode 3 schon früher, nämlich in Periode 2 produziert. "∪" bedeutet dabei das Verbinden eines Loses von einer zur nächsten Periode. Bei dieser Strategie ergibt sich ein Zielfunktionswert von 206.

Erlaubt man periodenübergreifende Lose, stellt sich das Problem formal wie folgt dar:

$$\min \sum_{j=1}^{J}\sum_{t=1}^{T}(s_j(x_{jt}-z_{jt})+h_j I_{jt}) \qquad (7)$$

$u.d.N.$

$$I_{j,t-1}+q_{jt}-I_{jt}=d_{jt} \qquad (j=1,...,J;t=1,...,T) \qquad (8)$$

$$\sum_{j=1}^{J} p_j q_{jt} \le C_t \qquad (t=1,...,T) \qquad (9)$$

$$C_t x_{jt}-p_j q_{jt} \ge 0 \qquad (j=1,...,J;t=1,...,T) \qquad (10)$$

$$\sum_{j=1}^{J} z_{jt} \le 1 \qquad (t=1,...,T) \qquad (11)$$

$$z_{jt}-x_{j,t-1} \le 0 \qquad (j=1,...,J;t=1,...,T) \qquad (12)$$

$$z_{jt}-x_{j,t} \le 0 \qquad (j=1,...,J;t=1,...,T) \qquad (13)$$

$$z_{j,t-1}+z_{jt} \le 1 \qquad (j=1,...,J;t=1,...,T) \qquad (14)$$

$$I_{jt}, q_{jt} \ge 0 \qquad (j=1,...,J;t=1,...,T) \qquad (15)$$

$$x_{jt}, z_{jt} \in \{0,1\} \qquad (j=1,...,J;t=1,...,T) \qquad (16)$$

mit der Bedeutung der Variablen wie oben und

z_{jt}: einer binären Variablen, die angibt, ob die Lose von Produkt j in Perioden t und t-1 verbunden werden ($z_{jt} = 1$) oder nicht ($z_{jt} = 0$).

In die Zielfunktion (7) gehen hier nur noch die Rüstkosten ein, wenn das Los nicht mit dem Los der Vorperiode verbunden ist ($z_{jt} = 0$). Gleichung (11) stellt sicher, daß in jeder Periode nur ein Los mit dem Los der nächsten Periode verbunden werden kann. Gleichungen (12) und (13) sind dafür verantwortlich, daß nur Lose verbunden werden können, die tatsächlich produziert werden. (14) vermeidet, daß dasselbe Los sowohl mit der Vorperiode als auch mit der Nachfolgeperiode verbunden wird, und damit evtl gleichzeitig am Anfang (für Verbindung mit Vorperiode) und am Ende (Verbindung mit Nachfolgeperiode) einer Periode eingeplant wird

Es sollte bemerkt werden, daß durch Gleichung (14) der Ausnahmefall, für den in einer Periode nur ein Produkt produziert wird und dieses Los mit den Losen der Vor- und der Nachperiode verbunden werden könnte, in dieser Notation (und in unserem Algorithmus) nicht darstellbar ist.

Falls $z_{jt} = 0$ für alle Perioden und alle Produkte entspricht diese Darstellung gerade dem oben schon beschriebenen Losgrößenproblem ohne periodenübergreifende Lose.

Natürlich ist es möglich, einen für das Problem ohne periodenübergreifenden Lose gefundenen Produktionsplan zu modifizieren, indem man versucht, durch geschicktes Verbinden von Losen oder das Verschieben eines Teilloses auf die nächste Periode Rüst- und Lagerkosten zu sparen (vgl. dazu Beispiel 2).

Wie sich jedoch herausgestellt hat, ist es sinnvoller, die Möglichkeit des Verbindens von Losen von vornherein bei der Losgrößenbestimmung zu berücksichtigen.

Beispiel 2 (nach [Ha94a]):

Sei J=4, T=4, C_t=100 für t=1 bis 4 und die anderen Daten gegeben wie in Tabelle 3:

Tabelle 3: Problemdaten

t =	1	2	3	4	p_j	h_j	s_j
d_{1t}	20	10	30	20	1	1	200
d_{2t}	30	10	30	30	1	1	150
d_{3t}	0	30	10	60	1	1	100
d_{4t}	20	20	0	10	1	1	150

Löst man dieses Problem optimal ohne das periodenübergreifende Verbinden von Losen zu erlauben (für kleine Probleme ist das z.B. mit LINDO [Schra86] möglich), so erhält man den in Tabelle 4 dargestellten Produktionsplan A. Durch Verbinden von Losgrößen und Verschieben der Losgrößen nach hinten erhält man schließlich den modifizierten Produktionsplan B.

Tabelle 4: Lösung des kapazitierten Losgrößenproblems ohne Verbinden benachbarter Lose (A) und mit Verbinden benachbarter Lose und Verschieben nach hinten (B).

	t =	1	2	3	4	Kosten
	q_{1t}	40	0	40	0	
A	q_{2t}	40	0	60	0	1320
	q_{3t}	0	40	0	60	
	q_{4t}	20	30	0	0	
	q_{1t}	40	0	40	0	
B	q_{2t}	40	0	30∪	30	1130
	q_{3t}	0	30∪	10	60	
	q_{4t}	20∪	30	0	0	

Das kapazitätsbeschränkte Losgrößenproblem mit Verbindung von Losen benachbarter Perioden ist bisher in der Literatur relativ wenig untersucht worden. Wir greifen hier auf eine Heuristik von Haase

[Ha94a] zurück und versuchen, diese in Verbindung mit einem genetischen Algorithmus zu verbessern.

Die Heuristik von Haase [Ha94a] beginnt mit dem Einplanen der Produkte in der letzten Planungsperiode. Es wird sichergestellt, daß, falls es einen möglichen Produktionsplan gibt, alle Produkte, die produziert werden müssen, auch noch produziert werden können.

Die Entscheidung, welches Produkt als nächstes eingeplant werden soll, wird von der Abschätzung der durch die Einplanung erhofften Kostenersparnisse gesteuert. Kurz gesagt erhofft man sich durch Einplanen eines Loses eine Verringerung der Lagerkosten und nimmt dafür zusätzliche Rüstkosten in Kauf. Verbindet man zwei Lose über eine Periodengrenze hinweg, so können dadurch Rüstkosten gespart werden. Die gesamte Kostenersparnis setzt sich also aus einer konvexen Kombination von Lagerhaltungs- und Rüstkosten zusammen, wobei die Bedeutung von Rüst- zu Lagerkosten durch einen Gewichtungsfaktor $\gamma \in [0; 1]$ gesteuert wird. Ein gutes γ wiederum wird durch eine einfache Suchheuristik bestimmt.

Für eine genauere Beschreibung der Losgrößen-Heuristik sei der Leser auf [Ha94a] verwiesen.

In dem vorliegenden Artikel wird die eben vorgestellte Heuristik verfeinert insofern als der Gewichtungsfaktor γ von Periode zu Periode variieren darf. Statt eines einzelnen Steuerungsparameters verwenden wir hier also ein Tupel $(\gamma_1, ..., \gamma_T)$ von Parametern, entsprechend der Anzahl der Planungsperioden. Eine wie in [Ha94a] einfache Suchheuristik zur Bestimmung der γ_t ist dann jedoch nicht mehr möglich, da sich Veränderungen des γ_t-Wertes in einer Periode auf den optimalen γ_k-Wert in anderen Perioden auswirken können.

Deshalb schlagen wir hier einen genetischen Algorithmus zur Suche nach einem möglichst guten γ-Tupel vor.

Der genetische Algorithmus

Genetische Algorithmen sind globale stochastische Optimierungsverfahren, die auf den Prinzipien der natürlichen Evolution beruhen. Als einführende Literatur sei hier beispielsweise auf [Gol89] oder [Mic94] verwiesen. Generell operiert ein genetischer Algorithmus auf einer Menge von Individuen, auch Population genannt. Jedes Individuum in der Population repräsentiert eine potentielle Lösung des Problems, in unserem Fall gibt ein Individuum also einen Losgrößenplan an. Die Güte eines Individuums wird durch die Kosten, die der entsprechende Plan verursacht, bestimmt. In einem sich mehrfach wiederholenden gesteuerten stochastischen Prozeß werden durch Auswahl guter Individuen

und durch Anwendung genetischer Operatoren (wie Crossover und Mutation) auf die Individuen neue Individuen erzeugt.

Der hier verwendete parallele genetische Algorithmus hat als Grundlage das Nachbarschafts-Modell (siehe hierzu u.a. [Mü91]. Dabei sucht sich jedes Individuum der Population aus seiner Nachbarschaft einen Partner für die Erzeugung von Nachkommen. Die Implementierung des Algorithmus erfolgte auf einem MasPar2 Rechner. Die Prozessoren sind in einem 2-dimensionalen gitterverbundenen Netz der Größe 128 mal 128 angeordnet. Jeder Prozessor hat Verbindungen zu seinen 8 Nachbarprozessoren (siehe Abbildung 1). Damit keine Randeffekte auftreten, haben auch die Randprozessoren dieselbe Verbindungsstruktur, d.h. die Prozessoren am linken Rand haben die Prozessoren des rechten Rands als ihre Nachbarn usw.

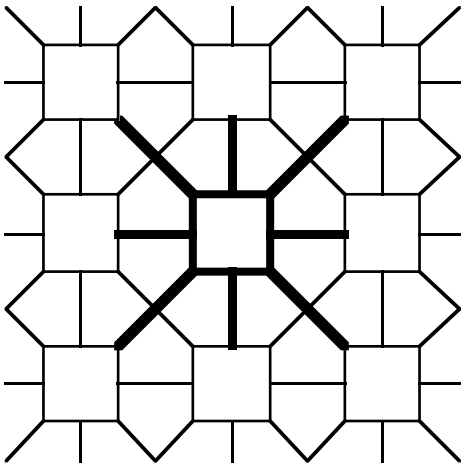

Abbildung 1: Verbindungsstruktur der Prozessoren auf dem Parallelrechner.

Die genetische Repräsentation (das Chromosom) eines Individuums ist das γ-Tupel der Länge T (Anzahl der Perioden). Das k-te Gen des Chromosoms enthält dabei den Wert von γ_k als real-Zahl.

Die Populationsgröße beträgt 16.384 Individuen. Auf jedem Prozessor liegt ein Individuum der Population. Für den Crossoverprozeß sucht sich jedes Individuum (Elter1) das beste Individuum aus seinen 8 Nachbarprozessoren und holt eine Kopie (Elter2) auf seinen Prozessor. Aus diesen beiden Eltern entstehen beim Crossover zwei neue Nachkommen (Kind1 und Kind2). Als Crossoverstrategie haben wir den two-point-crossover gewählt. Einer der beiden Nachkommen (Kind1) wird anschließend mutiert. Jedes Gen dieses Nachkommens wird mit einer Wahrscheinlichkeit von 5% verändert. Um den Wert des Gens nicht zu stark zu verändern, wird ein gleichverteilter Wert aus dem Intervall [-0.2, +0.2] auf den vorhandenen Wert addiert. Überschreitet der so erzeugter Wert die Grenzen des Intervalls [0, 1] , dann wird er je nach Bedarf um 0.4 erhöht oder verringert.

Danach beginnt der Selektionsprozeß. Dabei wird aus den beiden Nachkommen (Kind1, Kind2) und aus dem Individuum, das bisher auf diesem Prozessor lag (Elter1), das beste Individuum für die nächste Generation ausgewählt. Die anderen drei Individuen werden für den weiteren Fortgang des genetischen Algorithmus nicht mehr berücksichtigt. Der genetische Algorithmus bricht ab, wenn die Individuen (Elter1) auf allen Prozessoren denselben Zielfunktionswert haben, jedoch spätestens nach 400 Generationen.

Somit ergibt sich folgender Ablauf für den parallelen genetischen Algorithmus:

Zufälliges Erzeugung einer Anfangspopulation
DO
  Jedes Individuum (Elter1) wählt als Partner (Elter2) das beste Inividuum aus der Nachbarschaft
  Crossover: zwei Nachkommen (Kind1, Kind2) werden aus den beiden Eltern erzeugt
  Mutation auf einem Nachkommen (Kind1)
  Bewertung der beiden Nachkommen (Kind1, Kind2)
  Selektion: das beste Individuum aus Elter1, Kind1 und Kind2 wird das neue Elter1
UNTIL Endbedingung erfüllt

Empirische Ergebnisse

Wir vergleichen unseren Algorithmus (GA) zum einen mit der wie in Kapitel 1 beschriebenen modifizierten Dixon-Silver-Heuristik (mDS) und zum anderen mit der unserem Algorithmus zugrundeliegenden Heuristik von Haase (Haase).

Als Testbasis dienen die 120 Probleme aus [CMW90]. Es handelt sich dabei um 3 große Gruppen zufällig erzeugter Probleme mit der Anzahl der Produkte und Anzahl der Planungsperioden (J, T) von (50, 8), (20, 20) und (8, 50).

Jede Gruppe ist wiederum bezüglich dreier Merkmale unterteilt:

- die Kapazitätsauslastung ist entweder niedrig (LU, 75-85%) oder hoch (HU, 94-96%)
- der Kapazitätsverbrauch bei der Produktion einer Einheit eines Gutes ist konstant (CC) oder variabel (VC, gleichverteilt über das Intervall [1,5])
- die Standardabweichungen der Bedarfe ist niedrig (LS, 2-20%) oder hoch (HS, 40-80%).

Bisher haben wir davon lediglich die Gruppe der (8, 50)-Probleme getestet, und davon auch nur die Instanzen mit gleichem Kapazitätsverbrauch für alle Produkte. Da es sich bei unserem Verfahren um

ein stochastisches Verfahren handelt, wurde jeder Testlauf fünf Mal mit verschiedenen Startwerten für den Zufallsgenerator durchgeführt. Die Standardabweichung für verschiedene Startwerte ist relativ gering (vgl. Tabelle 5), so daß der Algorithmus durchaus als robust bezeichnet werden kann.

Tabelle 5: Ergebnisse für ein spezielles Problem (t82, LU-CC-LS)

mDS	Haase	GA
159454	140607	135257,8 ± 67,08

Tabelle 6 erlaubt einen Vergleich unserer Ergebnisse mit denen von Haase und der modifizierten Dixon-Silver-Heuristik. Angegeben ist jeweils die durchschnittliche prozentuale Abweichung über 5 Probleminstanzen derselben Untergruppe im Vergleich zu den Ergebnissen der modifizierten Dixon-Silver-Heuristik (also $100 \cdot \frac{Z_{mDS} - Z_{Haase}}{Z_{Haase}}$ bzw. $100 \cdot \frac{Z_{mDS} - Z_{GA}}{Z_{GA}}$).

Tabelle 6: Vergleich der Ergebnisse der modifizierten Dixon-Silver-Heuristik (mDS), der Heuristik von Haase (Haase) und des hier vorgestellten genetischen Algorithmus (GA).

(8, 50)	Haase	GA
LU-CC-LS	9,07	16,84
LU-CC-HS	4,40	12,93
HU-CC-LS	12,52	18,76
HU-CC-HS	10,92	19,94
Gesamtdurchschnitt	9,23	17,12

Insgesamt zeichnet sich ein sehr positives Bild, da unser Algorithmus um durchschnittlich über 17% besser abschneidet als die modifizierte Dixon-Silver-Heuristik und auch deutlich bessere Ergebnisse liefert als die Heuristik von Haase.

Durch die Verwendung des massiv parallelen Rechners MasPar2 konnte der Zeitaufwand in moderatem Rahmen gehalten werden. Für die Berechnung eines durchschnittlichen Losgrößenplanes wurden ca. 10 Minuten benötigt.

Zusammenfassung und Ausblick

Wir haben einen neuen Algorithmus für das kapazitierte Losgrößenproblem mit der Möglichkeit von periodenübergreifenden Losen vorgestellt. Der Algorithmus ist im wesentlichen eine Verfeinerung der Heuristik von Haase.

Erste Versuche zeigen eine deutliche Verbesserung der Ergebnisse im Vergleich zu einer für dieses Problem modifizierten Dixon-Silver-Heuristik. Das bestätigt unsere Vermutung, daß die Möglichkeit des Verbindends von Losen über Periodengenzen hingweg von vornherein bei der Losgrößenplanung berücksichtigt werden sollte. Eine nachträgliche Modifizierung der Lösung bringt nicht den gewünschten Erfolg.

Auch im Vergleich zu den Ergebnissen von Haase konnten durch unseren genetischen Algorithmus deutliche Verbesserungen erzielt werden.

Zukünftige Forschung mit diesem Algorithmus könnte beinhalten

- einen umfassenderen Vergleich mit den Heuristiken für sämtliche 120 Testprobleme aus [CMW90]
- Vergleiche mit aktuelleren und besseren Methoden für das kapazitierte Losgrößenproblem, wobei diese allerdings wieder wie die hier benutzte Dixon-Silver-Heuristik durch Verbinden und nach hinten Schieben von Losen modifiziert werden müßten. Anbieten würde sich hier etw der Algorithmus von Kirca und Kökten [KK94].
- Verbesserung des Genetischen Algorithmus z.B. durch ausgefeiltere Mutations- und Selektionsoperatoren.

Danksagung

Wir danken Herrn Ömer Kirca für Bereitstellung seiner Daten und ganz besonders Herrn Knut Haase für inspirierende Diskussionen und ebenfalls die Bereitstellung seiner Daten.

Literaturhinweise:

[CMW90] Cattrysse, D., Maes, J., and Van Wassenhove, L.N.: "Set partitioning and column generation heuristics for capacitated lotsizing", European Journal of Operations Research 46 (1990) 38-47

[DS81] Dixon, P. S., and Silver, E. A.: "A heuristic solution procedure for multi-item single-level, limited capacity, lot-sizing problem". Journal of Operations Management 2/1(1981), pp. 23-39.

[FLK80] Florian, M., Lenstra, J.K., and Rinnooy Kan, A.H.G.: "Deterministic production planning: Algorithms and complexity". Management Science 26/7 (1980) 669-679

[Gol89] Goldberg, D.E.: "Genetic Algorithms in Search, Optimization and Machine Learning". Addison-Wesley, 1989

[Ha94a] Haase, K.: "Capacitated Lot-Sizing with Linked Production Qualities of Adjacent Periods". Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universitaet Kiel, No. 334, 1994

[Ha94b] Haase, K.:"Lot-Sizing and Scheduling for Production Planing". Lecture Notes in Economics and Mathematical Systems Nr. 408, Springer Berlin, 1994

[KK94] Kirca, Ö., and Kökten, M.: "A new heuristic approach for the multi-item dynamic lot sizing problem". European Journal of Operational Research 75 (1994) 332-341

[LV79] Lambrecht, M.R., and Vanderveken, H.: "Heuristic procedure for the single operation, multi item loading problem", AIIE Transactions 11/4 (1979) 319-326

[Mic94] Michalewicz, Z.: "Genetic Algorithms + Data Structures = Evolution Programs". Springer Berlin, 1992

[Mü91] Mühlenbein, H.: "Evolution in time and space - the parallel genetic algorithm. In G. Rawlins, editor, Foundations of Genetic Algorithms, Morgan-Kaufman (1991), pages 316-337

[Schra86] Schrage, L.: "Linear, integer and quadratic programming with LINDO". 3rd Edition, Redwood City, 1986