universität wien

DIPLOMARBEIT

Titel der Diplomarbeit

„Ressourcenoptimierung von Netzplänen mit genetischen Algorithmen“

Verfasser

Markus Lopin

angestrebter akademischer Grad

Magister der Sozial- und Wirtschaftswissenschaften (Mag. rer. soc. oec.)

Wien, 2012

Studienkennzahl lt. Studienblatt:	A 157
Studienrichtung lt. Studienblatt:	Diplomstudium Internationale Betriebswirtschaft
Betreuer:	o. Univ.-Prof. Dipl.-Ing. Dr. Richard F. Hartl

Für meine liebe Ehefrau

und

meine wundervollen Kinder

Vorwort

Nach einer biomedizin-technischen Ausbildung am TGM Wien hat sich nach der Matura die Frage für mich gestellt, welche Richtung meine akademische Ausbildung nehmen soll. Nach reiflicher Überlegung habe ich mich für die „Internationale Betriebswirtschaft“ entschieden, welche damals noch als Studienversuch geführt wurde.

Meine erste Vorlesung habe ich bei Herrn em. o. Univ.-Prof. Loitlsberger († 2003), dem Begründer dieser Studienrichtung, besucht, der uns mit seiner Begrüßung, als Kommilitoninnen und Kommilitonen in den Kreis der Lehre als gleichgestellte Forscher aufgenommen hat. In weiterer Folge hat sich für mich die „Produktion und Logistik“ als das interessanteste Wissensgebiet herausgestellt. Durch die sehr gehaltvollen, interessanten und abwechslungsreichen Vorlesungen, Seminare, Übungen und Praktika des Produktionsmanagement von Herrn o. Univ.-Prof. Hartl und A.o. Univ.-Prof. Gronalt konnte ich dieses Wissen sehr vertiefen und auch als Grundlage für meine berufliche Karriere nutzen, die bereits während meiner universitären Ausbildung begonnen hatte. Einen besonderen Dank möchte ich hier an Herrn o. Univ.Prof. Hartl aussprechen, der mich bei der Erstellung dieser Diplomarbeit mit seinem vielfältigen Fachwissen, seiner umfangreichen Literaturkenntnis und vielen anspruchsvollen Diskussionen immer in die richtige Richtung gelenkt hat.

Des Weiteren möchte ich mich bei meinem langjährigen Vorgesetzen Claus Schuster bedanken, der mir immer den nötigen Freiraum gegeben hat, um dieses Projekt zu Ende führen zu können und bei meinem Mentor auf den ersten Schritten der Softwareentwicklung, Reno Fiedermutz.

Sehr wichtig waren für mich auch meine Eltern und Schwiegereltern, die mir immer die notwendige Sicherheit gegeben haben, um in dieser Zeit, frei von Sorgen abseits des Studiums zu sein.

Der größte Dank von allen gebührt meiner lieben Ehefrau Melanie Niope, die mich zu jeder Zeit bedingungslos unterstützt hat und immer zur richtigen Zeit mit den passenden Worten zur Stelle war. Sie hat mir auch zwei liebe Kinder geschenkt, die für

mich die größte Motivation für meinen Studienabschluss dargestellt haben. Daher möchte ich ihnen auch diese Arbeit widmen.

Zuletzt sei noch allen Freunden, Lehrenden und StudienkollegInnen gedankt, die mich während meiner Studienzeit begleitet und vor allem während der Diplomarbeit unterstützt haben.

Merci beaucoup !

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

Formelverzeichnis

Einleitung

Projektmanagement ist ein wichtiger und zentraler Bestandteil der Unternehmensplanung. Die Netzplantechnik als ein Teil davon, bietet die Möglichkeit, die Vorgänge des Projekts zu visualisieren und zu optimieren.

Exakte Verfahren der Projektplanung sind sehr rechen- und zeitintensiv und daher mit einem planungstechnisch vertretbaren Aufwand nur bei Netzplänen von geringem Umfang anwendbar. Aus einer Reihe von einfachen Heuristiken, dessen Zielfunktionswerte der Näherungslösungen noch einigen Abstand zu den exakten Verfahren aufweisen, haben sich, durch die Verwendung von naturwissenschaftlichen Methoden anderer Domänen, eine Vielzahl an metaheuristischen Verfahren entwickelt.

Der **Forschungsschwerpunkt** dieser Arbeit liegt auf:

- Der **Implementierung** einer Applikation, die den hybriden genetischen Algorithmus von Valls et al. (2008) nachbildet.
- Einer **Optimierung** der Applikation, durch den Einsatz von bereits bekannten Crossover-Operatoren anderer Lösungen und einer Variation der bekannten Standard-Parametrisierung.
- Eine **Datenanalyse** und Beschreibung der Ergebnisse der Applikation.

Der erste Teil der Diplomarbeit beschäftigt sich mit dem Thema Netzplantechnik in der Projektplanung. Nach einigen allgemeinen Begriffsdefinitionen und einem Überblick der Darstellungsformen für die Projektplanung wird das kapazitätsbeschränkte Projektplan Problem (RCPSP – Resource-Constrained Project Scheduling Problem) allgemein beschrieben und eine Klassifizierung von Projektplänen vorgenommen.

Es wird gezeigt, wie mit exakten Verfahren, Prioritätsheuristiken und Metaheuristiken ein RCPSP gelöst werden kann. Der Schwerpunkt liegt bei den metaheuristischen Verfahren, insbesondere auf den genetischen Algorithmen. Hier werden nach einer allgemeinen Einführung in die Thematik und dessen Bezug zur Naturwissenschaft, die derzeit besten Algorithmen vorgestellt und deren Besonderheiten hervorgehoben. Es wird auch ein Ausblick auf neue Methoden im RCPSP gegeben, die sich ebenfalls das

Ziel gesetzt haben, naturwissenschaftliche Themen auf die Optimierung von kapazitätsbeschränkten Projektplänen zu übertragen.

Die Leistungsfähigkeit der Verfahren ist dank der Testdatenbank PSPLIB, welche für die Berechnung von RCPSP Standardtestfälle zur Verfügung stellt, vergleichbar geworden. Dieses Thema bildet den Abschluss des theoretischen Teils dieser Arbeit.

Valls et al. (2008) haben mit ihrem hybriden genetischen Algorithmus einen neuen revolutionären Crossover-Operator eingeführt, der sich während der Evolutionsphase bei der Bestimmung der Crossover-Punkte an der Ressourcenauslastung des Projekts orientiert. Bezeichnenderweise verwenden die derzeit besten genetischen Algorithmen diesen Operator, wenn auch in einer abgewandelten Form.

Nach einer Vorstellung der einzelnen Phasen des Algorithmus und dessen Parametrisierung wird der Aufbau des Programms erklärt. Den Abschluss dieser Arbeit bildet der Analyseteil. Hier werden die Ergebnisse der Standardimplementierung dargebracht und mit Variationen der Applikationsparameter und neuen Programmmethoden, die teilweise auf den anderen genetischen Algorithmen basieren, in eine Datenrelation gesetzt.

1. Netzplantechnik im Projektmanagement

1.1. Projektmanagement

1.1.1. Projektdefinition

Ein wesentliches Merkmal eines Projektes ist dessen Individualität in Bezug auf das Projektziel, die Zeitvorgaben, die Ressourceneinschränkungen, wobei hier zwischen Verbrauchs- und Nutzungsgütern unterschieden werden muss, die Projektorganisation und die Abgrenzung zu anderen Projekten.[1]

Ein Projekt muss exakt beschrieben werden in Hinblick auf den Inhalt, die vorgegebene Projektdauer und die beteiligten Personalressourcen. Dadurch entsteht eine zeitliche und themenbezogene Abgrenzung. Durch den Umstand, dass jedes Projekt individuell ist, besteht bei dieser Beschreibung aber eine gewisse Unsicherheit bzw. können manche Teile nur unbestimmt erfasst werden. Die Komponenten Dynamik und Komplexität tragen zudem dazu bei, dass Projektinhalte und deren Abhängigkeiten einem ständigen Wandel unterliegen können und eine Abgrenzung zu anderen Projekten sich meist schwierig gestaltet. Der Zugang zu Personalressourcen aus verschiedenen Fachbereichen, welche durch unterschiedliche Organisationseinheiten abgedeckt werden, vereinfacht sich durch den erhöhten Bedeutungsgrad des Projektes innerhalb der Organisation.[2]

Die Art eines Projektes kann sehr verschieden sein. Mögliche Unterscheidungsmerkmale können hinsichtlich des Funktionsbereiches (Beteiligungsprojekt, Marketingprojekt, IT-Projekt, Investitionsprojekt, usw.), des Lösungsansatzes (Revolutionsprojekte, Evolutionsprojekte, Expansionsprojekte), der personellen Ressource (Ein- bzw. Mehrpersonenprojekt), der Kundenbeziehung (Interne, externe Projekte oder Mischprojekte), der repetitiven Eigenschaft und dem Schwierigkeitsgrad (Projektkomplexität, Dauer) getroffen werden.[3]

Ziele stellen eine gewünschte Situation dar, die durch das Projekt erreicht werden sollen. Diese können durch ein gewünschtes Ergebnis determiniert sein oder die Durchführung selbst betreffen. Die Bewertung des Ergebnisses kann quantitativ durch

[1] (Olfert, 2010 S. 13), (Schwarze, 2010 S. 13)
[2] (Olfert, 2010 S. 13), (Patzak, et al., 2009 S. 19f), (Schwarze, 2010 S. 13)
[3] (Olfert, 2010 S. 15f), (Patzak, et al., 2009 S. 21f)

Maßzahlen erfolgen oder es werden qualitative Ziele verfolgt, die nur indirekt messbar sind (z.B. die Steigerung des Betriebsklimas). Projekte können langfristig (strategisch), mittelfristig (taktisch) oder operativ (kurzfristig) ausgerichtet werden.[4] Grundsätzlich ist ein Projekt durch die Zielfunktion gekennzeichnet, ohne Fehler, ökonomisch, vollständig, frei von Konflikten und ressourcenoptimiert zu einem bestimmten Termin fertig gestellt zu werden und den gesetzten Inhalt zu erfüllen.[5]

1.1.2. Management von Projekten

Zielorientierte Projekte bedürfen einer Organisation, an deren erster Stelle die Projektplanung steht. In der Projektplanung wird u.a. der Ablauf des Projektes definiert. Es werden Ressourcen und Termine festgelegt, Aufgaben erstellt und Kosten geplant. In einem weiteren Schritt wird das Projekt organisiert. Hier werden die Aufgaben für die Ressourcen festgelegt, die Projektrollen definiert und die Rollen des Projektteams und deren Informationsfluss und Kommunikationsmethoden determiniert. Projekte werden geführt, damit der Fokus des Projektziels klar bleibt und nicht verloren geht, Mitarbeiter motiviert und weiterentwickelt werden, deren Zusammenarbeit gefördert wird und Entscheidungen getroffen werden können. [6]

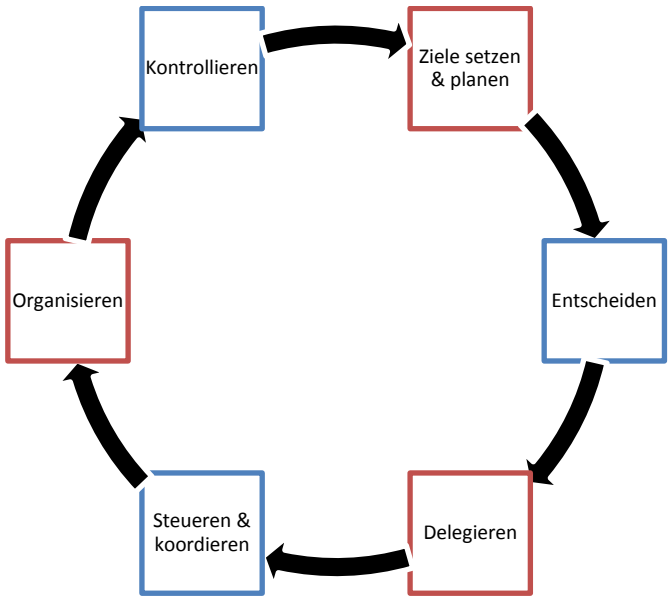

Abbildung 1: Aufgaben des Projektmanagements (Schwarze, 2010 S. 17)

[4] (Olfert, 2010 S. 18)
[5] (Schwarze, 2010 S. 15)
[6] (Patzak, et al., 2009 S. 24f)

Der Status des Projektes muss dokumentiert werden. Veränderungen sind Teil eines Projektes, die mit Steuerungsmechanismen in Hinblick auf Qualität, Ressourcen und Terminen bewältigt werden.[7] Diese Aufgaben des Projektmanagements sind in Abbildung 1 dargestellt.

Projekte sind ein zentraler Bestandteil im unternehmerischen Bereich. Deren Dimensionen wachsen stetig und es ist deshalb von immenser Bedeutung, diese effizient zu planen und organisieren und mit entsprechenden Vorgaben zu steuern. Des Weiteren gibt es viele externe Einflussfaktoren, die die Wichtigkeit von Projekten in Unternehmen unterstreichen. Dazu zählen der stetig steigende Wettbewerb, eine stärkere Fokussierung auf Kundenbedürfnisse und eine globalere, stärker verteilte Ausrichtung der Unternehmensorganisation. Dadurch werden auch Unternehmen selbst mehr ablauf- oder prozessgesteuert, wobei die einzelnen Aktivitäten als Prozess definiert werden können. Die Optimierung solcher Prozesse bedingt deren Organisation, woraus dann Geschäftsprozesse entstehen. Stehen diese Prozesse in einem Zusammenhang, spricht man von einer Vorgangskette. Die Darstellung solcher Geschäftsprozesse (Abbildung 2) weist eine große Ähnlichkeit mit Projektvorgängen auf, die mittels Netzplänen dargestellt werden.[8]

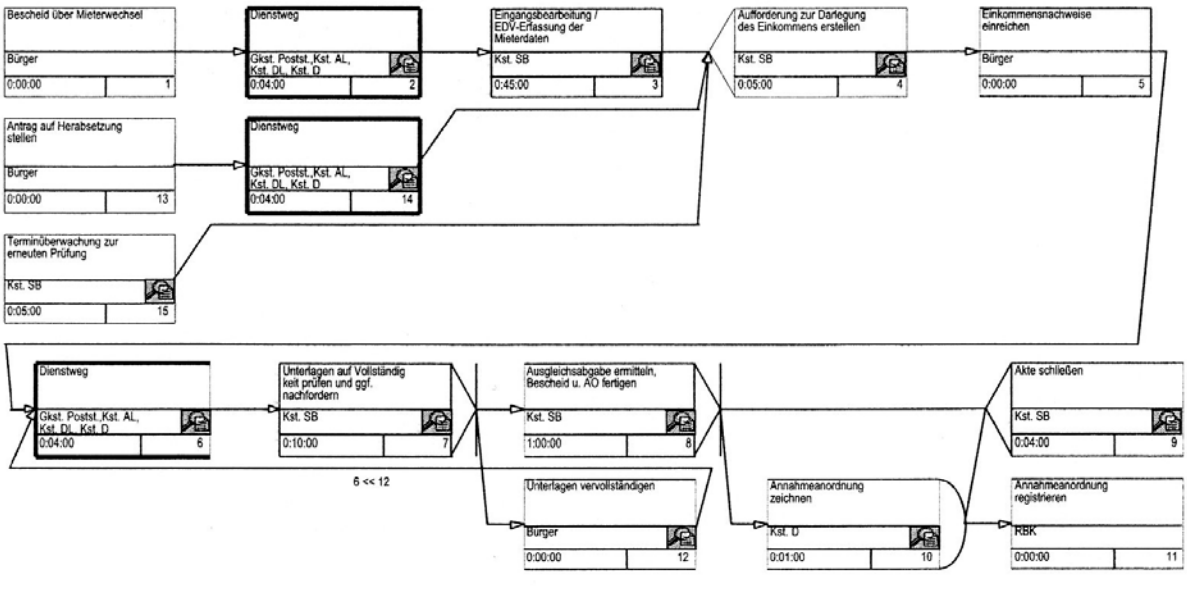

Abbildung 2: Geschäftsprozess (Schwarze, 2010 S. 21)

[7] (Patzak, et al., 2009 S. 24f)
[8] (Schwarze, 2010 S. 20f)

Diese Arbeit setzt den Fokus auf die Projektplanung, insbesondere die Netzplantechnik als eine Methode dessen. In Kapitel 1.1.3 wird nun eine Übersicht der vorhandenen Ablauf- und Terminplanungsmethoden gezeigt.

1.1.3. Darstellungsformen des Projektmanagements

Die einzelnen Aufgaben des Projektes besitzen eine bestimmte Durchführungsdauer und sind, durch Abhängigkeiten mit anderen Aufgaben, in einer gewissen Anordnung durchzuführen. Durch die Berücksichtigung dieser Bedingungen kann das Projekt vom Start bis zum Ende zeitlich und ablauforientiert geplant werden.[9] Die Beschreibung der folgenden Darstellungsmöglichkeiten ist nach deren Komplexität gestaffelt, wobei die einfachsten Formen zuerst beschrieben werden. Der Einsatz ist abhängig vom Projektumfang, dessen Komplexität der Abhängigkeiten und der Hilfsmittel (z.B. computerunterstützte Planung zur Verringerung des Berechnungsaufwandes), welche eingesetzt werden können.[10]

1.1.3.1. Geschwindigkeitsdiagramm

Das Geschwindigkeitsdiagramm stellt einen groben Überblick des Projektes dar. Der Projektablauf wird linear dargestellt, wobei hier eine Gegenüberstellung von Kenngrößen (Weg, Mengen, Leistung) in einem zeitlichen Zusammenhang vorgenommen wird. Diese Darstellungsform ist auf kleine Projekte oder Übersichtsvorgänge anwendbar. Vorteilhaft ist die Gegenüberstellung der einzelnen Projektabschnitte, in Bezug auf den zeitlichen Fortschritt, relativ zu den erbrachten Leistungen, die jedoch vorab definierbar sein müssen (z.B. in Abbildung 3 die erbrachte Kilometerleistung pro Woche) [11]

[9] (Patzak, et al., 2009 S. 248)
[10] (Schwarze, 2010 S. 103)
[11] (Patzak, et al., 2009 S. 250)

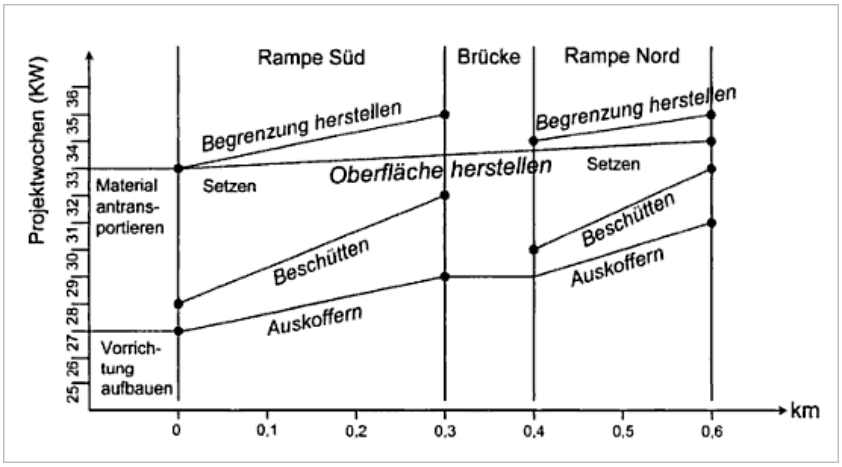

Abbildung 3: Beispiel für ein Zeit-Weg Diagramm (Patzak, et al., 2009 S. 250)

1.1.3.2. Listen

Diese terminorientierten Verfahren können grundsätzlich in zwei Kategorien unterteilt werden. Eine Orientierung an Projektphasen findet in den Verfahren der Meilenstein-Terminlisten, Phasenpläne oder Meilensteinplanung eine Anwendung. In der Vorgangsterminliste oder Vorgangsliste ist die Aggregationsebene viel detaillierter und ist auf den Vorgang selbst fokussiert.[12]

<u>Projektphasen</u>

Die Projektphasen-orientierten Verfahren bieten eine Planungsmöglichkeit auf einer sehr rudimentären Planungsebene, welche aber den Vorteil der Übersichtlichkeit bietet. Das Stage-Gate-Modell nach Cooper, auch Quality-Gate-Technik genannt, dient der Darstellung von solch einem übersichtlichen Projektablauf. In diesem Verfahren werden wichtige Meilensteine (Gates) bestimmt und die dazwischen liegenden Kernaufgaben des Projektes, welche einer Projektphase (Stage) zugeordnet sind, definiert. Durch die sehr verdichtete Planungsebene wird der Aufwand für Planung und Steuerung minimiert. Es existiert eine bessere Übersicht über das Projekt. Entscheidungen können klar getroffen werden, Prioritäten leicht gesetzt werden und ein Projektfortschritt schnell für einen Kunden aufbereitet werden. Zudem wird den

[12] (Schwarze, 2010 S. 127f)

Teammitgliedern eine Selbstständigkeit für die einzelnen Vorgänge gegeben, die innerhalb der Meilensteine liegen.[13]

Terminlisten

Diese Listen beinhalten die Vorgänge im Detail. Hier werden im ersten Schritt alle Aufgaben, mit deren Dauer und optional mit einem definierten Anfangszeitpunkt, des Projektes gelistet. Durch den Anfangszeitpunkt und die Dauer ergibt sich der mögliche Endzeitpunkt eines Vorganges. Der Anfangszeitpunkt kann durch den Endzeitpunkt eines anderen Vorganges beeinflusst werden, wenn er von der Fertigstellung des Vorgangs abhängig wird. [14]

Kurz-zeichen	Projektvorgang	Vorgangsdauer Tage	Anfang	Ende
A	Angebot für Festlokal einholen	10	0	10
E	Einladungen drucken	10	33	43
V	Einladungen versenden	2	43	45
F	Festlokal auswählen	6	10	16
E	Festredner auswählen	3	0	3
S	Showangebot ermitteln	15	0	15
W	Shownummern auswählen	10	15	25
L	Vertragsabschluss mit Festlokal	5	16	21
X	Verträge mit Shownummern abschließen	8	25	33
Z	Zusage Festredner einholen	12	3	15

Tabelle 1: Beispiel für eine Terminliste (Olfert, 2010 S. S.103)

1.1.3.3. Balkendiagramm

Die Balkendiagrammverfahren stellen die einzelnen Vorgänge des Projektes in einem Diagramm dar. Hierbei werden die Vorgänge als Balken auf einer Zeitachse visualisiert. Diese Methode eignet sich nur für den Einsatz in Projekten, die aus wenigen Vorgängen bestehen, da die Übersicht und Lesbarkeit mit vielen Vorgängen verloren geht. Unterschieden werden zwei verschiedene Arten: Gantt-Technik und PLANNET-Technik (auch Balkenplan bzw. vernetzter Balkenplan genannt[15]). Die PLANNET-Technik grenzt sich von der Gantt-Technik, die nach Henry Lawrence Gantt benannt wurde, durch die Möglichkeit ab, Pufferzeiten darzustellen und Abhängigkeiten von Vorgängen zeigen zu können.[16]

[13] (Kerzner, 2009 S. 66f)
[14] (Olfert, 2010 S. 103)
[15] (Patzak, et al., 2009 S. 255)
[16] (Olfert, 2010 S. 104ff)

GANTT-Technik

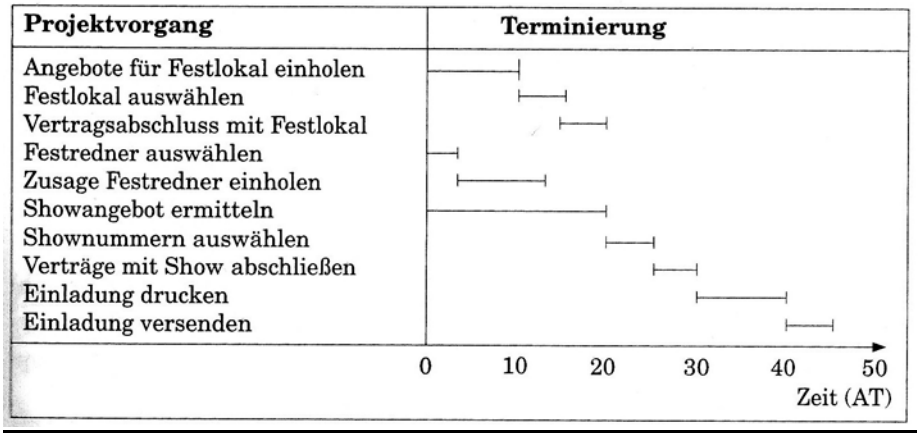

Abbildung 4: Gantt-Diagramm (Olfert, 2010 S. 105)

PLANNET-Technik

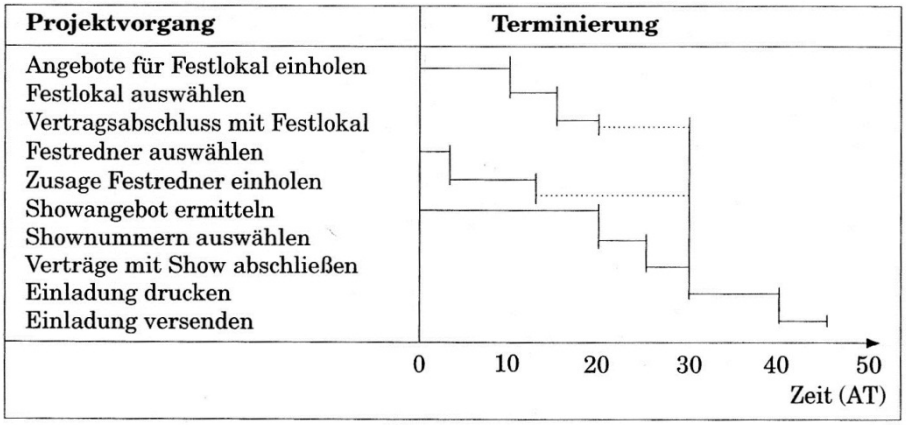

Abbildung 5: PLANNET-Diagramm (Olfert, 2010 S. 106)

1.1.3.4. Netzplantechnik in der Projektplanung

Diese Methode zeigt die Abhängigkeit der Vorgänge in einem Graph. Daher liegt der Netzplantechnik die Graphentheorie (siehe 1.2.2) zugrunde. Mittels der Netzplantechnik können Projekte in unterschiedlichen Arten dargestellt werden[17]:

Der vorgangsorientierte Netzplan stellt keine Ereignisse, sondern nur Vorgänge (z.B.: Arbeitsschritt 1) dar. Hierbei kann unterschieden werden zwischen:

- Vorgangspfeilnetz (Grundverfahren: CPM, siehe Kapitel 1.2.1)
- Vorgangsknotennetz (Grundverfahren: MPM, siehe Kapitel 1.2.1)

Der ereignisorientierte Netzplan stellt bestimmte Ereignisse im Projekt dar, anstatt die Vorgänge zu zeigen (z.B.: Arbeitsschritt 1 fertig gestellt).

- Ereignisknotennetz (Grundverfahren: PERT, siehe Kapitel 1.2.1)

Zusätzlich zu den Darstellungsformen der Netzplantechnik können auch Berechnungsmethoden verwendet werden. Grundsätzlich besteht die Netzplantechnik aus einem vierstufigen Prozess: [18]

- Strukturanalyse zur Bestimmung der abhängigen Vorgänge
- Zeitanalyse für die Errechnung der Projektdauer im Allgemeinen und der möglichen Beginn- und Endzeitpunkte der Vorgänge sowie deren Pufferzeiten
- Kapazitätsanalyse für die Optimierung der Vorgangsanordnung bei bestehenden Ressourcenrestriktionen
- Kostenanalyse für die Beschleunigung des Projektes bei minimalen Mehrkosten

Eine detaillierte Beschreibung der Darstellungsformen, Prozesse und Methoden ist in dem nächsten Kapitel zu finden.

[17] (Schwarze, 2010 S. 29ff)
[18] (Zimmermann, 2008 S. 365f)

1.2. Netzplantechnik

1.2.1. Entstehung[19]

Die Grundlagen der drei bekanntesten Verfahren der Netzplantechnik wurden von 1956-1958, unabhängig voneinander, in Frankreich und der USA geschaffen. Alle Ansätze hatten das Ziel die Projektplanung zu verbessern, dabei die Projekte in ihre Vorgänge zu zerlegen, diese graphisch mit Abhängigkeiten darzustellen und dass Terminveränderungen automatisch berechenbar sind. Generell werden diese drei Methoden unter Netzplantechnik subsumiert:

Critical Path Methode (CPM)

1956 bis 1957 entwickelten die Firmen du Pont und Remington Rand Univac in den USA eine Methode, um den Stillstand einer chemischen Fabrik zu reduzieren. Die Ursprungsbezeichnung der Critical Path Methode war „Project Planning and Scheduling System“.

Program Evaluation and Review Technique (PERT)

Das PERT Verfahren wurde 1958 von der amerikanischen Marine zusammen mit Lockheed (damals: Loughead Aircraft Manufacturing Company) und Booz, Allen & Hamilton (Beratungsfirma) für das Polaris Raketensystem für eine Projektkoordination von ca. 11 000 beteiligten Firmen entwickelt. Daraus ist ein Standardverfahren entstanden, welches auch für zukünftige, staatliche Projekte verwendet wurde.

METRA Potential Methode (MPM)

Die Societe d'Economie et de Mathematique Appliquees (Beratungsfirma der METRA-Gruppe) hat 1958 in Frankreich für Electricité France eine Projektplanungsmethode erforscht um den Bau eines Atomkraftwerks unterstützen zu können.

[19] (Bradke, 2003 S. 82f)

1.2.2. Graphentheorie

Das grundlegende Element eines Graphen bilden die Knoten oder Ecken, welche mittels Kanten oder Strecken[20] miteinander verbunden werden können. Durch die entsprechende Verwendung der Kanten wird der Typ des Graphen bestimmt. Weisen die Kanten eine Richtung auf, so spricht man von einem **gerichteten Graphen** oder Digraphen, welcher dadurch einen Anfangsknoten (oder Vorgängerknoten[21]) und Endknoten (Nachfolgerknoten[21]) besitzt (siehe Abbildung 6: gerichteter Graph). Gerichtete Kanten werden auch als Pfeile bezeichnet. Andernfalls spricht man von **ungerichteten Graphen**, welche nur Endknoten besitzen (siehe Abbildung 7: ungerichteter Graph).[22]

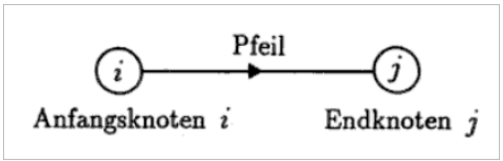

Abbildung 6: gerichteter Graph (Neumann, et al., 2002 S. 178)

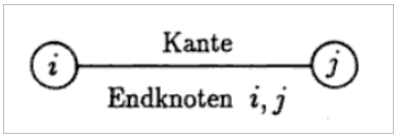

Abbildung 7: ungerichteter Graph (Neumann, et al., 2002 S. 178)

Graphen können auch durch Schlingen miteinander verbunden sein, durch parallele Kanten bzw. Pfeile oder durch entgegengesetzt gerichtete Pfeile. Letztere werden auch als antiparallel bezeichnet. Wenn man diese Typen von Graphen nicht in Betracht zieht, so ist jede Kante in einem Graph durch zwei eindeutige Endknoten bestimmt. Ein Pfeil anstatt einer Kante bringt zusätzlich die Unterscheidung in Anfangs- und Endknoten.[23]

Für die Darstellung von Graphen gibt es verschiedene Möglichkeiten. In den folgenden Kapiteln (1.2.2.1 - 1.2.2.5)[24] wird von einem ungerichteten bzw. gerichteten Graph (G bzw. D) ausgegangen, welcher n Knoten ($v \in V$) mit m Kanten ($e \in E$) besitzt.

[20] (Runzheimer, et al., 2005 S. 143)
[21] (Hansen, et al., 2001 S. 1019)
[22] (Neumann, et al., 2002 S. 177)
[23] (Neumann, et al., 2002 S. 178)
[24] (Zimmermann, 2008 S. 340-342)

1.2.2.1. Graphische Darstellung

Für eine übersichtliche Darstellung der Abläufe und Strukturen wird die graphische Form gewählt. Ein Digraph kann folgendermaßen visualisiert werden:

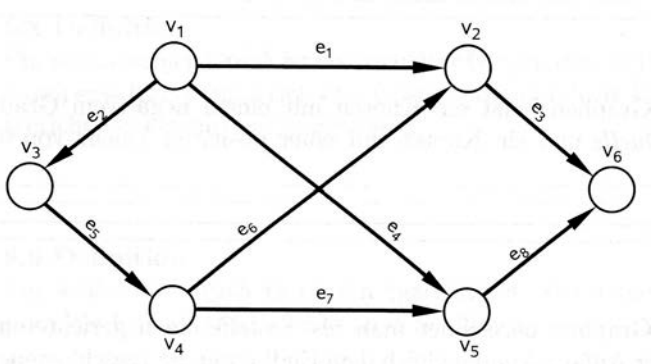

Abbildung 8: Digraph (Zimmermann, 2008 S. 340)

1.2.2.2. Abbildung (Funktion)

Der gerichtete Graph aus Abbildung 8 kann durch ein Quadrupel (V, E, Ψ_1 = Funktion 1, Ψ_2 = Funktion 2) dargestellt werden, wobei Ψ_1 die Zuordnung der Pfeile zu den Anfangsknoten beschreibt und $\Psi 2$ die Endknoten-Pfeil Beziehung darlegt. Beide Funktionen können zu einem Tripel (V, E, Ψ) zusammengeführt werden[25]:

$$\Psi_1 : \begin{cases} e_1 \to v_1 \\ e_2 \to v_1 \\ e_3 \to v_2 \\ e_4 \to v_1 \\ e_5 \to v_3 \\ e_6 \to v_4 \\ e_7 \to v_4 \\ e_8 \to v_5 \end{cases} \quad \Psi_2 : \begin{cases} e_1 \to v_2 \\ e_2 \to v_3 \\ e_3 \to v_6 \\ e_4 \to v_5 \\ e_5 \to v_4 \\ e_6 \to v_2 \\ e_7 \to v_5 \\ e_8 \to v_6 \end{cases} \quad \text{beziehungsweise} \quad \Psi : \begin{cases} e_1 \to v_1 \\ e_2 \to v_1 \\ e_3 \to v_2 \\ e_4 \to v_1 \\ e_5 \to v_3 \\ e_6 \to v_4 \\ e_7 \to v_4 \\ e_8 \to v_5 \end{cases}$$

Abbildung 9: Funktionsdarstellung (Zimmermann, 2008 S. 341)

1.2.2.3. Kanten- oder Pfeillisten

Durch die Restriktion, parallele Pfeile (bzw. Kanten) nicht zuzulassen, da diese nicht erkennbar sind, kann ein Digraph auch mittels Pfeilliste (Kantenliste im Falle des

[25] (Zimmermann, 2008 S. 341)

ungerichteten Graphen) abgebildet werden:

(v_1, v_2)
(v_1, v_3)
(v_2, v_6)
(v_1, v_5)
(v_3, v_4)
(v_4, v_2)
(v_4, v_5)
(v_5, v_6)

Abbildung 10: Kanten- /Pfeilliste (Zimmermann, 2008 S. 341)

1.2.2.4. Adjazenzmatrix[26]

Matrizen sind ebenfalls Darstellungswerkzeuge für gerichtete und ungerichtete Graphen.[27] Die Adjazenzmatrix (A), eine symmetrische Matrix (m x m), da Knoten zu Knoten Verbindungen dargestellt werden, ist binär organisiert. Der Wert 1 legt eine Verbindung von v_i nach v_j fest (i, j = 1, …, m), ansonsten wird der Wert 0 verwendet.

$$\boldsymbol{A} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ \text{Spalten-summen} \end{array} \begin{array}{c} \begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array} \\ \left(\begin{array}{cccccc} 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 1 & 2 & 2 \end{array}\right) \end{array} \begin{array}{c} \text{Zeilensummen} \\ 3 \\ 1 \\ 1 \\ 2 \\ 1 \\ 0 \end{array}$$

Abbildung 11: Adjazenzmatrix (Zimmermann, 2008 S. 342)

Die Auswertung der Matrix erfolgt zeilen- und spaltenweise, wobei jeweils Summen gebildet werden, welche die Pfeilanzahl des Digraphen repräsentieren. Die Zeilensummen determinieren die Anzahl ausgehender Pfeile der Knoten und die Spaltensummen dessen Eingangsgrade.

1.2.2.5. Inzidenzmatrix[28]

Die Inzidenzmatrix (H) ist im Gegensatz zur Adjazenzmatrix eine unsymmetrische Matrix (m x n Matrix) und stellt bei einem Digraphen drei Ausprägungen dar. Ein, von

[26] (Zimmermann, 2008 S. 342)
[27] (Neumann, et al., 2002 S. 178)
[28] (Zimmermann, 2008 S. 342)

einem Knoten v_i, ausgehender Pfeil e_j wird mit dem Wert 1 beschrieben bzw. mit -1 wenn dieser in einen Knoten eingeht. 0 repräsentiert wie in der Adjazenzmatrix keine Verbindung. Ein ungerichteter Graph ist absolut inzident[29] und benötigt somit keine Unterscheidung zwischen 1 und -1.

$$H = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{array} \begin{array}{c} \begin{array}{cccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & -1 & 1 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 \end{pmatrix} \end{array}$$

Abbildung 12: Inzidenzmatrix eines Digraphen (Zimmermann, 2008 S. 342)

1.2.3. Der Graph als Netz

Vorgängerlose Knoten werden als Quelle (oder Wurzelknoten[30]) bezeichnet[31]. Endknoten des Graphen ohne weitere Nachfolger werden als Senke definiert. Ist der Graph zudem zyklenfrei, so wird von einem Netz gesprochen, welches durch gewichten der Pfeile bewertet werden kann und damit in ein Netzwerk übergeführt wird.[32]

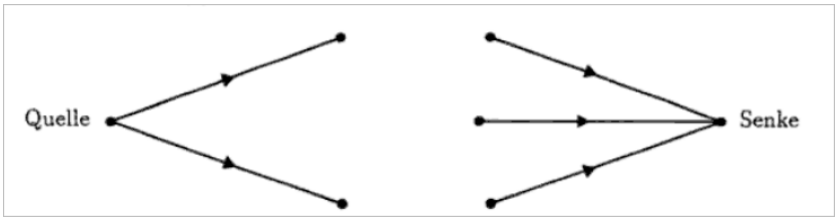

Abbildung 13: Quelle und Senke (Neumann, et al., 2002 S. 180)

1.2.4. Strukturplanung

1.2.4.1. Erhebungsanalyse

Die Grundlage für die Erstellung einer Projektplanung bildet die Erhebung der notwendigen Vorgänge, deren Anordnung und Abhängigkeiten, die Vorgangsdauer der einzelnen Aktivitäten und eventuelle Ressourcenbeschränkungen. Am Anfang des Projektes steht der Projektstrukturplan. Dadurch kann das Projekt in sinnvolle Teilbereiche (Teilprojekte) zerlegt werden und in diesen wiederum in Subprojekte oder

[29] (Neumann, et al., 2002 S. 179)
[30] (Hansen, et al., 2001 S. 1021)
[31] (Neumann, et al., 2002 S. 180)
[32] (Zimmermann, 2008 S. 345)

Subaktivitäten. Der Detailgrad der Ausführung des Projektstrukturplans hängt von der Größe und Komplexität des Projektes ab. Es wird eine Unterscheidung zwischen einem funktionsorientierten Projektstrukturplan, in dem die Untergliederung nach Funktionskategorien wie Abteilung, Bereich o.ä. erfolgt, und einem objektorientierten Projektstrukturplan, der eine Projektstruktur auf inhaltlicher Basis darstellt, vorgenommen. Dabei sollte in beiden Fällen beachtet werden, dass die Unterkategorien immer gleichwertige Ebenen aufweisen.[33]

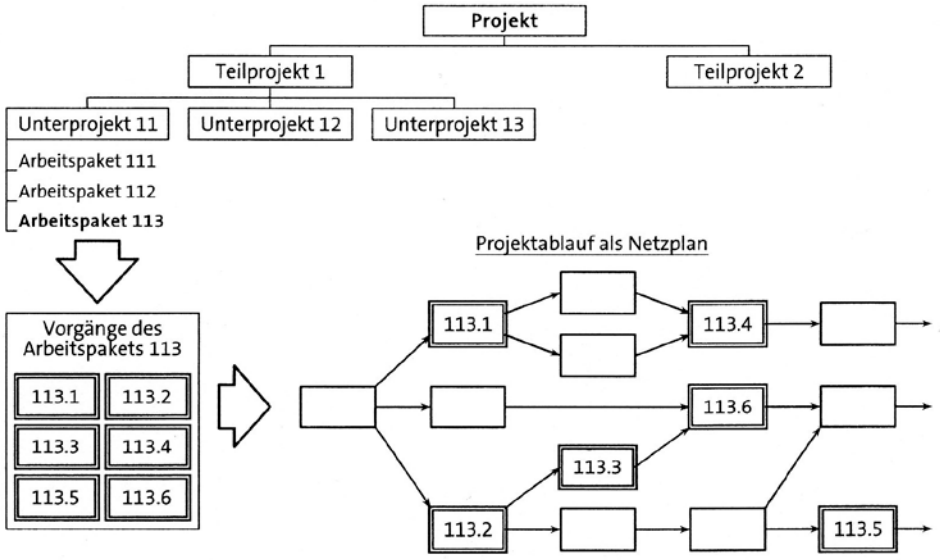

Abbildung 14: Projektstrukturplan und Netzplan (Schwarze, 2010 S. 82)

Die Relation zwischen Projektstrukturplan und Netzplan wird in Abbildung 14 dargestellt

1.2.4.2. Ablaufanalyse

Jedes Projekt besteht aus einzelnen Vorgängen, welche einen Anfangs- und Endzeitpunkt, mit einer daraus resultierenden Dauer, besitzen, Ressourcen beanspruchen und dabei Kosten verursachen. Ereignisse in Projekten repräsentieren die Zeitpunkte der Vorgänge, die bei besonderer Gewichtung als Meilensteine bezeichnet werden. Vorgänge können in gewisser Reihenfolge zueinander oder in Abhängigkeit von Ereignissen stehen. Dazu werden die Begriffe Vorgänger, im Falle eines vorhergehenden Vorganges, bzw. Vorereignis, im Falle eines vorgelagerten

[33] (Schwarze, 2010 S. 79ff)

Ereignisses, determiniert. Diese Beziehungen können logisch begründet oder kapazitäts- bzw. zeitbeschränkt verursacht sein.[34]

Die einzelnen Vorgänge, deren Abhängigkeiten und Zuordnungen zu den Ressourcen können in einer Vorgangsliste (vgl. Kapitel 1.1.3.2 Listen) eingetragen werden.

1.2.4.2.1. Vorgangsknotennetz[35]

In einem Vorgangsknotennetz werden Vorgänge, oder auch Aktivitäten genannt, durch Vorgänger- und Nachfolgerbeziehungen visualisiert. Diese Beziehungen werden durch Pfeilverbindungen erreicht. Der Vorgang enthält die für die Planung notwendigen Informationen (Name der Vorgangs, Dauer, frühester Endzeitpunkt, usw.). Abhängig von dem Einsatzgebiet des Plans, kann diese Information sehr detailliert ausgeführt sein. Die Standarddarstellung zeigt eine Normalfolge von Aktivitäten. Die Darstellungsmöglichkeiten weiterer Varianten sind in Kapitel 1.2.4.2.4 angeführt. Abgleitet von dem Einsatz dieser Darstellungstechnik in der Metra-Potential-Methode (MPM) ist daraus der Name MPM-Netzplan als Synonym für das Vorgangsknotennetz entstanden.

Ein derartiger Netzplan verbindet einen Nachfolger eines Vorganges mit einem richtungsweisenden Pfeil, der zwischen zwei Vorgängen nur einmal existiert und keine Zyklen hervorrufen darf. Die Richtung der Pfeile ist innerhalb eines Netzplanes einheitlich gestaltet. Üblicherweise beginnt ein Projekt an einer Quelle und endet mit einer Senke, die einen künstlichen Vorgang mit der Dauer Null repräsentieren.

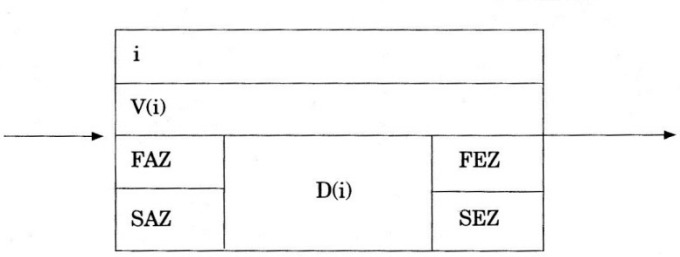

Abbildung 15: Darstellung eines Vorgangs (Olfert, 2010 S. 112)

[34] (Schwarze, 2010 S. 83ff)
[35] (Schwarze, 2010 S. 95f)

Die in Abbildung 15 verwendeten Abkürzungen, können in Kapitel 1.2.5 (Tabelle 2) nachgeschlagen werden. Weitere Darstellungsformen und Zeitberechnungen dieser Arbeit basieren auf dem Vorgangsknotenprinzip (Metra-Potential-Methode).

1.2.4.2.2. Vorgangspfeilnetz

In dieser Darstellungsart der Netzplantechnik repräsentieren die Pfeile des Digraphen die einzelnen Vorgänge des Projektes, welchen jeweils ein Startereignis vorgelagert ist und welche von einem Endereignis gefolgt werden. Abbildung 6 aus Kapitel 1.2.2 stellt dieses Element dar. Bei zwei aufeinanderfolgenden Vorgängen werden der Endzeitpunkt des Vorgängers und der Startzeitpunkt des Nachfolgers zu einem Ereignis zusammengefasst.

Das Vorgangspfeilnetz wird in der neueren Literatur nicht mehr verwendet.

1.2.4.2.3. Ereignisknotennetz und Mischformen

Diese Darstellungsform der Netzplantechnik stellt im Gegensatz zu den Formen aus Kapitel 1.2.4.2.1 und 1.2.4.2.2 keine Vorgänge dar, sondern Meilensteine. Diese Technik ist einerseits eine gute Methode, um das Projekt übersichtlich darzustellen und andererseits dann notwendig, wenn aufgrund der Projektanalyse keine Möglichkeit einer detaillierten Vorgangsbeschreibung vorhanden ist. Eine Überleitung der bisher erwähnten Netzplantypen in diese Form ist durchführbar, jedoch kann dieser Plan nicht zurückgeführt werden. Diese Überleitung ist in Abbildung 16 dargestellt, wobei die schraffierten Felder die Meilensteine des Netzplans repräsentieren.[36]

Es ist in weiterer Folge auch möglich, Vorgänge und Meilensteine in einem Netzplan darzustellen. Diese Form der Darstellung dient der größeren Projektübersicht von Vorgangsknotennetzen. Bei einer geringen Anzahl von Vorgängen in einem Netzplan, können einzelne Vorgänge auch gewissen Rubriken zugeordnet werden. Bei einer großen Anzahl von Aktivitäten in einem Projekt wird der Netzplan aber nicht mehr lesbar, da es durch die gruppierte Anordnung der Vorgänge zu vielen Überschneidungen der Verbindungspfeile kommen kann.[37]

[36] (Schwarze, 2010 S. 103f)
[37] (Schwarze, 2010 S. 105)

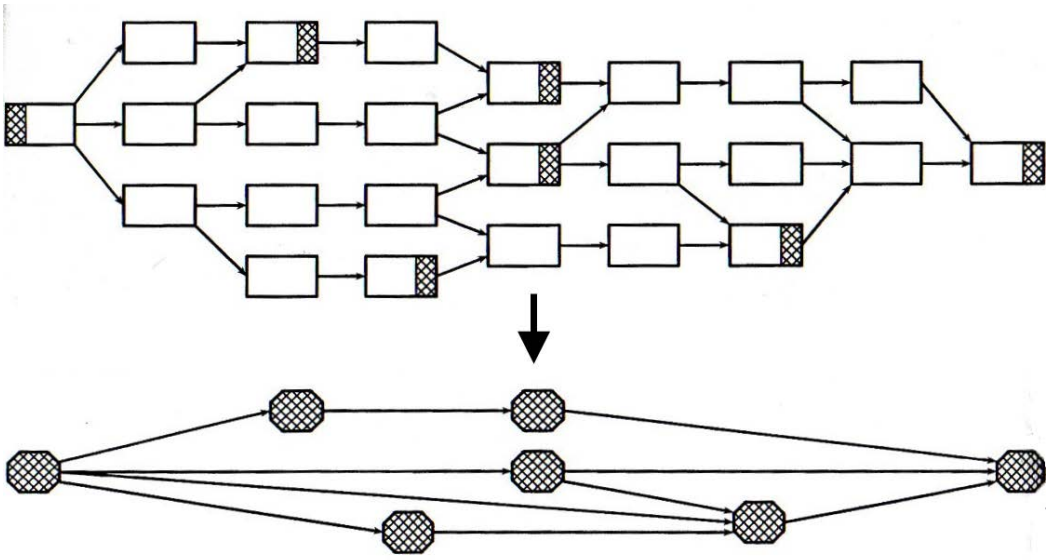

Abbildung 16: Vorgangknotennetzplans als Meilensteinplan (Schwarze, 2010 S. 103f)

1.2.4.2.4. Anordnungsbeziehungen im Netzplan

Im Normalfall ist die Reihenfolge in einem Netzplan so festgelegt, dass Vorgang B unmittelbar nach dem Ende von Vorgang A beginnen kann (Normalfolge). Die Beziehung zwischen zwei Vorgängen kann mit zwei Einflussfaktoren verändert werden. Der Zeitabstand Z kann mit einem Maximalwert begrenzt werden oder durch einen Minimalwert einen Mindestabstand erzeugen. Dabei können auch negative Werte verwendet werden, um Aktivitäten bereits zeitverzögert vor dem Ende des Vorgängers beginnen lassen zu können. Mit der Anordnungsfolge F wird festgelegt, welche Ereignisse der Vorgänge für die Anordnung der Aktivität B maßgeblich sind. Zusätzlich zu der bereits bekannten Normalfolge gibt es die Anfangsfolge, Endfolge und Sprungfolge.[38] Eine Kombination dieser Faktoren ist in Abbildung 17 dargestellt.

[38] (Schwarze, 2010 S. 109ff)

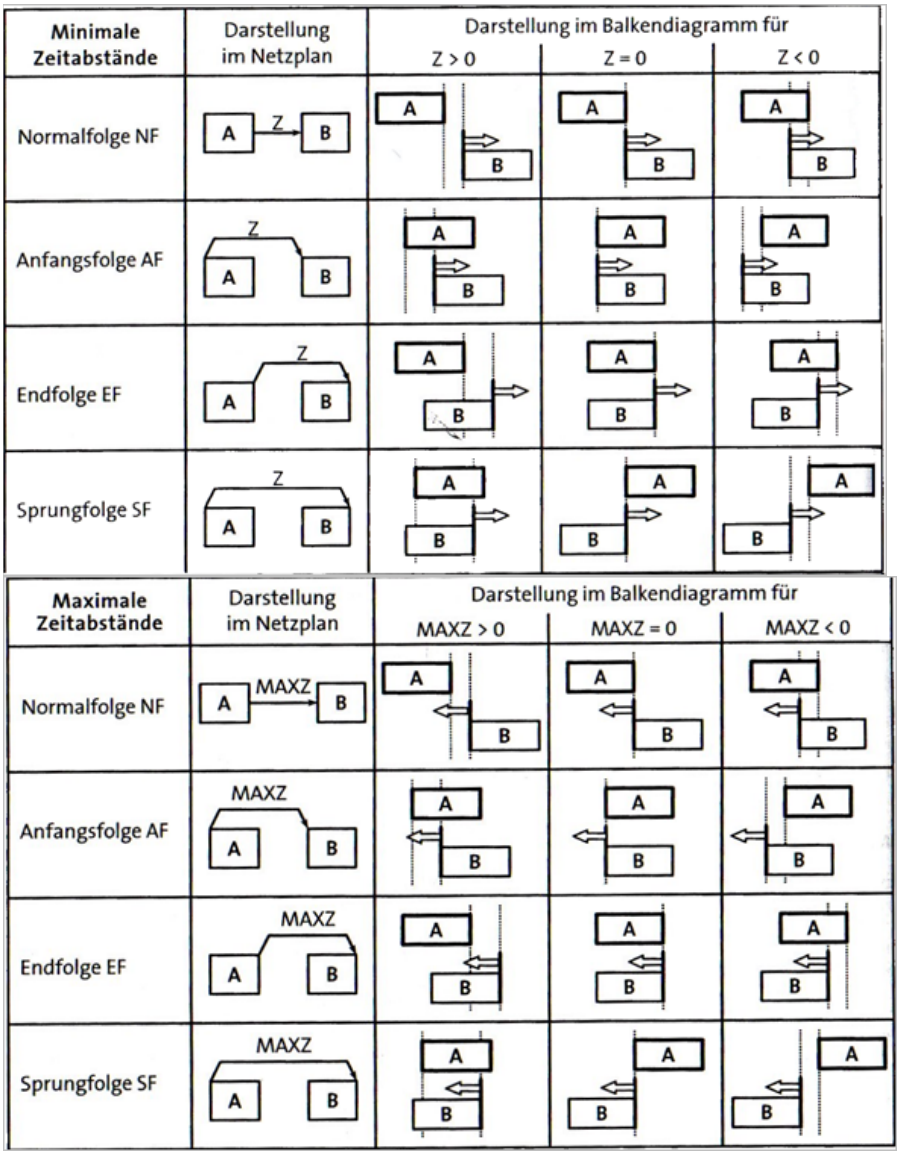

Abbildung 17: Anordnungsbeziehungen im Netzplan (Schwarze, 2010 S. 116)

1.2.5. Zeitplanung

Die Berechnung, die die Kenntnis der Dauer aller Vorgänge, dessen Vorgängerbeziehungen und ihrer Vorgangszeitpunkte und Pufferzeiten voraussetzt, erfolgt in drei Phasen. Tabelle 2 zeigt die in der Zeitplanung verwendeten Begriffe in einer Übersicht. Alle Schritte der Zeitplanung sind in Kapitel 1.2.5.1 - 1.2.5.3 beschrieben und zudem in Abbildung 18 als Programmablauf dargestellt.

Abkürzung	Bezeichnung
FAZ	Frühestmöglicher Anfangszeitpunkt
FEZ	Frühestmöglicher Endzeitpunkt
SAZ	Spätestmöglicher Anfangszeitpunkt
SEZ	Spätestmöglicher Endzeitpunkt
d	Dauer
GP	Gesamte Pufferzeit
FP	Freie Pufferzeit
FRP	Freie Rückwärts-Pufferzeit
UP	Unabhängige Pufferzeit
V(i)	Vorgänger der Aktivität
N(i)	Nachfolger der Aktivität

Tabelle 2: Begriffsbestimmungen in der Zeitplanung (Schwarze, 2010 S. 145)

1.2.5.1. Phase 1: Vorwärtsrechnung

In dem ersten Teil der Zeitplanung werden die FAZ der Vorgänge bestimmt. Gleichzeitig mit der Bestimmung eines FAZ, kann auch der FEZ durch FEZ = FAZ + d ermittelt werden. Der Planvorgang ist ein iteratives Verfahren, bei dem jeweils nur Aktivitäten mit bereits geplanten Vorgängern der Berechnung unterliegen. Der FAZ eines Vorgangs ist aus dem Maximum der FEZ aller Vorgänger zu errechnen. Dies wird solange durchgeführt, bis alle Aktivitäten einen FAZ (und implizit einen FEZ) besitzen. Eine Formalisierung dieser Regel ist in der Nebenbedingung (2) der Formel 9 (Kapitel 2.2) zu finden. Wenn mehrere Aktivitäten mit abgeschlossenen Vorgängern in der Liste E_n zur Verfügung stehen, so wird die Aktivität mit der geringsten Nummer gewählt. Dieser Vorgang ist in Abbildung 18 dargestellt.[39]

1.2.5.2. Phase 2: Rückwärtsrechnung

Ausgehend von der letzten Aktivität, werden nun die spätest möglichen Vorgangszeitpunkte berechnet. Die Werte der letzten Aktivität (Senke) werden wie die der ersten Aktivität (Quelle) ermittelt. Hierbei gilt FAZ=FEZ=SEZ=SAZ ($d_0=d_J=0$). Analog zu der Vorwärtsrechnung, werden in absteigender Reihenfolge die Aktivitäten gesucht,

[39] (Schwarze, 2010 S. 145ff)

deren Nachfolger bereits einen SAZ besitzen und selbst noch keine SEZ erhalten haben. In der Entscheidungsliste E_n, deren Mitglieder bereits abgeschlossene Vorgänger besitzen, wird die Aktivität mit der höchsten Nummer gewählt, wobei dieses Auswahlkriterium nicht zwingend notwendig ist, denn eine zufallsgenerierte Wahl ist ebenfalls möglich. Die Errechnung des SEZ des Vorgangs ergibt sich aus dem Minimum der SAZ seiner Nachfolger. Sobald der SEZ oder SAZ des Vorgangs bekannt ist, kann auch dessen gesamter Puffer GP durch GP = SEZ-FEZ (oder SAZ-FAZ) bestimmt werden. Da allerdings nicht alle Pufferzeiten zu diesem Zeitpunkt bestimmt werden können, so wird die gesamte Pufferzeit ebenfalls in Phase 3 bestimmt. Alle Vorgänge die nach der Vorwärts- und Rückwärtsrechnung eine gesamte Pufferzeit von GP=0 aufweisen, werden als **kritische Vorgänge** bezeichnet, jene die eine geringe oder im Projekt vorgegebene Pufferzeit unterschreiten als **subkritische Vorgänge**, die zusammen den kritischen bzw. subkritischen Weg bilden. Eine Verzögerung eines kritischen Vorgangs hat eine unmittelbare Verlängerung der Projektdauer zur Folge.[40]

1.2.5.3. Phase 3: Pufferzeiten

Die Bestimmung der Pufferzeiten hat den Zweck festzustellen, in welchem Bereich nicht-kritische Vorgänge verschoben werden können. Die bereits in Kapitel 1.2.5.2 angeführte **gesamte Pufferzeit** eines Vorgangs legt fest, wie weit die Aktivität verschoben werden kann, ohne die gesamte Projektdauer zu verlängern. Dabei ist darauf zu achten, dass nicht alle Pufferzeiten der Vorgänge zu einer gesamten Projektpufferzeit addiert werden können, da das Aufbrauchen einer Pufferzeit, die Pufferzeit einer anderen Aktivität verringert. Für eine gezieltere Verschiebung mit der Kenntnis, welche Auswirkung dies auf den Projektplan hat, gibt es weitere Pufferzeiten. Um festzustellen, wie weit ein Vorgang verschoben werden kann, aber seine Nachfolger ihre FEZ Werte nicht verlieren, d.h. von ihren Pufferzeiten keinen Gebrauch machen, wird die **freie Pufferzeit** (FP) errechnet. Damit festgestellt werden kann, welchen Verschiebungsbereich ein Vorgang besitzt, wenn alle Vorgänger so spät als möglich beendet werden und alle Nachfolger frühest möglich beginnen, kann die **unabhängige Pufferzeit** (UP) bestimmt werden. Der **freie Rückwärts-Puffer** (FRP) gibt

[40] (Schwarze, 2010 S. 150f)

den Abstand zu dem spät möglichst beendeten Vorgänger und der Aktivität selbst an, wenn diese ebenfalls spät möglichst beendet wurde. [41]

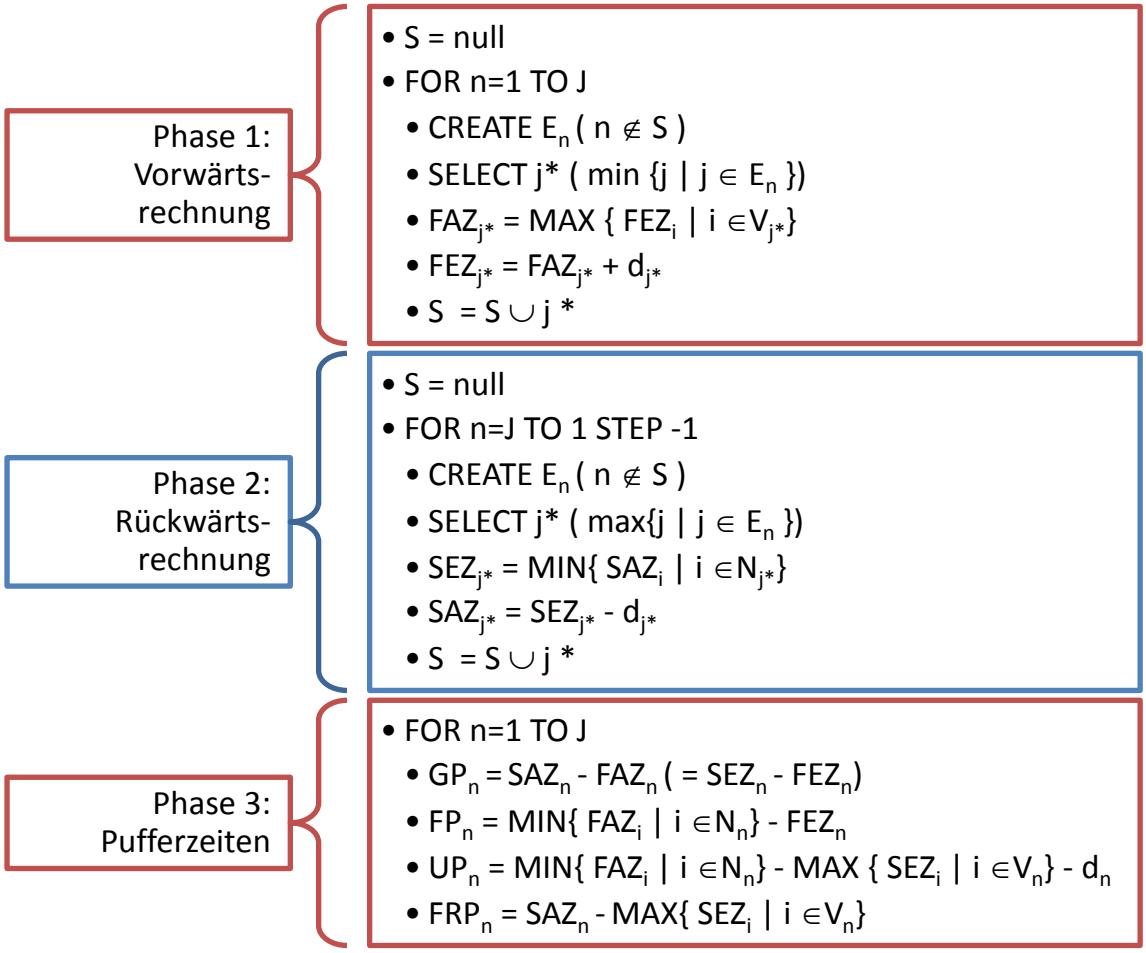

Abbildung 18: Ablaufplan der Zeitplanung

1.2.5.4. Netzpläne mit Zeitabständen

Da es auch möglich ist, Vorgänge negativ oder positiv zeitverzögert Z zu starten[42], so ist dies bei der Berechnung der Start-, End- und Pufferzeiten zu berücksichtigen. Diese Änderungen sind in Tabelle 3 zusammengefasst.[43]

[41] (Schwarze, 2010 S. 153ff)
[42] Siehe Kapitel 1.2.4.2.4
[43] (Schwarze, 2010 S. 152 und 160)

Zeit	Neue Berechnung	Phase
$\mathbf{FAZ_{j*}}$	MAX { FEZ_i **+** $\mathbf{Z_i}$ | i ∈V_{j*}}	1
$\mathbf{SEZ_{j*}}$	MIN{ SAZ_i **-** $\mathbf{Z_i}$ | i ∈N_{j*}}	2
$\mathbf{FP_n}$	MIN{ FAZ_i **-** $\mathbf{Z_i}$ | i ∈N_n} - FEZ_n	3
$\mathbf{UP_n}$	MIN{ FAZ_i **-** $\mathbf{Z_i}$ | i ∈N_n} - MAX { SEZ_i **+** $\mathbf{Z_i}$ | i ∈V_n} - d_n	3
$\mathbf{FRP_n}$	SAZ_n - MAX{ SEZ_i **+** $\mathbf{Z_i}$ | i ∈V_n}	3

Tabelle 3: Start-, End- und Pufferzeiten bei Zeitplänen mit Zeitabständen (Schwarze, 2010 S. 152 und 160)

1.2.5.5. Stochastische Zeitplanung PERT[44]

Ist es nicht möglich, einen eindeutigen Wert für die Dauer eines Vorganges festzulegen, so bietet die PERT[45] Methode die Dreizeitenschätzung. Durch die Festlegung der wahrscheinlichsten (HD), pessimistischen (PD) und optimistischen (OD) Dauer kann ein Erwartungswert MD, für die spätere Verwendung in der Zeitplanung, berechnet werden.

$$MD = \frac{1}{3}\frac{OD + HD}{2} + \frac{2}{3}PD = \frac{OD + 4HD + PD}{6}$$

Formel 1: PERT Erwartungswert für die Ausführungsdauer (Schwarze, 2010 S. 168)

Die Annahme, dass den Zeitwerten eine β-Verteilung, eine Form der Wahrscheinlichkeitsverteilung, zugrunde liegt, führt zu der Berechnungsmethode in Formel 1. Für die Berechnung der Unsicherheit der Dauer der Vorgänge werden deren Varianzen VD festgestellt

$$VD = \left(\frac{PD - OD}{6}\right)^2$$

Formel 2: PERT Varianz der Ausführungsdauer (Schwarze 2010, S.168)

Mit der weiteren Annahme, dass es ein angestrebtes Projektende APE und ein erwartetes Projektende EPE, welches sich durch die Zeitplanung mit den Erwartungswerten MD errechnet, gibt und dass für die Summe der Erwartungswerte MD und deren Varianzen eine Normalverteilung vorliegt, so kann man für Einhaltung des Gesamtprojektendes die Standabweichung Z feststellen aus der dann die Wahrscheinlichkeit bestimmt werden kann:

[44] (Schwarze, 2010 S. 168ff)
[45] Program Evaluation and Review Technique

$$Z = \frac{APE - EPE}{\sqrt{\sum VD}}$$

Formel 3: Wahrscheinlichkeit der Projektdauer (Schwarze 2010, S.168) (Heizer, et al., 1990 S. 708)

1.2.6. Kostenplanung

Eine Aufgabe der Projektplanung ist die Analyse und Kontrolle der Projektkosten, deren Ziel es ist, Mehrkosten frühzeitig aufzudecken, Kosten für das laufende Projekt stetig zu kontrollieren und die Wirtschaftlichkeit des Projektes sicher zu stellen. Kosten treten im laufenden Betrieb (Personal, Wirtschaftsgüter), durch Verbrauch von Ressourcen, in Interaktion mit Dritten und aus rechtlichen Gründen (Steuern) auf und werden durch eine Menge und einem Preis pro Einheit bestimmt. In der Zeitplanung des Kapitel 1.2.5 wurde die Dauer eines Vorgangs als nicht veränderbar angenommen. Diese Dauer wird in Folge Normaldauer DN genannt, der die Kosten KN zugeordnet sind. Eine Veränderung im Produktionsprozess, die zeitlich, quantitativ oder qualitativ bedingt sein kann, führt zu einer Beeinflussung der Vorgangsdauer. Der Bereich der Veränderung der Vorgangsdauer ist mit einer minimalen (DC) und maximalen Dauer (DM) begrenzt. Eine einfache Berechnung der **Beschleunigungskosten b** kann durch eine lineare Approximation der minimalen Vorgangskostenkurven (Abbildung 19) erreicht werden. [46]

$$b = \frac{KC - KN}{DN - DC}$$

Formel 4: Beschleunigungskosten eines Vorgangs (Schwarze, 2010 S. 219)

Eine weitere Annäherung an die Realität kann durch die stückweise lineare Approximation erreicht werden (Abbildung 19). In einem nicht kapazitätsbeschränkten Netzplan kann man in einem iterativen Prozess die Projekt-Vorgangsdauer einfach reduzieren, indem aus der Menge VK_n der kritischen Vorgänge der Kandidat mit den geringsten Beschleunigungskosten gewählt wird, dessen Dauer um eine Zeiteinheit reduziert, sofern die minimale Dauer nicht erreicht ist, und nach jedem Schritt die Liste VK_n aktualisiert. Dies wird solange wiederholt, bis die gewünschte Dauer erreicht ist oder keine kritischen Wege mehr beschleunigt werden können.[47]

[46] (Schwarze, 2010 S. 208ff)
[47] (Zimmermann, 2008 S. 385)

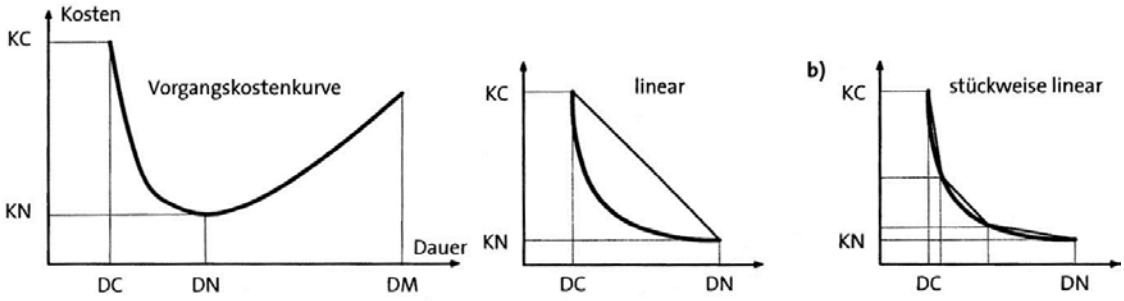

Abbildung 19: Kostenfunktion (Schwarze, 2010 S. 218)

Die zuerst erwähnten Kostenfunktionen, können mathematisch definiert werden und für die Kostenoptimierung mit folgendem Modell beschrieben werden.

$$KG = \sum_{j=1}^{n} KN_j + (b_j\, d_j\,)$$

$$KD = \sum_{j=1}^{n} KN_j + b_j(DN_j - d_j) = KG - \sum_{j=1}^{n} b_j d_j$$

Formel 5: Konstante Normalkosten (KG) und direkte Projektkosten (KD) (Al-Momani, 2000 S. 80f)

n	Anzahl Aktivitäten
j = {1,.., n}	Aktivität
d_j	Dauer der Aktivität j
a_j	Anfangszeitpunkt der Aktivität j
P_j = {1,.., h}	Menge an Vorgängern einer Aktivität j
b_j	Zusatzkosten bei Beschleunigung der Aktivität j
e	geplantes Projektende

$$x_{jt} \begin{cases} 1\,, Aktivität\ j\ wird\ zum\ Zeitpunkt\ t\ beendet \\ 0\,, \quad sonst \end{cases}$$

$$\sum_{j=1}^{n} b_j\, d_j \rightarrow min$$

Formel 6: Zielfunktion des beschleunigten Netzplans (Al-Momani, 2000 S. 80f)

$$\sum_{t=a_h+DC_j}^{a_h+DN_j} x_{jt} = 1 \qquad j = 1, \dots, n\,, \;\; DC_j \leq d_j \leq NC_j \qquad (1)$$

$$\sum_{t=a_h+DC_h}^{a_h+DN_h} t\; x_{ht} \leq \sum_{t=a_j+DC_j}^{a_j+DN_j} (t - d_j) * x_{jt} \qquad j = 1, \dots, n\;, h \in P_j \qquad (2)$$

$$b_n + d_{jn} \leq e \qquad (3)$$

Formel 7: Nebenbedingung des beschleunigten Netzplans (Al-Momani, 2000 S. 80f)

Es werden dadurch die Zusatzkosten pro Zeiteinheit der beschleunigten Vorgänge optimiert, unter der Berücksichtigung, dass die Dauer einer eingeplanten Aktivität die kritische Dauer DC nicht unterschreitet (1), alle Vorgängerbeziehungen berücksichtigt sind (2) und dass das Projektende eingehalten wird (3).

1.2.7. Kapazitätsplanung

Die bisherige Betrachtung der Netzplantechnik in dieser Arbeit geht davon aus, dass ein Vorgang, unter Berücksichtigung seiner Anfangs-, End und Pufferzeiten, frei planbar ist bzw. durch eine Erhöhung der Projektkosten in einer kürzeren Zeit ausgeführt werden kann. Durch die endliche Verfügbarkeit von Projektressourcen kann es notwendig sein, dass ein Vorgang zu einem späteren Zeitpunkt ausgeführt werden muss, da er auf das Freiwerden einer Ressource angewiesen ist. Dies hat zur Folge, dass insbesondere durch die kritischen Vorgänge, die Projektdauer verlängert wird.[48]

Für eine graphische Darstellung des Kapazitätsproblems, können die Aktivitäten mit ihren Ausprägungen Startzeit, Dauer und Ressourcenbelegung in ein Kapazitätsbelastungsdiagramm übergeführt werden. Hierbei ist für jede Ressource R ein eigenes Diagramm zu verwenden. Ausgehend von Abbildung 25 (Kapitel 2.5.1) ist die Verletzung der Ressourcenbedingung in einem Kapazitätsbelastungsdiagramm erkennbar, wenn bei der Festlegung der Startzeiten die Kapazitäten der Ressource R_1 nicht berücksichtigt werden. In Abbildung 20 ist auf der linken Seite ein nicht zulässiger Projektplan zu sehen und rechts, durch eine simple Blockverschiebung von Aktivität 4 und 6 um eine Einheit, die optimale Lösung.

[48] (Zimmermann, 2008 S. 375)

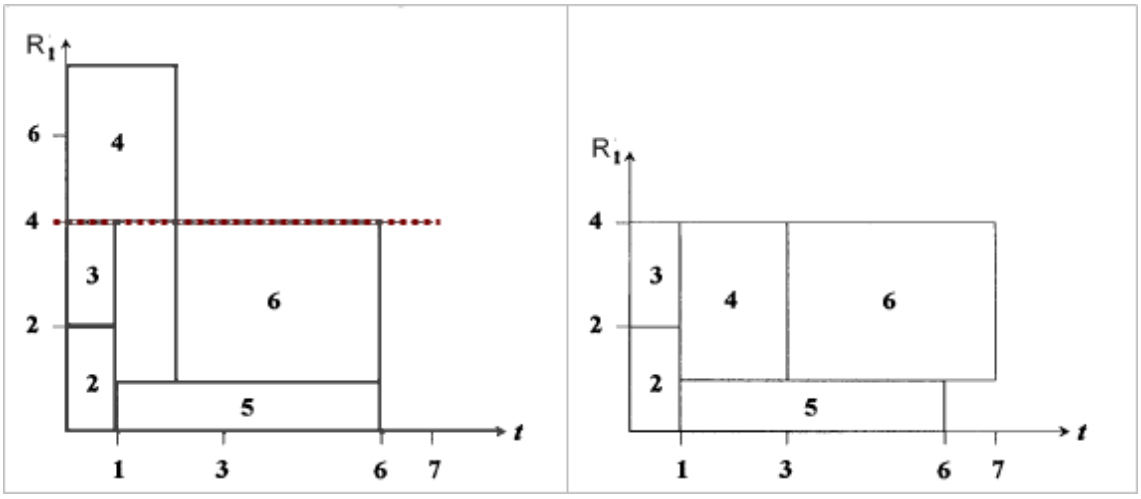

Abbildung 20: Kapazitätsbelastungsdiagramme (Kolisch, 1995 S. 78, rechts)

Für komplexere Probleme mit vielen Aktivitäten und Ressourcen, ist die Lösung in einem Kapazitätsbelastungsdiagramm nicht mehr so einfach erkennbar und durchführbar wie in Abbildung 20.

Eine detaillierte Beschreibung des kapazitätsbeschränkten Netzplans und die Möglichkeit optimierte Lösungen zu finden ist in Kapitel 2 beschrieben.

2. Lösungsmethoden für Netzpläne mit beschränkten Ressourcen

In der Literatur des ressourcenbeschränkten Netzplan Problems (infolge RCPSP[49] genannt) wird zwischen Single Mode RCPSP (SMRCPSP) und Multi Mode RCPSP (MMRCPSP) unterschieden, die von einem Multi Project RCPSP abgeleitet werden können. Das MPRCPSP ist durch die gleichzeitige Berechnung mehrerer Projekte gekennzeichnet, wobei eine Aktivität jeweils Mitglied eines Modi M_j ist und mit allen anderen Aktivitäten zusammen ein sogenanntes „Superprojekt“ (oder Gesamtprojekt) bildet. Hinzu kommt die Erweiterung, dass es zusätzlich zu den erneuerbaren Ressourcen nicht erneuerbare gibt oder eine Kombination aus beiden. Solche Ressourcen werden als doppelt beschränkte Ressourcen bezeichnet. [50] Diese Arbeit beschränkt sich auf das SMRCPSP und verwendet in weiterer Folge für SMRCPSP die vereinfachende Bezeichnung RCPSP. Weitere, generalisierte, Ausprägungen des RCPSP können in Herroelen, de Reyck und Demeulemeester (1998) nachgelesen werden. Eine Übersicht der Erweiterungen des RCPSP bietet Hartmann und Briskorn (2010).

2.1. Problembeschreibung

Das klassische RCPSP kann folgendermaßen definiert werden: Ein Projekt beinhaltet n Aktivitäten j = {1,…, n}, welche ohne Unterbrechungen eingeplant werden sollen. Die Aktivität j_1 und j_n legen den Beginn bzw. das Ende des Projektes (Quelle bzw. Senke[51]) fest. Jede Aktivität j besitzt einer Dauer d_j. Die Quelle j_1 und die Senke j_n besitzen jeweils keine Dauer ($d_1 = d_n = 0$). Ein Netzplan mit beschränkten Ressourcen ist dadurch charakterisiert, dass jede Aktivität j einen Ressourcenverbrauch r_{jk} besitzt, wobei K erneuerbare Ressourcentypen k = {1,…, K} zur Verfügung stehen. Die Ressource R_k ist zu jedem Zeitpunkt t verfügbar und konstant. Der Ressourcenverbrauch der Quelle j_1 und der Senke j_n ist wiederum nicht vorhanden ($r_{1k} = r_{nk} = 0$). Alle Parameter sind ganzzahlig und nicht negativ. Aktivitäten sind durch die Definition von Nachfolgern S_j (bzw. daraus errechenbaren Vorgängern P_j) relationiert.[52]

[49] RCPSP: Resource constrained project scheduling problem [Engl.]
[50] (Kolisch, et al., 1992 S. 1f)
[51] Siehe Kapitel 1.2.3
[52] (Valls, et al., 2008 S. 495f) (Hartmann, 1998 S. 2f)

2.2. Zielsetzung

Die Erstellung eines Projektplans[53] S, welcher n Startzeiten s = {1,…,n} für jede Aktivität enthält, unter der Berücksichtigung aller Ressourcenbeschränkungen und Vorgängerrelationen, mit einer minimalen Gesamtprojektdauer T(S) = s_n (s_1=0), ist die Zielsetzung des RCPSP.[54]

Die mathematische Formulierung des Problems lautet wie folgt[55]:

n	Anzahl Aktivitäten
j = {1,.., n}	Aktivität
d_j	Dauer der Aktivität j
FEZ_j	Frühester Endzeitpunkt der Aktivität j
SEZ_j	Spätester Endzeitpunkt der Aktivität
P_j = {1,.., h}	Menge an Vorgängern einer Aktivität j
K	Anzahl Ressourcen
k = {1,.., K}	Ressource
r_{jk}	Kapazitätsbedarf der Aktivität j in der Ressource k
R_k	Verfügbare Kapazität der Ressource k

$$x_{jt} \begin{cases} 1\,, Aktivität\ j\ wird\ zum\ Zeitpunkt\ t\ beendet \\ 0\,, \quad sonst \end{cases}$$

$$\sum_{t=FEZ_j}^{SEZ_j} t\, x_{jt} \rightarrow min$$

Formel 8: Zielfunktion des SMRCPSP (Kolisch, et al., 1992 S. 3ff)

$$\sum_{t=FEZ_j}^{SEZ_j} x_{jt} = 1 \qquad j = 1, \dots, n \qquad (1)$$

$$\sum_{t=FEZ_h}^{SEZ_h} t\, x_{ht} \leq \sum_{t=FEZ_j}^{SEZ_j} \left(t - d_j\right) x_{jt} \qquad j = 1, \dots, n\ , h\ \in P_j \qquad (2)$$

$$\sum_{j=1}^{n} r_{jk} \sum_{q=t}^{t+d_j-1} x_{jq} \leq R_k \qquad k = 1, \dots, K\ , t = 1, .., T \qquad (3)$$

Formel 9: Nebenbedingung des SMRCPSP (Kolisch, et al., 1992 S. 3ff)

[53] S: Schedule [engl. für Projektplan]
[54] (Valls, et al., 2008 S. 496)
[55] Definition nach (Kolisch, et al., 1992 S. 3ff) reduziert auf das SMRCPSP

Die erste Nebenbedingung legt fest, dass jeder Zeitpunkt innerhalb seines frühest möglichen und spätest möglichen Zeitpunkt beendet wird (1). Weiters müssen die Vorgängerbeziehungen eingehalten werden. Hier werden die Endzeitpunkte der Vorgänger dem Startzeitpunkt der Aktivität gegenübergestellt und somit sichergestellt, dass keine Aktivität vor dem Endzeitpunkt eines Vorgängers beginnen kann (2). Zuletzt wird geprüft, ob der Ressourcenbedarf der Aktivitäten die verfügbaren Ressourcen nicht übersteigt. Eine allgemeine Darstellung des Problems bietet das MPRCPSP. Das SMRCPSP ist eine Vereinfachung des MPRCPSP durch die Festlegung von P=1, M_j=1, j=1,…,J , N=D=0, $p_{p=}0$ und $\delta_p = \overline{T}$, wobei N die nicht erneuerbaren Ressourcen und D die doppelt belegten bezeichnen. Das MMRCPSP unterscheidet sich von dem MPRCPSP durch P=1, $p_{p=}0$ und $\delta_p = \overline{T}$. [56]

p	Anzahl Projekte
δ_p (p_p)	Fälligkeitstermin (Freigabetermin) des Projektes p
m	Anzahl Modi
M_j	Modus der Aktivität j
EJ_p (LJ_p)	Erste (Letzte) Aktivität des Projekte p
d_{jm}	Dauer der Aktivität j im Modus m
$\overline{T}$	Obere Grenze für die Projekdauer
r^p_{jmk}	Kapazitätsbedarf der Aktivität j in der erneuerbaren Ressource k im Modus m
R^p_k	Verfügbare Kapazität der erneuerbaren Ressource k
r^v_{jmk}	Kapazitätsbedarf der Aktivität j in der nicht ern. Ressource k im Modus m
R^v_k	Verfügbare Kapazität der nicht erneuerbaren Ressource k
E,N,D	Erneuerbare, nicht erneuerbare und doppelt belegte Ressourcen

Dadurch wird auch die Einplanvariable um den Modus m erweitert:

$$x_{jmt} \begin{cases} 1\,, Aktivität\ j\ wird\ in\ Modus\ m\ ausgeführt\ und\ um\ Zeitpunkt\ t\ beendet \\ 0\,, \quad sonst \end{cases}$$

Die Zielfunktion und die bisherigen Nebenbedingungen werden analog angepasst:

$$\sum_{m=1}^{M_j} \sum_{t=FEZ_j}^{SEZ_j} t\ x_{jmt} \rightarrow min$$

Formel 10: Zielfunktion des MMRCPSP (Kolisch, et al., 1992 S. 3ff)

[56] (Kolisch, et al., 1992 S. 3f)

$$\sum_{m=1}^{M_j} \sum_{t=FEZ_j}^{SEZ_j} x_{jmt} = 1 \qquad j = 1, \dots, n \qquad (1)$$

$$\sum_{m=1}^{M_j} \sum_{t=FEZ_h}^{SEZ_h} t\, x_{hmt} \leq \sum_{m=1}^{M_j} \sum_{t=FEZ_j}^{SEZ_j} (t - d_{jm})\; x_{jmt} \qquad j = 1, \dots, n\ , h \in P_j \qquad (2)$$

$$\sum_{j=1}^{n} \sum_{m=1}^{M_j} r_{jmk}^{p} \sum_{q=t}^{t+d_j-1} x_{jmq} \leq R_k^p \qquad k \in E \cup D\ , t = 1, .., T \qquad (3)$$

Formel 11: Nebenbedingung des MMRCPSP (Kolisch, et al., 1992 S. 3ff)

Durch die nicht erneuerbare Ressource wird eine neue Nebenbedingung (4) eingeführt. Der Fälligkeitstermin der Projekte und der jeweilige Freigabetermin werden mit Nebenbedingung (5) und (6) dargestellt.[57]

$$\sum_{j=1}^{n} \sum_{m=1}^{M_j} r_{jmk}^{v} \sum_{t=FEZj}^{SEZ_j} x_{jmt} \leq R_k^v \qquad k \in N \cup D\ , t = 1, .., T \qquad (4)$$

$$\sum_{m=1}^{M_j} \sum_{t=FEZ_j}^{SEZ_j} t - d_{jm} \geq p_p \qquad p = 1, \dots, P, \quad j = EJ_p, \dots, LJ_p \qquad (5)$$

$$\sum_{m=1}^{M_j} \sum_{t=FEZ_j}^{SEZ_j} t\; x_{jmt} \leq d_p \qquad p = 1, \dots, P, \quad j = EJ_p, \dots, LJ_p \qquad (6)$$

Formel 12: Erweiterte Nebenbedingung des MMRCPSP (Kolisch, et al., 1992 S. 3ff)

Das RCPSP ist ein NP[58]-vollständiges Problem.[59] Selbst Netzpläne von geringem Umfang, die durch schwierige Instanz-Parameter [60]- Kombinationen gekennzeichnet sind, benötigen sehr großen Rechenaufwand um exakt gelöst zu werden. Bevor diese Verfahren und alternativ dazu Heuristiken und Metaheuristiken vorgestellt werden, die versuchen in kurzer Zeit eine Lösung nahe dem Optimum zu finden, klassifiziert das nächste Kapitel die Arten der Projektpläne in Bezug auf die Anordnung der geplanten Aktivitäten.

[57] (Kolisch, et al., 1992 S. 3f)
[58] NP: nondeterministic polynomial [Engl.]
[59] (Blazewicz, et al., 1983 S. 555ff)
[60] Siehe Kapitel 2.9.1

2.3. Klassifizierung von Projektplänen

Die Anordnung der Aktivitäten eines Projektplans, deren Beziehung untereinander und die Auswirkungen auf die Ressourcenbedingungen determinieren die Klassifizierung eines Projektplans.[61]

2.3.1. Zulässiger Projektplan

Ein **zulässiger Projektplan** ist dadurch gekennzeichnet, dass alle Vorgängerbeziehungen und Ressourcenbedingungen der Aktivitäten des Projektplans erfüllt sind. Nebenbedingung (2) und (3) aus dem Kapitel 2.1 der Formel 9 repräsentieren diese Bedingungen.[62]

2.3.2. Aktiver Projektplan

In einem zulässigen Projektplan, kann es möglich sein, einzelne Aktivitäten nach links zu verschieben, d.h. deren Startzeitpunkte zu reduzieren. Die Linksverschiebung kann lokal oder global erfolgen. Die lokale Linksverschiebung der Aktivität, bei gleichzeitiger Fixierung aller anderen Aktivitäten erfolgt periodisch, wobei diese Verschiebung mehrfach hintereinander durchgeführt werden kann. Hierbei gilt zu beachten, dass die, durch die Verschiebung entstandenen Projektpläne zulässig sein müssen. Eine lokale, oder auch ein-periodische, Verschiebung ist dann nicht mehr möglich, wenn die Ressourcenbedingung oder Vorgängerbeziehung verletzt wird. Sind keine lokalen Verschiebungen innerhalb des Projektplans mehr möglich, so spricht man von einem **semi-aktiven Projektplan**. Ein semi-aktiver Projektplan stellt noch keinen aktiven Projektplan dar. Es ist möglich, in einem semi-aktiven Projektplan eine Aktivität global zu verschieben (mehr-periodische Verschiebung), wobei die theoretischen Zwischenprojektpläne nicht zulässig wären (daher ist auch keine lokale Verschiebung mehr möglich). Sind weder lokale noch globale Verschiebungen mehr möglich, so erhält man einen **aktiven Projektplan**, d.h. es ist unter Berücksichtigung der Ressourcenbedingungen und der Vorgängerbeziehungen nicht möglich, eine Aktivität früher zu beginnen.[63]

---

[61] (Kolisch, 1995 S. 38)
[62] (Kolisch, 1995 S. 40)
[63] (Kolisch, 1995 S. 40ff),

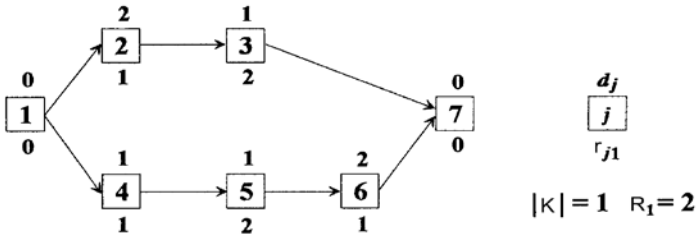

Abbildung 21: Beispielprojekt 1 (Kolisch 1995, S.43)

Abbildung 23 zeigt einen zulässigen Projektplan für das Beispielprojekt aus Abbildung 21, welcher als Ausgangsbasis für die lokale und globale Linksverschiebung dient.

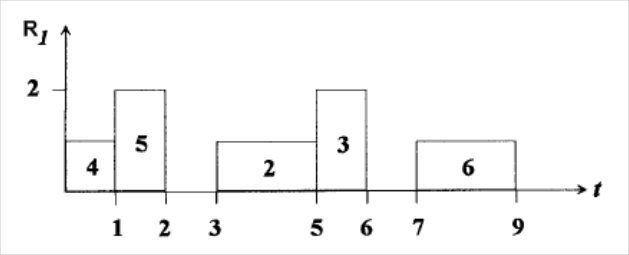

Abbildung 22: Zulässiger Projektplan (Kolisch, 1995 S. 43)

Durch die lokale Linksverschiebung der Aktivitäten {2,3,6} kann ein semi-aktiver Projektplan erreicht werden:

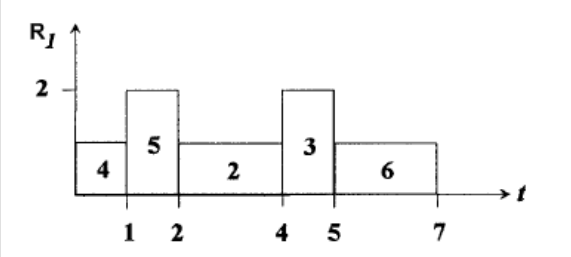

Abbildung 23: Semi-aktiver Projektplan (Kolisch, 1995 S. 44)

Eine weitere Optimierung des Projektplans ist nur noch durch eine globale Linksverschiebung möglich, die mittels Aktivität 6 (3 Perioden) durchgeführt werden kann. In dieser Abbildung 23 kann man gut erkennen, dass die theoretischen ein-periodischen Linksverschiebungen der Aktivität 6 zu einer unzulässigen Lösung führen würden. Hierbei würde die Ressourcenbedingung (R_1) verletzt werden. Die globale Linksverschiebung um drei Perioden verletzt die Zulässigkeit der Lösung hingegen nicht. Ein aktiver Netzplan repräsentiert nicht zwangsläufig eine optimale Lösung,

jedoch kann die Optimalität der Lösung aufgrund der Einfachheit des Beispielprojekts leicht erkannt werden.[64]

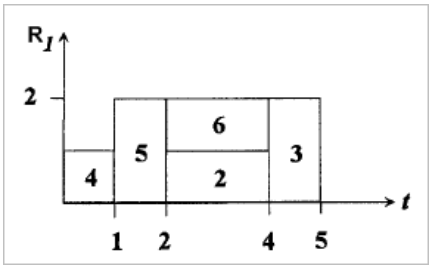

Abbildung 24: Aktiver Projektplan (Kolisch, 1995 S. 44)

[64] (Kolisch, 1995 S. 43ff)

2.4. Exakte Verfahren

Zur Lösung des RCPSP werden vor allem Branch-and-Bound-basierende Verfahren verwendet. Diese Methode verwendet einen Suchbaum, indem entschieden wird, ob eine Aktivität sofort eingeplant wird oder verspätet gestartet wird. Maßgeblich für die Rechenleistung eines Algorithmus für exakte Lösungen, ist die Möglichkeit, den Ausbreitungsraum der Suche begrenzen zu können. Zumeist wird ein zeit-orientiertes Verzweigungsschema gewählt, bei dem die möglichen Startzeitpunkte der zu planenden Aktivität die Verzweigung bestimmen und die abhängigen, darunterliegenden Knoten durch die Aktivität bestimmt werden. Eine weitere Möglichkeit der Verzweigung bieten binäre Suchschemata, die zum Beispiel das Intervall einer Aktivität in gleiche Bereiche aufteilt. Generell wird für die Lösung des RCPSP eine Suchstrategie angestrebt, die alle möglichen Lösungen erreichen kann aber auch den Suchbereichs intelligent reduziert. Die Constrainedprogrammierung[65] ist der Ursprung der Technik der beschränkten Ausbreitung, die durch die wiederholte Untersuchung der beschreibenden Variablen, Domänen und Restriktionen der Probleminstanz, die vergebenen Restriktionen implizit bewertet. Ein Kennzeichen eines gültigen Projektplans stellen die Vorgängerrestriktionen dar, die über die Startzeiten geprüft werden können. Die beschränkte Ausbreitung untersucht und löst diese möglichen Inkonsistenzen. Wichtigster Teil der Suchraumreduktion sind Konsistenzregeln. Es ist weiters notwendig, in dem Zeitintervall des Suchbereichs eine Kapazitätskonsistenz aller Aktivitäten zu erreichen, wobei eine Aktivität den Bereich komplett überspannen kann, ein geringer oder großer Teil davon ist, gänzlich darin vorkommt oder gar nicht vorhanden ist. Der Konsistenztest beinhaltet auch eine Paarprüfung, die auf einander ausschließende Aktivitäten [66] prüft und ein symmetrischer Test prüft Vorgängerbeziehungen.[67]Für weitere Informationen und die beispielhafte Lösung des RCPSP mittels Branch and Bound wird auf Dorndorf, Pesch und Phan-Huy (2000) und Herroelen, de Reyck und Demeulemeester (1998) verwiesen.

[65] **Constrained Programming** [Engl.]: „The idea of constraint programming is to solve problems by stating constraints (requirements)about the problem area and, consequently, finding solution satisfying all the constraints.“ (Barták, 1999)

[66] Die Summe des Ressourcenverbrauchs beider Aktivitäten übersteigt die verfügbare erneuerbare Gesamtressource

[67] (Dorndorf, et al., 2000 S. 421ff)

2.5. Prioritätsheuristiken

2.5.1. Beispielprojekt

Folgendes Beispielprojekt wird für die Berechnungserklärungen in diesem Kapitel verwendet. Es handelt sich hierbei um ein SMRCPSP mit einer erneuerbaren Ressource R_1 und 5 Aktivitäten, die von einer Quelle und Senke umgeben sind:

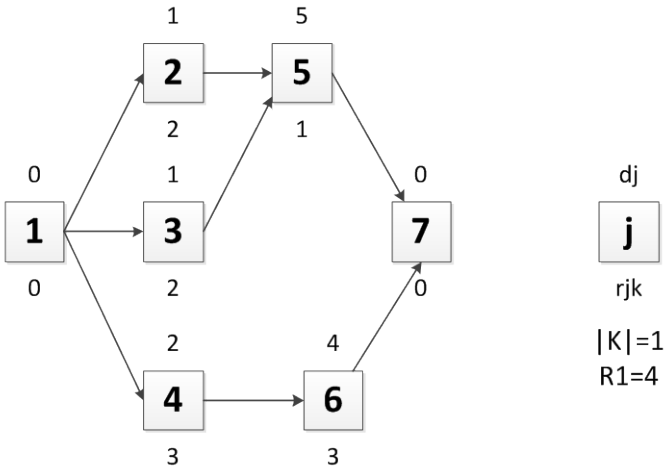

Abbildung 25: Beispielprojekt 2 (Kolisch, 1995 S. 77)

2.5.2. Generierung eines Projektplans basierend auf Prioritätswerten

In den meisten Heuristiken wird für die Lösung des RCPSP die serielle oder parallele Methode für die Generierung eines Projektplans (SGS)[68] verwendet. Dieser wichtige Baustein wird verwendet, da sie (a) einfach in bestehende Lösungen integriert werden kann, (b) eine performante Leistung bietet und (c) sie sich bei der Lösung der Zielfunktion im oberen Bereich der bestehenden Methoden befindet.[69]

Der Projektplan wird Schritt für Schritt generiert, wobei die jeweils zur Auswahl stehenden Aktivitäten durch Prioritätsregeln gewählt werden. Bei dieser Vorgangsweise werden drei Aktivitätslisten verwendet. Aus der Liste der noch nicht geplanten Aktivitäten, welche zum Anfangszeitpunkt alle Aktivitäten enthält, werden für die Anwendung der Prioritätsregel(n) Aktivitäten, dessen Vorgänger bereits geplant sind, in eine Entscheidungsliste geladen. Nach der Auswahl der Aktivität, wird diese in

[68] SGS: Schedule generation scheme [Engl.]
[69] (Kolisch, 1996b S. 322)

die Liste der geplanten Aktivitäten, welche zum Anfangszeitpunk keine Aktivitäten enthält, verschoben. [70]

2.5.2.1. Prioritätsregeln

Zu dem Zeitpunkt der Aktivitätsauswahl wird jeder Aktivität der Entscheidungsliste (alle ungeplanten Aktivitäten deren Vorgänger bereits eingeplant sind) ein Prioritätswert v(j) zugewiesen und festgelegt, ob der minimale oder maximale Prioritätswert aller Aktivitäten für die Entscheidung herangezogen werden soll. [71]

Prioritätsregeln können nach folgenden Bereichen klassifiziert werden, welche jedoch keine Aussage über die Qualität der Regel selbst treffen[72]:

- Klassifikation nach der Information der Regel: Regeln basierend auf Netzwerk, Dauer, Ressourcen
- Klassifikation nach deren Dynamik: Statische (Regel bleiben gleich) und dynamische (Regel ändert sich im Laufe des Generierens) Regeln
- Klassifikation nach dem Umfang der Information: Lokale Regeln (nur Informationen der Aktivität selbst werden in Betracht gezogen) und globale Regeln.
- Klassifikation nach der Verwendung von unteren Schranken: Regeln unterscheiden sich durch die Verwendung von unteren Schranken.

Tabelle 4 zeigt eine Übersicht der bekanntesten Prioritätsregeln die zur Lösung von RCPSP verwendet werden. RSM und WCS sind Prioritätsregeln, die mit Aktivitätspaaren, die aus der Entscheidungsliste generiert werden, agieren. WCS berechnet hierbei unter Berücksichtigung der Ressourcenbeschränkungen den frühest möglichen Start einer Aktivität j, wenn die Aktivität i zum Zeitpunkt t_g eingeplant wird.[73] Die RSM Regel nimmt an, dass eine Aktivität aus dem Aktivitätspaar immer nach dem Ende der anderen Aktivität eingeplant werden muss.[74] Ein detailliertes Beispiel ist hierzu in Kolisch (1996a) zu finden.

[70] (Kolisch, et al., 1999 S. 150)
[71] (Kolisch, et al., 1999 S. 154)
[72] (Kolisch, 1996a S. 182)
[73] (Kolisch, et al., 1999 S. 155f)
[74] (Kolisch, 1996a S. 182)

Tabelle 4: Übersicht Prioritätsregeln (Kolisch, et al., 1999 S. 155)

Regel	Name	Autor	v(j)
GRPW	Greates Rank Positional Weight	Alvarez-Valés, Tamarit (1989)	$d_j+\sum_{i\in Sj} di$
LFT	Latest Finish Time	Davis, Patterson (1975)	SEZ_j
LST	Latest Start Time	Kolisch (1995)	$SEZ_j - d_j$ (=SAZ_j)
MSLK	Minimum Slack	Davis, Patterson (1975)	$SEZ_j - SAZ_j$)
MTS	Most Total Successors	Alvarez-Valés, Tamarit (1989)	Alle $\|S_j\|$
RSM	Resource Scheduling Method	Shaffer et al. (1965)	$Max_{(i,j)AP}$[75] {0, $t_g+d_j-SAZ_i$}[76]
SPT	Shortest Processing Time	Alvarez-Valés, Tamarit (1989)	d_j
WCS	Worst Case Slack	Kolisch (1996b)	$SEZ_j-d_j- Max_{(i,j)AP}$ {E(i,j)}

MTS verwendet die Anzahl aller (nicht nur direkten) Nachfolger der Aktivität als Prioritätsregel. LFT, LST und MSLK vergleichen die spätest möglichen Einplanzeiten für die Entscheidungsfindung. Die Dauer der Aktivität findet in GRPW und SPT Anwendung.[77]

Eine detaillierte Beschreibung der Prioritätsregeln und entsprechende Klassifizierung findet sich in Kolisch (1995).

Abbildung 26 zeigt eine mögliche Ausprägung der Prioritätswerte, wobei hier die statische Prioritätsregel LFT[78] verwendet worden ist.

[75] Aktivitätspaare
[76] Der minimalste Wert wird gewählt
[77] (Kolisch, et al., 1999 S. 154)
[78] Späteste Endzeitpunkte ohne die Berücksichtigung von Ressourcenbeschränkungen

j	1	2	3	4	5	6	7
$v(j)$	0	1	1	2	6	6	6

Abbildung 26: Prioritätswerte für Abbildung 25 (Kolisch, 1995 S. 77)

2.5.2.2. Serielle Methode (S-SGS) [79]

Die serielle Methode (S-SGS)[80] plant iterativ eine Aktivität nach der anderen ein (n = {1,…, J}). Dazu wird in jeder Iteration eine Entscheidungsliste aus der Liste der ungeplanten Aktivitäten U_n generiert. Die Entscheidungsliste darf nur Aktivitäten enthalten, deren Vorgänger bereits Teil der Liste der geplanten Aktivitäten S_n sind. Aus der Entscheidungsliste D_n wird mittels einer Prioritätsregel eine Aktivität gewählt und zum frühest möglichen Zeitpunkt, unter Berücksichtigung der Ressourcenbedingungen, eingeplant. Bei Gleichheit der Prioritätswerte wird die minimale Aktivitätsnummer als Entscheidungskriterium herangezogen.

Formalisiert kann die Entscheidungsliste folgendermaßen dargestellt werden:

$D_n = \{ j \mid j \notin S_n , P_j \subseteq S_n \}$

Während der Durchführung der Methode ist S_n noch ein unvollständiger Projektplan und ist zum Zeitpunkt J (welche der Anzahl Aktivitäten entspricht) vollständig.

In jeder Iteration werden die Restkapazitäten ΔR_{kt} für die möglichen Einplanzeitpunkte t der vorhandenen Ressourcen berechnet, um festzustellen, ob die gewählte Aktivität zum Zeitpunkt t eingeplant werden kann. Dies kann folgendermaßen berechnet werden:

$\Delta R_{kt} = R_k - \sum_{j \in A_t} r_{jk}$

Die allgemeine Beschreibung der seriellen Methode lautet:

[79] (Kolisch, 1996b S. 322)
[80] S-SGS: Serial Schedule Generation Scheme [Engl.]

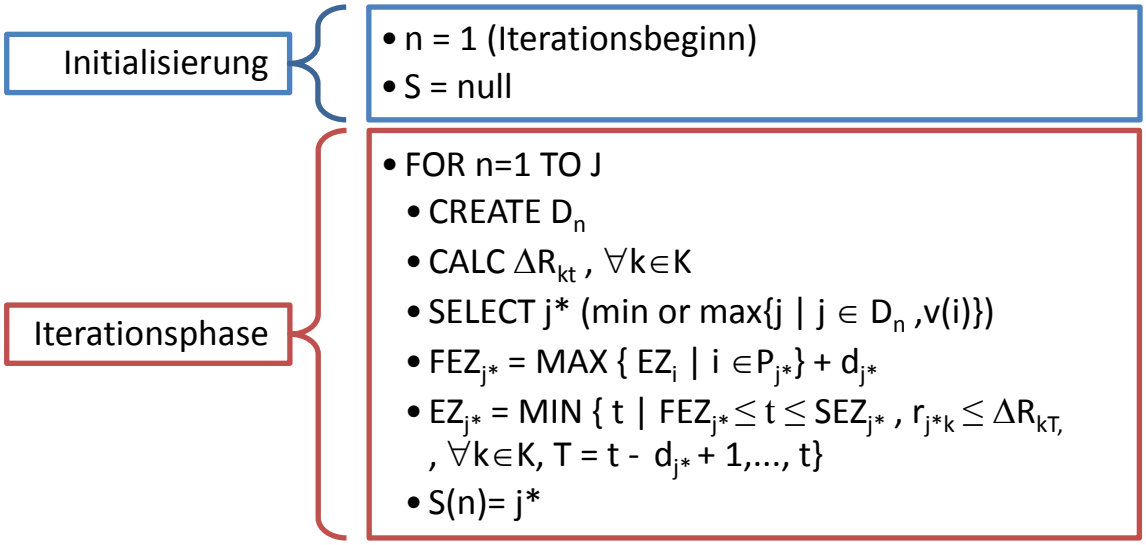

Abbildung 27: S-SGS zur Generierung eines Projektplans (Kolisch, 1996b S. 322)

Bezugnehmend auf das Beispielprojekt (Abbildung 25) und die dazu errechneten Prioritätswerte (Abbildung 26) kann das Ergebnis der Seriellen Methode in einer Tabelle dargestellt werden.

n	$\Delta R_{1t,\ t=\{1,\ldots,8\}}$	S_n	D_n	U_n	v(j)	j*	EZ_{j*}
1	{4,4,4,4,4,4,4,4}	null	{1}	{2,...,7}	{0}	1	0
2	{4,4,4,4,4,4,4,4}	{1}	{2,3,4}	{5,6,7}	{1,1,2}	2	1
3	{2,4,4,4,4,4,4,4}	{1,2}	{3,4}	{5,6,7}	{1,2}	3	1
4	{0,4,4,4,4,4,4,4}	{1,2,3}	{4,5}	{6,7}	{2,6}	4	3
5	{0,1,1,4,4,4,4,4}	{1,...,4}	{5,6}	{7}	{6,6}	5	6
6	{0,0,0,3,3,3,4,4}	{1,...,5}	{6}	{7}	{6}	6	7
7	{0,0,0,0,0,0,1,4}	{1,...,6}	{7}	null	{6}	7	7

Tabelle 5: Berechnung S-SGS (Kolisch, 1995 S. 77)

Am Beispiel der Iteration 5 kann die serielle Methode wie folgt erklärt werden: Durch den Einplanvorgang der Aktivität 4 in der Iteration 4, wird die verfügbare Restkapazität der Ressource 1 zum Zeitpunkt 2 und 3 (Startzeitpunkt der Aktivität 4 ist 2 und die Dauer d_4 lautet 2) um 3 Einheiten (= r_{41}) reduziert. Aktivität 4 wird Teil der geplanten Aktivitäten S_n und aus der Liste D_n entfernt. Durch das Einplanen von Aktivität 4 kann Aktivität 6 teil der Liste D_n werden und wird gleichzeitig aus der Liste der ungeplanten Aktivitäten U_n entfernt. Die Auswahl mithilfe der Prioritätsregel zwischen Aktivität 5 und 6 ist aufgrund der Gleichheit der Prioritätswerte nicht möglich. Daher wird die

kleinere Aktivitätsnummer als Auswahlkriterium verwendet und Aktivität 5 eingeplant. Der Endzeitpunkt dieser Aktivität beträgt 6 durch die Erfüllung der Bedingungen min (EZ_2,EZ_3) + d_5 und $r_{51} \leq \Delta R_{1t} \mid t \in \{2,3,4,5,6\}$.

2.5.2.3. Parallele Methode (P-SGS) [81]

Die parallele Methode (P-SGS[82]) besteht aus n = {1,…,J} Iterationen, in welchen zum Zeitpunkt t_n ein Set an Aktivitäten eingeplant werden soll.

Für den Einplanvorgang werden 3 Listen benötigt. Die geplanten Aktivitäten werden in zwei Gruppen unterteilt. Es gibt geplante Aktivitäten, deren Endzeitpunkte kleiner oder gleich der aktuellen Iterationsnummer sind. Diese Aktivitäten sind Teil der geplanten Liste C_n. Solange eine Aktivität eingeplant, aber zum Zeitpunkt der aktuellen Iteration nicht beendet ist, wird diese Teil der aktiven Liste A_n. In der Entscheidungsliste befinden sich alle ungeplanten Aktivitäten, deren Vorgänger bereits Teil der geplanten Aktivitäten C_n sind und deren Einplanen keine Ressourcenbedingungen verletzt.

Die Ressourcenbedingung der erneuerbaren Ressourcen lautet:

$\Delta R_k = R_k - \sum_{j \in A_n} r_{jk}$

Jede Iteration ist somit in 2 Phasen unterteilt. Phase 1 verschiebt aktive Aktivitäten in die geplanten Aktivitäten C_n, sofern deren Endzeitpunkt der Iterationsnummer (= aktueller Zeitpunkt im Projektplan) entspricht. Dadurch wird es eventuell möglich, dass neue ungeplante Aktivitäten in die Entscheidungsliste D_n aufgenommen werden können. In der 2. Phase wird ein Kandidat aus der Entscheidungsliste mittels Prioritätsregel selektiert, eingeplant und dadurch Teil der aktiven Liste A_n. Gleichzeit verliert diese Aktivität seine Mitgliedschaft in der Entscheidungsliste. Dieser Schritt wird solange wiederholt, bis die Entscheidungsliste der Iteration keine Aktivitäten mehr beinhaltet. Sind alle Aktivitäten Teil von C_n oder A_n so ist der gesamte Einplanvorgang beendet.

Die formale Darstellung der Entscheidungsliste lautet wie folgt:

$D_n = \{ j \mid j \notin (C_n \cup A_n), P_j \subseteq C_n , r_{jk} \leq \Delta R_k , \forall k \in K\}$

[81] (Kolisch, 1996b S. 323f)
[82] P-SGS: Parallel Schedule Generation Scheme [Engl.]

Die allgemeine Beschreibung der parallelen Methode lautet

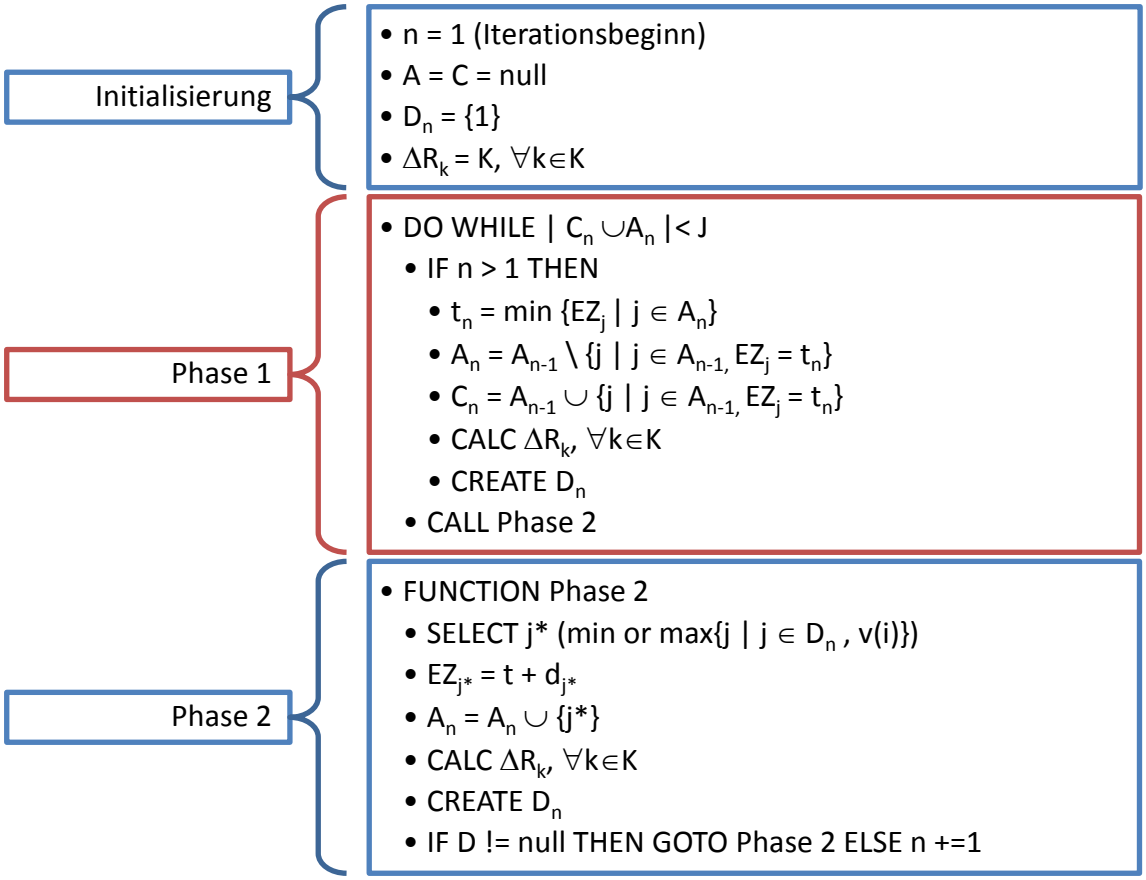

Abbildung 28: P-SGS zur Generierung eines Projektplans (Kolisch, 1996b S. 323)

n	t_n	ΔR_1	A_n	C_n	$A_n \cup C_n$	D_n	U_n	v(j)	j*	EZ_{j^*}
1	0	4	null	null	null	{1}	{2,…,7}	{0}	1	0
2	0	4	null	{1}	{1}	{2,3,4}	{5,6,7}	{1,1,2}	2	1
		2	{2}	{1}	{1,2}	{3}	{4,5,6,7}	{1}	3	1
		0	{2,3}	{1}	{1,2,3}	null	{4,5,6,7}	null		
3	1	4	null	{1,2,3}	{1,2,3}	{4,5}	{6,7}	{2,6}	4	3
		1	{4}	{1,2,3}	{1,2,3,4}	{5}	{6,7}	{6}	5	6
		0	{4,5}	{1,2,3}	{1,…,5}	null	{6,7}	null		
4	3	3	{5}	{1,2,3,4}	{1,…,5}	{6}	{7}	{6}	6	7
			{5,6}	{1,2,3,4}	{1,…,6}	null	{7}	null		
5	6	1	{6}	{1,…,5}	{1,…,6}	null	{7}	null		
6	7	4	null	{1,…,6}	{1,…,6}	{7}	null	{6}	7	7

Tabelle 6: Berechnung des P-SGS (Kolisch, 1995 S. 81)

Bezugnehmend auf das Beispielprojekt (Abbildung 25) und die dazu errechneten Prioritätswerte (Abbildung 26) kann das Ergebnis der parallelen Methode in Tabelle 6 dargestellt werden.

Am Beispiel der Iteration 2 kann die parallele Methode wie folgt erklärt werden: Zum Zeitpunkt 0 ist noch die komplette Restkapazität vorhanden. Es gibt keine aktiven Aktivitäten und die geplante Aktivität 1 (Teil von C_n). Die Aktivitäten {2,3,4} erfüllen die Ressourcenbedingung und Vorgängerbedingung und somit wird durch die Prioritätsregel (aufgrund der Gleichheit von v(2) und v(3) wird die Aktivität mit der niedrigeren Nummer gewählt) die Aktivität 2 für den Einplanvorgang gewählt. Dadurch vermindert sich die Restkapazität zum Zeitpunkt 0 auf 2 Einheiten und Aktivität 2 wird ein Teil der aktiven Aktivitätenliste A_n. Aufgrund der Ressourcenbedingung kann Aktivität 4 nicht mehr Teil der Entscheidungsliste D_n sein und daher wird die Aktivität 3 gewählt. Nachdem die Entscheidungsliste keine Aktivitäten mehr enthält, kann mit der nächsten Iteration (3) begonnen werden.

2.5.2.4. Kombination der Methoden und Prioritätsregeln

Die parallele oder serielle Methode zur Generierung eines Netzplans kann mit Prioritätsregeln kombiniert werden. Die in Kapitel 2.5.2.2 und 2.5.2.3 gezeigten Beispiele haben bereits die statische LFT Regel verwendet. Es wäre auch möglich gewesen, anstatt dessen die Aktivitäten nach dem Zufallsprinzip auszuwählen. Prioritätsbasierende Heuristiken die nur einen möglichen Projektplan generieren können nennt man Single Pass Methoden. Ist es möglich mit der Heuristik mehrere Projektpläne zu berechnen, so spricht man von Multi Pass Methoden.[83]

2.5.2.4.1. Single Pass Methoden

Diese älteste Technik der Prioritätsheuristiken verwendet die serielle oder parallele Methode in Kombination mit einer Prioritätsregel. Eine Auflistung der bekanntesten Implementationen ist in Kolisch und Hartmann (1999) vorzufinden.

[83] (Kolisch, et al., 1999 S. 155)

2.5.2.4.2. Multi Pass Methoden

Multiprioritätsmethoden

Diese Methoden bestehen aus mehreren Durchläufen, in denen jeweils ein neuer Projektplan errechnet wird, wobei eine Variation der Prioritätsregeln vorgenommen wird. Es ist möglich jeden Projektplan mit einer bestimmten Prioritätsregel zu erzeugen oder auch die Prioritätsregeln innerhalb des SGS zu variieren. Durch diese enormen Kombinationsmöglichkeiten sind der Anzahl generierter Projektpläne kaum Grenzen gesetzt. [84]

Vorwärts-Rückwärts Planmethoden

Die Vorwärts und/oder Rückwärtsmethode wird auf einen bestehenden Projektplan angewendet. Dabei bilden die Start- bzw. Endzeitpunkte des existierenden Projektplans die Grundlage für die Prioritätswerte im SGS. Die Aktivitäten werden so weit als möglich nach rechts oder links verschoben, um so einen neuen, optimierten Projektplan zu erhalten[85] Ursprünglich entwickelt von Jerome D. Wiest (1964) und weiterentwickelt von Özdamar und Ulusoy (1996), basierend auf Li und Willis (1992) haben Valls et al. (2005) gezeigt wie gut bestehende Heuristiken mittels der Vorwärts-Rückwärts Methode zu verbessern sind. Diese Methode ist in Kapitel 2.7 detailliert beschrieben. Hier ist auch eine Generalisierung der Methode enthalten.[86]

Sampling Methoden

Ähnlich der Simple Pass Methode verwenden die Sampling Methoden ein SGS mit einer Prioritätsregel. Durch den Einsatz von Wahrscheinlichkeitswerten p(j), die basierend auf den Prioritätswerten v(j) errechnet werden, erzeugt jeder Durchlauf einer Sampling Methode einen anderen Projektplan, da die Aktivitäten der Entscheidungsliste D_n aufgrund der errechneten Wahrscheinlichkeit selektiert werden. Die Errechnung der Wahrscheinlichkeit kann auf verschiedene Arten erfolgen, welche die Klassifizierung der Methoden determiniert.[87]

---

[84] (Kolisch, et al., 1999 S. 156)
[85] (Kolisch, et al., 1999 S. 156)
[86] (Valls, et al., 2006 S. 213ff)
[87] (Kolisch, et al., 1999 S. 156)

Random Sampling

In dieser Methode wird allen Aktivitäten die gleiche Auswahlwahrscheinlichkeit zugewiesen. Die Berechnung erfolgt durch:

$$p(j) = \frac{1}{|D_n|}$$

Formel 13: Wahrscheinlichkeitswerte (Random Sampling) (Schirmer, et al., 1997 S. 11)

Biased Random Sampling

Hier wird bei der Errechnung der Wahrscheinlichkeitswerte die Gewichtung der Prioritätswerte berücksichtigt. Davon ausgehend, dass die Auswahl der höher gewichteten Prioritätswerte bevorzugt wird, erfolgt die Berechnung durch:

$$p(j) = \frac{v(j)}{\sum_{i \in D_n} v(i)}$$

Formel 14: Wahrscheinlichkeitswerte (Max. BRS) (Schirmer, et al., 1997 S. 11)

Wenn die niedrigsten Prioritätswerte bevorzugt werden sollen, so ist die Berechnung folgendermaßen anzupassen[88]:

$$p(j) = \frac{\frac{1}{v(j)}}{\sum_{i \in D_n} \frac{1}{v(i)}}$$

Formel 15: Wahrscheinlichkeitswerte (Min. BRS) (Schirmer, et al., 1997 S. 11)

Bei der Anwendung der Biased Random Sampling Methode müssen 2 Limitationen berücksichtigt werden: Es sind nur positive Prioritätswerte zulässig und bei sehr hohen Prioritätswerten verlieren diese ihre Bedeutung und dadurch reduziert sich die Methode auf die Random Sampling Methode.[89]

[88] (Schirmer, et al., 1997 S. 11)
[89] (Kolisch, et al., 1996 S. 27f)

Regret Based Biased Random Sampling (R-BRS)

R-BRS ist eine Variation der Biased Random Sampling Methode, welche durch Drexl (1991) erstmalig verwendet wurde. In dieser Methode wird zuerst die Differenz des Prioritätswerts der Aktivität zum Maximalwert (Minimalwert) der Aktivitäten der Entscheidungsliste berechnet, mit der zusätzlichen Addition einer Konstante ε die Positivität des Prioritätswerts sichergestellt, mit einem weiteren Zusatzparameter α wird die Gewichtung (Bias) selbst beeinflusst und zuletzt der Wert mit der minimalsten (maximalsten) Wahrscheinlichkeit gewählt.[90]

$$v'(j) = \begin{cases} \max v(i) - v(j), & Zielfunktion => min \\ v(j) - \min v(i), & Zielfunktion => max \end{cases}$$

Formel 16: R-BRS: Prioritätswert Zielfunktion (Schirmer, et al., 1997 S. 13)

$$v''(j) = (v'(j) + \epsilon)^{\alpha}$$

Formel 17: R-BRS: Positivität der Wahrscheinlichkeit und Gewichtung (Schirmer, et al., 1997 S. 13)

$$p(j) = \frac{v''(j)}{\sum_{i \in D_n} v''(i)}$$

Formel 18: R-BRS: Wahrscheinlichkeit bestimmen (Schirmer, et al., 1997 S. 13)

Ein hoher Wert für den Parameter α eliminiert die Gewichtung und bei einem Wert von α=0 wird die Methode zu der Random Select Methode. Die folgende Tabelle zeigt die Wahrscheinlichkeitswerte der drei Methoden im Vergleich zueinander.

$j \in D_n$	2	3	4
v(j)	1	1	2
Random Sampling	0,33	0,33	0,33
BRS (max)	0,25	0,25	0,5
BRS (min)	0,4	0,4	0,2
R-BRS (max)	0,25	0,25	0,5
R-RBS (min)	0,4	0,4	0,2

Tabelle 7: Wahrscheinlichkeitswerte der Random Methode

[90] (Schirmer, et al., 1997 S. 13)

Die Basis für die Berechnung ist das vorhergehende Beispielprojekt (Abbildung 25). Es wurden die Prioritätswerte der Entscheidungsliste der Iteration 2 des S-SGS (Tabelle 5) als Basis für die Berechnung genommen.

Die Parameter α und ε wurden mit dem Wert 1 angenommen. Durch die Bereitstellung sehr einfacher Prioritätswerte, wurde bei BRS und R-BRS ein identes Ergebnis erzielt. Mit realitätsnäheren Prioritätswerten kann man die Unterschiede in den Ergebnissen besser erkennen:

$j \in D_n$	2	3	4
v(j)	15	21	25
Random Sampling	0,33	0,33	0,33
BRS (max)	0,25	0,34	0,41
BRS (min)	0,43	0,31	0,26
R-BRS (max)	0,65	0,29	0,06
R-RBS (min)	0,05	0,37	0,58

Tabelle 8: Wahrscheinlichkeitswerte der Random Methode (2)

2.6. Repräsentationslisten

Die bei dem Planungsprozess SGS erstellten Listen repräsentieren den Projektplan mit den gewählten Ausprägungen. Abhängig davon, welche Eigenschaft der Aktivität gewählt wird, sind folgende Repräsentationen von Listen möglich.[91]

2.6.1. Aktivitätslisten Repräsentation

Eine Aktivitätsliste $\lambda = \{j_1,\ldots,j_n\}$ ist eine Anordnung der Aktivitätsnummern, die alle Vorgängerbeziehungen berücksichtigt, und stellt die Grundlage der Projektplanung innerhalb des SGS dar. Für die Generierung der Initiallösung wurden die verwendbaren Prioritätsregeln bereits in Kapitel 2.5.2 vorgestellt. Alternativ dazu kann die Auswahl auch zufallsgeneriert erfolgen, die aber mit einer schlechteren Lösungsqualität verbunden ist. Bezüglich der Aktivitätsnummern gilt hier die Bedingung, dass Nachfolger immer höhere Nummern als die Vorgänger besitzen müssen. Folgende monadische Operatoren sind für die Veränderung einer Aktivitätsliste in der Literatur vorhanden[92]:

- Paarweises tauschen: Hier werden die Plätze zweier Aktivitäten j_q und j_s , $q, s \in \{1,\ldots,n\}$, $q \neq s$, der Liste vertauscht, sofern dies keine Vorgängerbeziehungen verletzt. Mit j_q und j_{q+1} , $q \in \{1,\ldots,n-1\}$ kann dieser Vorgang auf alle Aktivitäten der Liste angewendet werden, wobei wiederum gilt, dass eine Aktivität keinesfalls mit einem Vorgänger vertauscht wird.
- Aktivitäten verschieben: Unter Berücksichtigung der Vorgängerbeziehungen kann eine Aktivität nach einer beliebig anderen eingefügt werden. Der Verschiebeprozess muss nicht auf eine Aktivität beschränkt sein und kann auch durch ein inhaltliches Ziel gesteuert werden, welches z.B. die Beseitigung eines kritischen Pfades sein kann.
- Binäre Operatoren: Die Informationen zweier Aktivitätslisten können die Basis für eine neue Liste darstellen. Diese Technik findet vor allem bei genetischen Algorithmen im Crossover Bereich statt (siehe Kapitel 2.8.1)

[91] (Kolisch, et al., 1999 S. 159)
[92] (Kolisch, et al., 1999 S. 160f)

2.6.2. Zufallszahl Repräsentation

Die Zufallszahl Repräsentation $\rho = \{r_1,\ldots,r_n\}$ ordnet jeder Aktivität eine reelle Zahl r_j zu. Weitere Bezeichnungen in der Literatur für diese Art der Repräsentation sind die kodierte Zufallszahl-Repräsentation, Prioritätswert-Repräsentation oder auch Problemraum basierte Repräsentation, welche durch den Inhalt der reellen Zahl r_j determiniert werden. Analog zu der Aktivitätslisten-Repräsentation werden entweder Zufallszahlen oder Prioritätswerte für die Selektion der Aktivitäten der Entscheidungsliste (SGS) verwendet. Allerdings unterscheidet sich der erhaltene Vektor selbst, durch die Verwendung der Rechenwerte, anstatt Aktivitätsnummern zu verwenden. Zusätzlich zu den Operatoren der Aktivitätslisten-Repräsentation, können auch Methoden verwendet werden, welche die Vektorwerte selbst verändern. [93]

2.6.3. Sonstige Repräsentationen

Weitere Repräsentationen sind[94]:

- Prioritätsregel-Repräsentation: Gewählte Prioritätsregel für die Auswahl der Aktivität
- Projektplan-Repräsentation: Zusatzinformationen (z.B. Vorgängerbeziehungen) zu der Aktivität

[93] (Kolisch, et al., 1999 S. 161f)
[94] (Kolisch, et al., 1999 S. 163f)

2.7. Double Justification[95]

Die Grundlage für die Double Justification bildet der aktive Projektplan. Die in Kapitel 2.3 gezeigte Klassifizierung von Projektplänen, lässt sich auch auf eine Rechtsverschiebung anwenden. Ein aktiver Projektplan ist dadurch gekennzeichnet, dass auf keine Aktivität eine Rechtsverschiebung angewendet werden kann, weder lokal noch global, und somit keine Aktivität mehr später beendet werden kann, mit der Bedingung, dass es keine negativen Auswirkungen auf die Projektdauer geben darf und weiterhin alle Restriktionen der Nebenbedingungen (Kapitel 2.1) eingehalten werden. Die erste Aktivität (Quelle) ist bei der Linksverschiebung ausgenommen und die letzte Aktivität (Senke) bei der Rechtsverschiebung.

Durch diese Ergänzung wird die Klassifizierung des aktiven Projektplans um eine Dimension der Verschiebungsrichtung erweitert. Dadurch werden die Begriffe **links aktiver** Projektplan, welcher dem klassischen aktiven Projektplan entspricht, und **rechts aktiver** Projektplan eingeführt.

Eine Anwendung einer Rechtsausrichtung und anschließender Linksausrichtung auf einen bestehenden Projektplan wird „Double Justification“ genannt. Dazu werden die Aktivitäten des zu optimierenden Projektplans in absteigender Ordnung sortiert. Das Sortierkriterium ist hier der Endzeitpunkt der geplanten Aktivitäten. Die Linksausrichtung[96], besitzt eine analoge Vorgehensweise, wobei das Sortierkriterium durch den Anfangszeitpunkt der Aktivitäten determiniert wird und in aufsteigender Reihenfolge verwendet wird. Im Falle von gleichen Anfangs- oder Endzeitpunkten, muss ein weiteres Entscheidungskriterium angewendet werden, welches z.B. die Aktivitätsnummer[97] oder eine Zufallszahl[98] sein kann, und hat im letzteren Fall zur Folge, dass durch die Links- oder Rechtsausrichtung kein wiederholbares Ergebnis erzielt wird.

Die Verbesserung der Durchlaufzeit $T(S^R)$ eines rechts ausgerichteten Projektplans S^R kann an der Startzeit der Quellaktivität s^R_1 erkannt werden und ist im Falle des links ausgerichteten Projektplans S^L durch die Durchlaufzeit $T(S^L) = s^L_n$ selbst ersichtlich.

[95] (Valls, et al., 2006 S. 209ff)
[96] Left Justification [Engl.]
[97] (Hartmann, 1998 S. 3)
[98] (Valls, et al., 2005)

Weiters kann ein rechts ausgerichteter Projektplan, genauso wie der links ausgerichtete, niemals eine längere Projektdauer als der Ursprungsprojektplan S aufweisen:

$$T(S^R) \leq T(S) \quad \text{und} \quad T(S^L) \leq T(S)$$

Auf die Ursprünge der Double Justification ist in Kapitel 2.5.2.4.2 verwiesen, welche jedoch die Reihenfolge der Aktivitäten bei der Links- oder Rechtsverschiebung außer Acht lassen und somit nur indirekt als Vorgänger dieser Lösungsmethode gelten.[99]

Valls, Ballestin und Quintanilla 2005 zeigen in ihrer Studie die Anwendungsmöglichkeit der Double Justification auf verschiedene Heuristiken und Metaheurisitken und die damit verbundene Verbesserung der Projektdauer und vor allem die Rechenzeitersparnis.

Eine Variation der Double Justification stellt die **« Double Justification von auswählbaren Aktivitäten »** dar. Hier wird als Auswahlkriterium für die verfügbaren Aktivitäten der Rechtsausrichtung das Vorhandensein bereits rechts ausgerichteter Nachfolger festgelegt. Wenn alle Nachfolger dieser Aktivität bereits ausgerichtet sind, so ist diese ein Kandidat für den Ausrichtungsprozess.[100] Für die Auswahl der Aktivität sind drei Varianten möglich. Die Wahl kann

(i) zufallsgeneriert,
(ii) durch die spätesten Endzeitpunke der theoretisch möglichen Rechtsverschiebung der verfügbaren Aktivitäten, wobei im Falle von gleichen Werten die Endzeitpunkte des zu optimierenden Projektplans als weiteres Entscheidungskriterium herangezogen werden, oder
(iii) durch die spätesten Endzeitpunke der theoretisch möglichen Rechtsverschiebung der verfügbaren Aktivitäten, gefiltert auf maximal zwei Aktivitäten, mit der größten Dauer, entnommen aus dem zu optimierenden Projektplans

erfolgen.[101]

[99] (Valls, et al., 2005 S. 377)
[100] (Valls, et al., 2006 S. 209f.)
[101] (Valls, et al., 2006 S. 216)

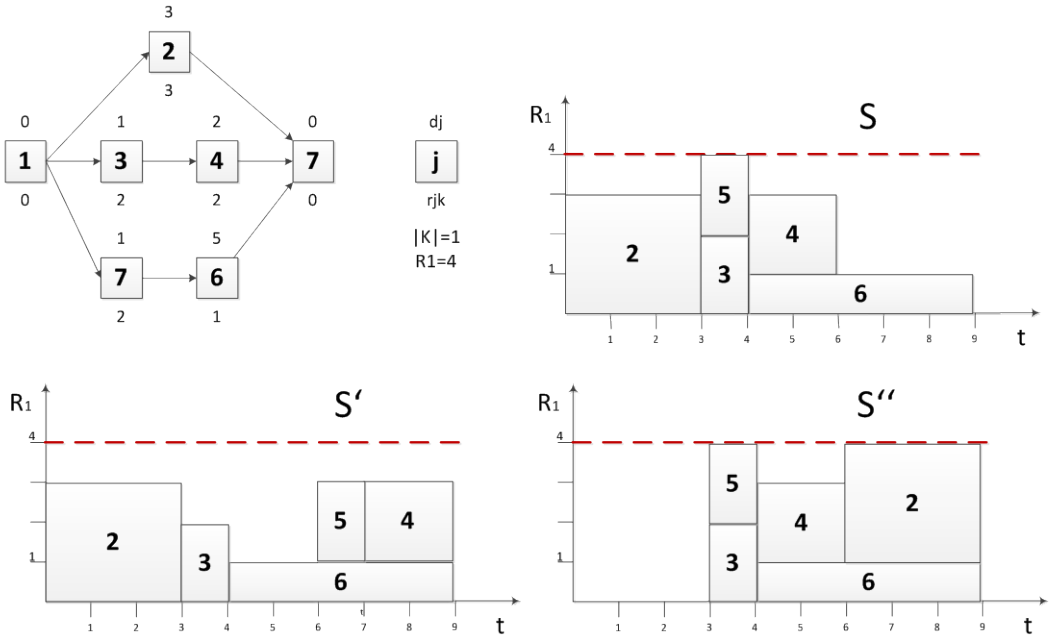

Abbildung 29: Varianten der Double Justification (Valls, et al., 2006 S. 211)

Im Falle der Linksausrichtung von auswählbaren Aktivitäten wird das Vorhandensein von links ausgerichteten Vorgängern für die Aufnahme in die Entscheidungsliste vorausgesetzt. [102] Die Auswahl der Aktivität erfolgt analog zur Rechtsverschiebung. Die Selektion kann wiederum

(i) zufällig erfolgen, oder

(ii) es ist der früheste Beginnzeitpunkt bei theoretischer Linksverschiebung der Aktivität maßgeblich, welcher auch als Kriterium bei Gleichheit der ersten Bedingung dient, allerdings entnommen aus dem zu optimierenden Projektplan, oder

(iii) die Wahl erfolgt durch den frühest theoretisch möglichen Beginnzeitpunkt der Aktivitäten mit den kürzesten Laufzeiten.

Das unterschiedliche Ergebnis zwischen normaler Rechtsausrichtung und Rechtsausrichtung von auswählbaren Aktivitäten kann man in Abbildung 29 erkennen. Die klassische Double Justification wählt bei der Rechtsausrichtung (S‘) von einem existierenden Projektplan (S) aufgrund der Ordnung der Aktivitäten nach Endzeitpunkten zuerst die Aktivitäten {6,4,3,5} bzw. {6,4,5,3} aus und kann aufgrund der Ressourcenbeschränkungen die Aktivität 2 nicht mehr verschieben. Dadurch wird die Durchlaufzeit nicht verringert. Durch die Erweiterung „auswählbare Aktivitäten“

[102] (Valls, et al., 2006 S. 210)

findet die Aktivität 2 den Weg in die Entscheidungsliste und ein rechtsausgerichtes Einplanen der Aktivität führt insgesamt zu einer Verbesserung der Durchlaufzeit des Projektes S''.[103]

Der Einsatz dieser alternativen Auswahl und Ausrichtungstechnik in dem bestehenden hybriden genetischen Algorithmus von Valls et. al (2008) [104] ermöglicht eine Verbesserung dieses Algorithmus bei dem J120 Standard-Set mit 5000 generierten Projektplänen um absolut 0,3 %, wenn der Double Justification Prozess bis zum 1000sten Projektplan durch „Double Justification von auswählbaren Aktivitäten" ersetzt wird. Allerdings ist damit eine negative Auswirkung der durchschnittlichen Rechenzeit von ca. 2,7 Sekunden verbunden.[105]

[103] (Valls, et al., 2006 S. 216)
[104] erstmalig veröffentlicht: (Valls, et al., 2003b)
[105] (Valls, et al., 2006 S. 220)

2.8. Metaheuristiken

Die bisherig erwähnten heuristischen Verfahren sind auf die Implementation einer oder die Kombination mehrerer Methoden beschränkt. Metaheuristiken versuchen durch Ansätze, die teilweise der Natur entnommen sind, durch die dynamische Adaption bestehender Verfahren eine Näherungslösung zu finden, mit dem Bestreben, nicht in einem lokalen Optimum zu verbleiben.[106] Eine sehr gute Übersicht der verfügbaren Metaheuristiken der letzten 10 Jahre bieten Hartmann, et al. (2010), Kolisch, et al. (2006) und Kolisch, et al. (2000).

2.8.1. Genetische Algorithmen (GA)

Die natürliche Auslese und Genetik bilden die Grundlage für die Suchtechnik des genetischen Algorithmus (in Folge GA genannt). Durch eine Kreuzung der genetisch fittesten Individuen mit anderen Individuen einer Zufallsmenge und anschließender Auslese, d.h. die fittesten kombinierten und bisherigen Individuen werden für die nächste Generation als Basis herangezogen, entstehen neue, bessere Lösungen. Dadurch können die historischen Informationen der vorangehenden Generationen weitergegeben werden und durch die Zugabe neuer Informationen Wege aus lokalen Optima hin zum globalen Optimum gefunden werden. Die gefundene Lösung der GA stellt aber nicht zwangsläufig ein Optimum dar, da es sich hier ebenfalls um ein Näherungsverfahren handelt.[107]

Der GA benötigt zuerst eine **Anfangspopulation**, die im Falle des RCPSP durch Projektpläne repräsentiert wird. Diese Population bildet die Grundlage für den **Reproduktionsprozess**. Die Auswahl der Individuen der nächsten Generation kann nach bestimmten Regeln erfolgen, am einfachsten zufallsgeneriert, gewichtet nach der Fitness f (Biased Random Methode). Durch ein **Crossover** werden die Teile von zwei Individuen kombiniert, wobei, abhängig von der Problemklasse, gewisse Regeln bei der Kombination eingehalten werden müssen (z.B. Vorgängerbeziehungen im RCPSP). In Analogie zu der natürlichen Mutation, kann die anschließende **Mutation** des GA als

[106] (Zimmermann, 2008 S. 298)
[107] (Goldberg, 1989 S. 1ff)

Reparaturfunktion angesehen werden, um genetisches Material wiederherzustellen, welches durch die Reproduktion und den Crossover unbrauchbar geworden wäre. [108]

Das Gen ist die kleinste Einheit innerhalb des GA. Durch die Gene wird ein Chromosom gebildet, wobei die Position des Gens Genort[109] genannt wird und dessen Ausprägung, oder Wert Allele. Im Falle des RCPSP besteht das Chromosom aus Aktivitäten, in denen, im Normalfall, das Allele die Aktivitätsnummer repräsentiert und der Locus die Anordnung der Aktivität. Durch die Kombination entsteht ein Individuum (die Aktivitätenliste), das auch als Genotyp bezeichnet wird. Der Phänotyp, das Gesamtsystem, wird durch den Projektplan dargestellt. [110]

2.8.1.1. Die Schematheorie[111]

In der Schematheorie wird das Chromosom durch ein Zeichen A[112] repräsentiert, dass aus sieben Bits (Genen) besteht (ASCII[113] Zeichen). Dabei kann jedes Bit einen Zustand aus dem Alphabet V = {0,1} einnehmen. Zusätzlich wird das Alphabet des Bits um den Platzhalter * erweitert, der beide Zustände repräsentiert. Das erweiterte Alphabet V+ = {0,1,*} wird nun für die Schemagenerierung herangezogen. Ein **Schema** λ wird aus dem Alphabet V+ gebildet und für das Zeichen A ist nur das Alphabet V zulässig. Das Zeichen A_1=(1,0,1,1,1,1,1) ist ein gültiger Repräsentant des 7-bit Schemas λ = (*,0,*,1,*,*,1) mit vier unbestimmten Genen. Die Anzahl der unbestimmten Gene a eines Schemas bestimmen die Anzahl möglicher Zeichen (2^a). Umgekehrt kann, ausgehend von einem Zeichen, die Anzahl möglicher Schemata berechnet werden. Für ein Zeichen der Länge l gibt es 2^l Schemata. Die **Ordnung** o eines Schemas wird durch die Anzahl der Gene die Teil des Alphabet V sind determiniert. Der größte Abstand zwischen zwei bestimmten Genen legt die **bestimmte Länge** δ fest. Das oben angeführte Schema λ besitzt die Ordnung O=3 und die Länge δ=5 (Position 7 minus Position 2).

Davon ausgehend, dass ein Zeichen A_i im **Reproduktionsprozess** aus einer Population A(g), der Generation g_1, mit einer Größe von n Zeichen, in der das Schemata λ m =

[108] (Goldberg, 1989 S. 10ff)
[109] Locus [Engl.]
[110] (Goldberg, 1989 S. 21)
[111] (Goldberg, 1989 S. 28f)
[112] String [Engl.]
[113] ASCII: American Standard Code for Information Interchange [Engl.] (Hansen, et al., 2001 S. 1004)

$m(\lambda,g_1)$ mal auftritt, ausgewählt wird, ist durch die Wahrscheinlichkeit $p_i=f_i/\sum f_j$ (f = Fitness) festgelegt. Es kann nun ein durchschnittlicher Fitnesswert $f(\lambda)$ für das Schema λ bestimmt werden. Durch die Annahme, dass das Vorkommen des Schemas in der nächsten Generation $g_2=g_1+1$ durch $m(\lambda,g_{2)} = m(\lambda,g_1) * n * f(\lambda)/\sum f_j$ und die Berechnung der durchschnittlichen Fitness einer Generation mit $\bar{f} = \sum f_j/n$ determiniert wird, kann die Wachstumsrate des Schemas abgeleitet werden:

$$m(\lambda, g_{0+G}) = m(\lambda, g_0) * \left(\frac{f(\lambda)}{\bar{f}}\right)^G$$

Formel 19: Wachstum eines Schemas durch den Reproduktionsprozess (Goldberg, 1989 S. 30)

Hiermit wird ersichtlich, dass überdurchschnittlich fitte Schemata mit exponentieller Geschwindigkeit wachsen.

Das Überleben eines Schemas in dem **Crossover-Prozess** wird durch die Zerstörungswahrscheinlichkeit des Schemas, welche sich durch das Verhältnis zwischen der Länge l des String und der bestimmen Länge δ definiert, also ob der Kreuzungspunkt innerhalb des Schemas liegt, festgelegt und wird zusätzlich, durch eine optionale Anwendungswahrscheinlichkeit p_C des Crossovers beeinflusst. Dadurch kann die Wahrscheinlichkeit des Überlebens eines Schemas wie folgt dargestellt werden:

$$p_S \geq 1 - p_C * \frac{\delta(\lambda)}{l-1}$$

Formel 20: Wachstum eines Schemas durch den Crossover-Prozess (Goldberg, 1989 S. 32)

Durch eine Erweiterung des Reproduktionsprozesses, der noch keine Veränderung an der Population selbst vornimmt, durch den Crossover-Prozess, kann das Vorkommen eines Schemas in der nächsten Generation wie folgt errechnet werden:

$$m(\lambda, g_{0+G}) \geq m(\lambda, g_0) * \left(\frac{f(\lambda)}{\bar{f}}\left(1 - p_C * \frac{\delta(\lambda)}{l-1}\right)\right)^G$$

Formel 21: Wachstum eines Schemas durch den Reproduktions- und Crossover-Prozess (Goldberg, 1989 S. 32)

Durch den kombinierten Prozess von Reproduktion und Crossover unterliegt das Wachstum des Schemas in neuen Generationen der durchschnittlichen Fitness des Schemas im Vergleich zur Gesamtpopulation und der bestimmten Länge des Schemas.

Der **Mutationsprozess** führt mit einer Wahrscheinlichkeit p_M eine abschließende Veränderung an der neuen Generation durch. Jedes Gen überlebt daher mit der Wahrscheinlichkeit 1- p_M und da für das Überleben eines Schemas nur die bestimmten Gene maßgeblich sind, dessen Anzahl durch die Ordnung o determiniert ist, kann die gesamte Überlebenswahrscheinlichkeit mit $p_S = 1 - p_M * o(\lambda)$ festgelegt werden. Dadurch ist die Gleichung der Schematheorie, die besagt, dass überdurchschnittlich fitte Schemata, mit einer kurzen bestimmten Länge und von niedriger Ordnung sich exponentiell ausbreiten, vollständig[114]:

$$m(\lambda, g_{0+G}) \geq m(\lambda, g_0) * \left(\frac{f(\lambda)}{\bar{f}} \left(1 - p_C * \frac{\delta(\lambda)}{l-1} - (p_M * o(\lambda)) \right) \right)^G$$

Formel 22: Wachstum eines Schemas durch den Reproduktions-, Crossover- und Muations-Prozess (Goldberg, 1989 S. 33)

2.8.1.2. Das Entscheidungsproblem

Die Grundtheorie des GA zeigt den Grad der Ausbreitung von Schemata. Es stellt sich dazu die grundlegende Frage, ob dies zur Optimalität der Lösung beiträgt, da durch die exponentielle Verbreitung der fittesten Schemata, der Raum für andere Schemata, die für die fittesten Schemata eine Verbesserungsmöglichkeit darstellen, in der Population des GA sehr klein wird. Ein ähnliches Dilemma ist in der Spieltheorie zu finden. Hierbei wird ein Automat für das Glückspiel (der zwei-armige Bandit) herangezogen, der zwei Spielmöglichkeiten und einem damit verbundenen Gewinn anbietet. Es ist in dieser Theorie möglich, einen erwarteten Verlust zu berechnen, allerdings sind die Erwartungswerte der Gewinnmöglichkeiten und dessen Varianzen, eine Voraussetzung dafür. Da diese Kenngrößen nicht zur Verfügung stehen, muss durch ein Testen der Lösungen, welches dem Spielen am Automat entspricht, die bessere Seite herausgefunden werden. Hierbei entsteht aber die Gefahr, dass durch ein verfrühtes Ende der Testphase, die Wahrscheinlichkeit die falsche Seite zu verwenden, erhöht wird. Eine wiederum zu lange Testphase schränkt die Gewinnmöglichkeit aufgrund der verbliebenen geringen Anzahl an Versuchen stark ein. Die vorgestellte Schematheorie bietet hier eine Alternative. Bezogen auf das zweiarmige Banditenproblem ist die

[114] (Goldberg, 1989 S. 32f)

Lösungssuche im GA ein k-armiges Banditen-Problem. Bei Schemata mit der Ordnung o, gilt es 2^o-armige Banditen-Probleme zu lösen.[115]

2.8.1.3. Die implizite Parallelität

Die implizite oder intrinsische Parallelität wird in der Literatur oft als effektiver Vorteil des GA genannt. Hier wird angenommen, dass der GA während der gesamten Berechnung, abhängig von der Populationsgröße n insgesamt n^3 Schemata parallel verarbeitet. Ausgehend von einer Zeichenlänge l und einer Schemalänge l_S=l/2, kann ein Zeichen durch eine l- l_S+1-malige Verschiebung durchlaufen werden. Bei einer Fixierung eines Gens des Schemas, kann eine Gesamtanzahl von 2^{l_s-1} Schemata errechnet werden. Bei einer Populationsgröße n wird somit die Anzahl von 2^{l_s-1}*n*(l- l_S+1) Schemata erreicht. Ausgehend davon, dass Schemata redundant vorkommen können, wird die Populationsgröße auf n=$2^{l_s/2}$ festgelegt. Bei einer Betrachtung der Schemata mit einer höheren Ordnung als l_S/2, die aufgrund der Binomialverteilung die Hälfte der Schemata repräsentieren, kann als untere Grenze für die Anzahl der Schemata n * (l- l_S+1) * 2^{l_s-2} festgelegt werden. Durch die Reduzierung der Populationsgröße n=$2^{l_s/2}$, kann diese untere Schranke auf n^3* (l- l_S+1)/4 fixiert werden und dadurch das Verhältnis zwischen Populationsgröße und Anzahl Schemata gezeigt werden.[116]

2.8.2. Implementationen von GA

In weiterer Folge werden die derzeit drei besten genetischen Algorithmen vorgestellt, sowie der GA von Hartmann, der einige Basiselemente für diese und andere GA liefert. Der GA von Valls et. al (2008), als einer der drei besten GA, der dem Anwendungsteil dieser Diplomarbeit zugrunde liegt und ebenfalls ein zentrales Element im Crossover-Bereich beisteuert, ist in Kapitel 3.1 beschrieben. Eine kurze Übersicht dieser Algorithmen ist in Tabelle 9 zu finden.

[115] (Goldberg, 1989 S. 36ff)
[116] (Goldberg, 1989 S. 40f)

GA Bezeichnung	Autor	Jahr	Charakteristika
Permutation (Kapitel 2.8.2.1)	Hartmann	1998	• BRS (S-SGS,LFT) • Point Crossover • Pair Mutation • Rang/RBRS/Turnier Selektion
Prioritätswert (Kapitel 2.8.2.2)	Hartmann	1998	• S-SGS Zufalls- Prioritätswert • Point Crossover • Pair Mutation • Rang/RBRS/Turnier Selektion
Prioritätsregel (Kapitel 2.8.2.3)	Hartmann	1998	• S-SGS zufällige Wahl der Prioritätsregel • Point Crossover • Pair Mutation Rang/RBRS/Turnier Selektion
Dekomposition (Kapitel 2.8.2.4)	Debels Vanhoucke	2007	• Dekomposition • S-SGS Zufalls- Prioritätswert • 2 Point Crossover (Area-RUR(max) based) • Priority-Value based Mutation
Hybrider GA (Kapitel 3.1)	Valls et al.	2008	• Neighborhood Search • BRS (S-SGS,LFT) • k Point Crossover (Point-RUR(max) based) • Pair Mutation
Froschsprung (Kapitel 2.8.2.5)	Fang Wang	2010	• BRS (S-SGS,LFT) • 2 Point Crossover (Point-RUR(min) based) • Evaluated Pair Mutation

Tabelle 9: Genetische Algorithmen für das RCPSP

Eine Gegenüberstellung der Leistungscharakteristika der angegebenen GA ist in Tabelle 10 dargestellt, die für die PSPLIB Standard-Instanzen J30 und J120 vorgenommen worden ist. Es ist für die J30 Instanz die durchschnittliche Abweichung von der optimalen Lösung (Rechenzeit in Sekunden) angeführt und für die J120 Instanz die durchschnittliche Abweichung von der optimalen Projektdauer ohne Ressourcenbeschränkungen (Rechenzeit in Sekunden). Es sind jeweils maximal 1.000, 5.000 und 50.000 Projektpläne errechnet worden. Zu den durchschnittlichen

Rechenzeiten ist anzumerken, dass ein Vergleich der Algorithmen selbst keine objektiven Ergebnisse zeigen kann, da verschiedene Prozessoren verwendet worden sind, jedoch die CPU Zeiten innerhalb der einzelnen Algorithmen eine gute Vergleichbarkeit bieten.

GA Bezeichnung	Autor	J120			J30		
		1000	5000	50000	1000	5000	50000
Permutation	Hartmann	39,37	36,74	34,03	0,54 (0,54s)	0,25	0,08
Prioritätswert	Hartmann	45,82	45,25	38,83	1,03 (0,64s)	0,56	0,23
Prioritätsregel	Hartmann	39,93	38,49	36,51	1,38 (0,91s)	1,12	0,88
Dekomposition (nur GA)	Debels et al.	34,19 (0,05s)	32,34 (0,27s)	30,82 (3,0s)	0,15 (0,01s)	0,04 (0,05s)	0,02 (0,5s)
Dekomposition (komplett)	Debels et al.	**33,55** (0,05s)	**32,18** (0,27s)	**30,69** (3,0s)	**0,12** (0,01s)	**0,04** (0,05s)	0,02 (0,5s)
Hybrider GA	Valls et al.	34,07	32,54 (2,03s)	31,24	0,27	0,06 (0,31s)	**0,01**
Hybrider GA Variante[117]	Valls et al.		32,27 (4,74s)				
Froschsprung	Fang Wang	34,83 (5,69s)	33,20 (30,7s)	31,11 (333s)	0,36 (0,13s)	0,21 (0,65s)	0,18 (5,7s)

Tabelle 10: Genetische Algorithmen für das RCPSP - Performance[118]

2.8.2.1. GA Hartmann 1998 – Permutation[119]

Die **Initialpopulation** wird mit der Biased Random Sampling Methode (S-SGS, LFT Prioritätsregel) erzeugt. Versuche mit der Random Sampling Methode zeigten schlechtere Ergebnisse. Aus der generierten Aktivitätsliste wird ein Projektplan

[117] Verbesserung des Hybriden GA von Valls et al (2008), basierend auf Kapitel 2.7 (Double justification von auswählbaren Aktivitäten)
[118] (Debels, et al., 2007), (Fang, et al., 2012), (Hartmann, 1998), (Kolisch, et al., 2006), (Valls, et al., 2008), (Valls, et al., 2006)
[119] (Hartmann, 1998 S. 6f)

erzeugt, bei der jede Aktivität in Reihenfolge der Liste so früh als möglich eingeplant wird und somit ein aktiver Projektplan entsteht.

Die geordneten Aktivitätslisten, welche durch das S-SGS produziert wurden, auch Jobsequenz genannt, dienen als Basis für die Erzeugung der nächsten Generation. Es gibt in diesem GA 3 Variationen des **Crossovers.** Durch das Erzeugen einer Zufallszahl q zwischen 1 und der Anzahl Aktivitäten N wird der Punkt bestimmt, der für den **Ein-Punkt Crossover** maßgeblich ist. Bis zu dieser Stelle werden bei der Tochter T alle Aktivitäten der Mutter M gewählt. Anschließend werden die Aktivitäten des Vaters V der Tochter hinzugefügt, wobei die bereits hinzugefügten Mutteraktivitäten ignoriert werden. Der Sohn S wird mit umgekehrten Elternteilen analog erzeugt. Die Variation **Zwei-Punkt Crossover** erzeugt zwei zufallsgenerierte Schnittpunkte $\{q_1,q_2\}$ zwischen 1 und N . Bis zum Punk q_1 wird wie in Variante 1 verfahren. Ab q_1+1 und bis q_2 werden q_2-q_1-1 Vateraktivitäten wie in Variante 1 eingeplant, allerdings nur bis zum Punkt q_2 und nicht bis N. Die verblieben Mutteraktivitäten, die noch der Tochter (sowohl vom Vater, als auch der Mutter) hinzugefügt wurden, sind nun einzuplanen. Der Sohn wird wieder analog erzeugt. Der **Uniform Crossover** Operator erzeugt für jede Aktivität der Jobsequenz eine Entscheidungsvariable $p_i \in \{0,1\}$. Bei der Generierung der Tochter werden bei p_i=1(0) die Mutteraktivitäten (Vateraktivitäten) gewählt. Es wird nun immer nach dem niedrigsten Index in der Mutteraktivitätsliste (Vateraktivitätsliste) gesucht und die Aktivität gewählt, die noch nicht Teil der Tochter ist.

Die **Mutation** tauscht, mit einer Mutations-Wahrscheinlichkeit p_{random}, sofern die Vorgängerbeziehungen nicht verletzt werden, die Aktivitäten paarweise (j_i <-> j_{i+1} | i = 2, … N-2).

Die **Selektion** kann mit einer **Rangmethode** (sortieren aller Elemente und nur die Anzahl Individuen, die der Populationsgröße entspricht, werden weitergeben) oder mit einer **proportionalen Selektion** (die Wahrscheinlichkeit entfernt zu werden, wird mit der Regret Based Biased Sampling Methode durchgeführt, wobei die Fitness des Projektplan dem Prioritätswert v(i) entspricht und ε=1 und α=2 ist) erfolgen. Weiters ist es noch möglich mit der **Turnier Selektion** mit 2 Teilnehmern (Zwei zufällig gewählte Individuen müssen gegeneinander antreten und wenn der erste nicht besser ist als der zweite, so wird er eliminiert) und eine Erweiterung auf 3 Teilnehmer

(Individuum eins muss zusätzlich gegen Individuum drei bestehen) die Anzahl der Populationsgröße zu erreichen.

Die besten Werte konnte der GA mit einer Mutationswahrscheinlichkeit von 0,05, dem Zwei Punkt Crossover, der Rangselektion, einer Populationsgröße von 40 Individuen und 25 Generationen, das zu einer Generierung von 1000 Projektplänen führt, erreicht werden.

2.8.2.2. GA Hartmann 1998 – Prioritätswert[120]

Die **Initialpopulation** wird mit einer Kombination aus S-SGS und zufallsgenerierten Prioritätswerten $pv_i \in \{0,1\} \mid i = \{1,\dots,N\}$ erzeugt, wobei immer der höchste Wert gewählt wird. Die Einplanung der gewählten Aktivität der Entscheidungsliste erfolgt wieder so früh als möglich.

Der **Crossover,** die **Mutation** und die **Selektion** erfolgen wie in Kapitel 2.8.2.1, mit der Änderung, dass die Prioritätswerte an die nächste Generation weitergegeben werden.

2.8.2.3. GA Hartmann 1998 – Prioritätsregel[121]

Hier wird bei der Generierung der **Initialpopulation** ein Set von sechs verschiedenen Prioritätsregeln (LFT, LST, MTS, MSLK, WRUP, GRPW) verwendet. WRUP ist in Kapitel 2.5.2.1 nicht definiert und lautet wie folgt: 0,7 * $|S_j|$+ 0,3 * $\sum_{k\in K} r_{jk}/R_k$ (Zusammenstellung aus der direkten Nachfolgersumme und dem gewichteten Ressourcenverbrauch). Der Einplanvorgang besteht wiederum aus einem S-SGS, wobei bei der Auswahl einer Aktivität der Entscheidungsliste eine Prioritätsregel zufallsgeneriert aus einem Set an Regeln gewählt wird und der maximale Prioritätswert für die Selektion verwendet wird.

Der **Crossover,** die **Mutation** und die **Selektion** erfolgen wie in Kapitel 2.8.2.1, mit der Änderung, dass die Information der Prioritätsregeln an die nächste Generation weitergegeben wird.

[120] (Hartmann, 1998 S. 7f)
[121] (Hartmann, 1998 S. 9f)

2.8.2.4. GA Debels und Vanhoucke 2007 - Dekomposition[122]

Der GA von Debels und Vanhoucke stellt den derzeit besten GA im Bereich der besonders schwierig zu lösenden PSPLIB[123] J120 Instanzen dar. Der Dekompositionsteil teilt das RCPSP und löst das Subproblem mit einem genetischen Algorithmus.

Ein serieller SGS mit zufallsgenerierten Prioritätswerten (Kapitel 2.8.2.2) bildet die **Initialpopulation,** die hier auch als Linkspopulation bezeichnet wird. Der erhaltene RK Vektor enthält die Endzeitpunkte. Für die **Selektion** werden die Elemente der Population iteriert und jeweils mit dem Gewinner einer Turnier Selektion, mit zufallsselektierten 2 Teilnehmern, kombiniert. Die Mutter- bzw. Vatereigenschaft der selektierten Elemente wird zufallsgeneriert bestimmt.

Die durchschnittliche Ressourcenauslastung RUR von Valls et. al (2008, siehe Kapitel 3.1.2.3), zur Bestimmung von Ressourcenspitzen, dient als Berechnungsgrundlage für ein **Zwei-Punkt Crossover.** Die Länge l des Intervalls wird zufallsgeneriert aus 0,25*PL (Projektlänge) und 0,75*PL gewählt. Auf der Basis der RUR-Formel wird für jeden Zeitpunkt t (0≤t≤PL-l) die gesamte Ressourcenauslastung TRU berechnet und der Startpunkt gewählt, dessen Bereich die maximale Auslastung besitzt.

$$TRU(t,l,S) = \sum_{time=t}^{t+l-1} RUR(time,S)$$

Formel 23: Gesamte Ressourcenauslastung (Debels, et al., 2007 S. 460)

Der Aufbau des Kindes erfolgt in drei Schritten, wobei in diesem Prozess ein Zwischenvektor v_c produziert wird. Zuerst werden alle Aktivitäten des Vaters gesucht, deren Prioritätswerte geringer oder gleich sind als der erste Crossover-Punkt. Es wird an der Indexstelle dieser Vateraktivitäten im Muttervektor (der die Endzeitpunkte der Aktivitäten repräsentiert) eine vorab definierte Konstante c abgezogen und analog für Aktivitäten des Vaters, deren Prioritätswerte größer oder gleich dem zweiten Crossover-Punkt sind, hinzugezählt. Der Bereich zwischen den Crossover-Punkten wird mit den Prioritätswerten des Vaters aufgefüllt. Ist die durchschnittliche Summe der absoluten Abweichungen zwischen den Prioritätswerten der Eltern kleiner als eine vorgegebene Konstante τ, dann wird eine **Mutation** des Vektors v_c mit einer auf einige

[122] (Debels, et al., 2007 S. 457ff)
[123] Siehe Kapitel 2.9.2

Aktivitäten eingeschränkte Addition einer Zufallszahl zwischen –n und n durchgeführt. Aus diesem Vektor v_c wird dann ein rechtsausgerichteter Projektplan erzeugt.

Für die nächste Generation werden die Kinder berücksichtigt, die die geringsten Werte der Zielfunktion aufweisen, allerdings wird sichergestellt, dass die bis dato gefundene beste Lösung des GA dadurch nicht verloren geht. Die Erzeugung der rechtsausgerichteten Projektpläne stellt nur einen Teil der Generationsgenerierung dar. Im zweiten Teil wird die Population nochmals einem Crossover und einer Mutation unterworfen, allerdings zum Schluss linksgerichtet.

Die Erweiterung „Dekomposition" besteht nun aus den Schritten Teilprobleme erzeugen, GA Teil ausführen und die Teile zusammenführen. Zwischen zwei Zeitintervallen pt_1 und pt_2 eines bestehenden Projektplans S_n werden bei dem Teilproblem alle Aktivitäten bis pt_2 mit der LST Prioritätsregel normal eingeplant. Danach wird ein Zeitfenster von $lft = \max\{f_i \mid s_i < pt_1\}$ und $est = \min\{s_i \mid f_i > pt_2\}$ errechnet. Die Aktivitäten innerhalb von lft und est werden als Kern C gespeichert, jene die nur teilweise in pt_1 und pt_2 ihre Dauer haben in der Liste B_1 und B_2 und die Vorgängerbeziehungen in der Liste A_{sub}. Der GA muss im Bereich des SGS so angepasst werden, dass die Aktivitäten von B_1 nach den S_n Starzeiten geplant werden, Core wird wie bisher geplant und B_2 wieder nach den S_n Starzeiten. Es gilt zu beachten, dass alle Aktivitäten von B_1 mit ihrer Teildauer des Bereichs pt_1 und pt_2 geplant werden. B_2 Aktivitäten sind mit ihrer kompletten Dauer einzuplanen, aber danach ist die Dauer auf die zuerst ermittelte Teildauer zu reduzieren. Für die Zusammenführung wird wieder ein RK Vektor aus dem Kern (pt_1 wird zu den Zeiten addiert) und den restlichen Aktivitäten gebildet und damit ein neuer Projektplan S* erstellt.

2.8.2.5. GA Fang und Wang 2012 – Froschsprungalgorithmus[124]

Dieser GA ist inspiriert durch das Jagdveralten von Fröschen. Ein künstlicher Frosch repräsentiert eine erweiterte Aktivitätenliste, die zusätzlich zu der Anordnung der Aktivitäten auch die Start und Endzeiten dieser enthalten. Die **Initialpopulation** wird mit einem S-SGS (LFT Prioritätsregel) erzeugt, die anschließend einer Double Justification unterworfen wird und in absteigender Reihenfolge sortiert. Danach wird

[124] (Fang, et al., 2012 S. 890ff)

die Population in m Kulturen[125] geteilt, die sich unabhängig voneinander entwickeln sollen. Die Zuordnung der n Frösche erfolgt über eine Methode, die den ((i-1) * n + j)ten Frosch der Kultur j zuweist (m=3, n=4, m_1={1,4,7,10}, m_2={2,5,8,11} usw.). Diese Kulturen werden in Subkulturen mit einer Auswahlwahrscheinlichkeit von p_j=2*(n+1-j)/(n(n+1)), in der jeweils q Frösche enthalten sind, geteilt.

In der **Selektion** findet das Froschsprungverhalten ihre Anwendung. Es wird der schlechteste Frosch P_W jeder Subkultur mit dem besten Frosch P_B der Kultur gekreuzt und anschließend das Ergebnis bewertet. Tritt eine Verschlechterung auf, so wird die Prozedur mit dem generell besten Frosch P_G wiederholt. Die positive bzw. negative Sprungweite, um einen neuen Frosch zu selektieren beträgt dabei min(int(rand*(P_B- P_W)),S_{max}) bzw. max(int(rand*(P_B- P_W)),-S_{max}) im ersten Fall und analog im zweiten Fall mit dem generell besten Frosch P_G. Sind beide Kreuzungen schlechter, so wird ein zufallsgenerierter Frosch erzeugt.

Der Zwei-Punkt-**Crossover** (Kapitel 2.8.2.1) wählt als ersten Punkt t_1 eine Zufallszahl zwischen 0,25*PL (Projektlänge) und 0,75*PL. Für den zweiten Punkt wird das minimalste Ressourcenauslastungstal VRUR zwischen Punkt 1 und PL gesucht (L = PL-t_1), wobei jeweils die Aktivitäten aus der Liste A_t herangezogen werden, die sich in diesem Bereich befinden. Der schlechte Frosch bestimmt die Aktivitäten $0 \leq i \leq q_1 \cup q_2 \leq i \leq J+1$, der Beste den zwischenliegenden Bereich. Die Positionen q_1 und q_2 werden aus der erweiterten Aktivitätenliste errechnet. q_1 wird durch die größte Position der Endzeitpunktliste, deren Wert kleiner ist als t_1, und q_2, durch die kleinste Position der Startzeitpunktliste deren Wert größer oder gleich ist als $t_1 + T$, bestimmt.

$$VRUR = \min_{0 \leq t_1 \leq PL-L]} \left\{ \frac{1}{K} \sum_{k=1}^{K} \sum_{t=t_1}^{t_1+T} \frac{\sum_{j \in A_t} r_{jk}}{R_k} \right\}$$

Formel 24: Minimale Ressourcenauslastung (Fang, et al., 2012 S. 894)

Die **Mutation** verwendet einen lokalen Suchalgorithmus (Kapitel 2.8.2.1), der einen Mutationsschritt nur dann manifestiert, wenn die Projektlänge dadurch verbessert wird. Abschließend wird der Frosch noch mit der Double Justification verbessert. Da bestehende Frösche durch neue Frösche ersetzt werden, bleibt die Populationsgröße

[125] Mexeplexes [Engl.]

immer konstant. Nach jeder Generation werden alle Frösche wieder in eine Gesamtpopulation integriert und wie oben bereits beschrieben wieder aufgeteilt. [126]

2.8.3. Tabu Suche[127]

Als Teil der lokalen Suchmethoden, versucht die Tabu Suchmethode in der Nachbarschaft einer existierenden besten Lösung eine bessere zu finden. Dabei wird geprüft, unter Zuhilfenahme einer Tabuliste, die bestimmte Lösungen aus der Nachbarschaft ausschließt, ob ein globales Optimum möglich ist, das nicht zwangsläufig erreicht werden muss.

Tabulisten sind nach dem FIFO[128] Prinzip organisiert und durch eine bestimmte Länge definiert, deren begrenzte Wirkung den Speicherverbrauch positiv beeinflusst und die Rechenzeit verkürzt, da die jeweils optimale Lösung der neuen Nachbarschaft mit der Tabuliste verglichen werden muss. Weitere Verbesserungen in diesem Bereich können erreicht werden, wenn nur Veränderungen der Lösung gespeichert werden und nicht die Lösung selbst. Die Länge einer Tabuliste muss nicht statisch sein und kann während der Lösungssuche dynamisch verändert werden. Die Struktur der Inhalte selbst kann auch einer Veränderung im Laufe der Iterationen unterliegen, allerdings mit der Bedingung, dass die Strukturinformation in der Liste mitgespeichert wird (z.B. Strukturnummer, Iterationsnummer, usw.). Die Tabuliste verhindert nicht nur die Aufnahme von bereits gefundenen Lösungen. Diese können auch dazu verwendet werden, dass bestimmte Attribute oder Schritte, die zu einer bestimmen Lösung führen, eine Lösung als unzulässig kennzeichnen. Die Tabuliste dient nur als Träger der Kennzeichen, kennt aber selbst nicht die Algorithmen der Lösungssuche.

Die Methode ist dadurch gekennzeichnet, dass es zusätzlich zu der Tabuliste, deren Länge die Qualität der Lösung sehr stark beeinflusst, notwendig ist, eine **Nachbarschaftslösung** aus einer **Anfangslösung** zu generieren. Aus dieser wird, unter Berücksichtigung der verbotenen Lösungen der Tabuliste, wiederum eine neue Basis für die nächste Nachbarschaftsgeneration geschaffen. Dies wird solange wiederholt, bis das **Abbruchkriterium** erfüllt ist.

---

[126] (Fang, et al., 2012 S. 890ff)
[127] (Zäpfel, et al., 2010 S. 101ff)
[128] FIFO: First In First Out [Engl.]

Für die Generierung der Initiallösung, wird wie bei Simulated Annealing (Kapitel 2.8.4) mit Hilfe eines SGS, idealweise mit einer Prioritätsregel verbunden, ein Projektplan erstellt. Die Nachbarschaftssuche kann zum Beispiel mit der β - Biased Random Sampling Methode (siehe Kapitel 2.8.4) erfolgen. Weitere Methoden verwenden in der Nachbarschaftssuche Gruppeninformationen von Aktivitäten (es gibt zusammengehörige Aktivitäten innerhalb einer Gruppe), die die Verschiebung einer Aktivität zulassen[129]

Tabulisten im RCPSP zeichnen meist die vorangegangen Schritte auf, die zu der neuen Lösung geführt haben. Neuere Studien unterscheiden zwischen Kurzzeitlisten und Langzeitlisten. Die Abbruchbedingungen sehen entweder eine Maximalgrenze bei den Berechnungsschritten (z.B. Anzahl Projekte) oder eine bestimmte Anzahl verbesserter Lösungen.[130]

2.8.4. Simulated Annealing

Diese Methode startet mit einer **Initiallösung**, die üblicherweise im Falle des RCPSP aus einem SGS, kombiniert mit einer Prioritätsregel, besteht[131], die mit Hilfe einer lokalen Suchmethode verbessert wird. Dabei sucht der Algorithmus ausgehend von einer Anfangstemperatur in iterativen Schritten eine bessere Lösung. Die neue Lösung wird in der Nachbarschaft der aktuell besten Lösung gesucht.[132]

Ein Beispiel für solch eine Methode für die Suche in der **Nachbarschaft** einer Population, ist die **β - Biased Random Sampling Methode**, in der die Variable β den Grad der Veränderung der Lösung bestimmt. Die Ausgangsbasis für diese Methode wird durch die Aktivitätsliste eines Projektplans repräsentiert. Bei der Anwendung des S-SGS wird zum Zeitpunkt der Auswahl einer Aktivität aus der Entscheidungsliste eine reelle zufallsgenerierte Zahl $p \in (0,1)$ erstellt. Ist p kleiner gleich der Variable β, so wird die Aktivität mit der geringsten Anordnung in der Aktivitätsliste ausgewählt. Andernfalls wird die Biased Random Sampling Methode mit der Positionsnummern

[129] (Nonobe, et al., 1999 S. 11)
[130] (Improved Tabu Search Algorithm Application in RCPSP, 2009 S. 4)
[131] (Kolisch, et al., 2000 S. 397), (Kolisch, et al., 2006 S. 383)
[132] (Eglese, 1990 S. 272)

Prioritätsregel angewendet.[133] Diese Methode wird auch in dem Algorithmus von Valls, et al. (2008) in Phase 2 zur Bestimmung der Nachbarschaftspopulation verwendet.

Ist die Lösung eines Iterationsschritts besser als die globale Lösung, so wird diese dadurch ersetzt und als Basis für die weitere lokale Suche verwendet. Ist die neue Lösung schlechter, so wird diese einer Qualitätsprüfung unterworfen, die abhängig vom Fortschritt des Algorithmus berechnet und im positiven Fall akzeptiert wird. Je länger die Suche dauert, desto größer wird die Wahrscheinlichkeit, dass die schlechte Lösung nicht akzeptiert wird. Die Temperaturänderung der Lösung, die fortschreitend abnimmt, gibt der Methode den Namen. Die Akzeptanz von schlechteren Lösungen soll dazu beitragen, dass man nicht in einem lokales Optimum verweilt.[134]

Die Implementation von Bouleimen und Lecocq (2003) generiert die Initallösung ebenfalls mit einem S-SGS und einer Proiritätsregel. Es wird zuerst eine Aktivitätsliste, die alle Vorgängerbeziehungen berücksichtig, erstellt und auf dessen Basis frühest möglich und unter der Berücksichtigung der Ressourcenbedingungen eingeplant. Die Nachbarschaftsgeneration wird dadurch gebildet, dass für eine zufallsgenerierte Aktivität der späteste Vorgänger und der früheste Nachfolger gesucht werden und die Aktivität wiederum zufallsgeneriert innerhalb dieses Intervalls platziert wird. Die restlichen Aktivitäten des Intervalls, die nach der neuen und vor der alten Position auftreten, werden um eine Stelle nach rechts verschoben. Durch diese Vorgehensweise kann eine globale Linksverschiebung der Aktivität erfolgen. Eine detailierte Beschreibung kann dem verwiesenen Dokument entnommen werden.[135]

2.8.5. Ameisensysteme

Die erste Implementation von Ameisensystemen für das RCPSP wurde erstmals von Merkle et al. (2002) durchgeführt.

Das Ameisensystem repräsentiert eine Population von Agenten, ein Objekt, das versucht gewisse Verhaltensmuster einer Ameise nachzubilden. Die Agenten sind durch ein autokatalytisches Verhalten, in dem die Agenten bevorzugt die Wege einschlagen, die bereits von der Großzahl der anderen Agenten benutzt worden sind,

[133] (Valls, et al., 2003a S. 291)
[134] (Eglese, 1990 S. 272)
[135] (Bouleimen, et al., 2003 S. 268ff)

gekennzeichnet. Pheromone bestimmen dieses kollektive Verhalten und durch die Benutzung des Weges werden weitere Pheromone durch den Agenten hinzugefügt.[136]

In dem Algorithmus von Merkle et al. (2002) repräsentiert das S-SGS einen Agenten. Gewichtet mit der LST Prioritätsregel werden die Pheromone für die Auswahl der Aktivitäten der Entscheidungsliste verwendet. Durch die Pheromone wird der S-SGS um einen Lerneffekt erweitert. Es wird versucht, die jeweils beste Lösung einer Generation durch paarweises tauschen der Aktivitäten zu verbessern. Um ein Annähern an die beste generelle Lösung zu vermeiden, wird, bestimmt durch eine geringe Wahrscheinlichkeit, die generell beste Lösung durch die beste Lösung einer Generation ersetzt.[137]

Für weiterführende Informationen zu neueren Lösungen für das RCPSP mit Ameisensystemen sei auf Xiao et al. (2012) und Thiruvady et al. (2012, kombiniert mit Strahlensuche und Constraint Programming) verwiesen.

2.8.6. Weitere Lösungsansätze

Neben den klassischen Metaheuristiken gibt es noch Ansätze der **lokalen Suchmethoden,** die mit blockweisem Verschieben von Aktivitäten arbeiten, in Teilprojektplänen durch lokale Suchmethoden Ressourcen optimieren oder in der Nachbarschaft von existierenden Projektplänen nach weiteren Lösungen suchen. Das Vorgehen in den **populations-basierenden Ansätzen** kann als GA interpretiert werden, wobei diese meist auf die Mutation beschränkt sind. Auch eine Kombination klassischer Metaheuristiken kann dieser Gruppe zugeordnet werden.[138]

Neue Ansätze bietet auch die **Partikel Schwarm Optimierung (PSO),** die auf dem natürlichen Schwarmverhalten von Fischen und Vögeln basiert. Jedes Mitglied eines Schwarms wird durch die global beste Position und seine eigene beste Position gesteuert und durch eine zufällige Komponente. Yang et al. (2011) zeigen mit einer vereinfachten, auch beschleunigt genannten, PSO, welche nur globale Positionen betrachtet und für die Diversitität der Lösung den Zufallsfaktor verwendet, einer

[136] (Dorigo, et al., 1996 S. 2f)
[137] (Merkle, et al., 2002 S. 333ff)
[138] (Kolisch, et al., 2006 S. 5f)

Support Vector Machine, die durch nicht lineare Transformationen die Dimension der Daten transformiert (Strukturen aus Daten schaffen) eine Anwendung auf das RCPSP. Sie erreichen damit für die J30 Instanz mit 5000 Projektplänen eine durchschnittliche Abweichung von der optimalen Lösung von 0,025 (Vergleiche Tabelle 10 in Kapitel 2.8.1) mit einer Rechenzeit von 15,4s. Messwerte weiterer PSPLIB Instanzen sind in diesem Artikel leider nicht vorhanden.

2.9. Testdaten - PSPLIB[139]

Um vorhandene Lösungsmethoden des RCPSP vergleichen und neue Ansätze in Relation dazu stellen zu können wurde die „Project Scheduling Problem Library“ (PSPLIB) entwickelt. Bis dahin verfügbare Testdaten-Bibliotheken waren meist nur für eigene Zwecke generiert oder nicht gezielt parametrisierbar. Der Schwierigkeitsgrad der Probleme ließ eine schnelle Lösung zu und teilweise waren sie nur für das SMRCPSP konzipiert oder die Projektlaufzeitminimierung. Diese umfangreiche Bibliothek (1216 Probleminstanzen) wurde mit dem Projektgenerator ProGen erzeugt.

2.9.1. Project Scheduling Problem Instance Generator

Das Programm „Project Scheduling Problem Instance Generator“ (ProGen) steht auf der PSPLIB Homepage zum Download[140] zur Verfügung. Das Programm ist in Borland Turbo Pascal 6.0 implementiert. Eine genaue Beschreibung der Programmschnittstelle und des Exportformats ist im Anhang von Kolisch et al. (1992) zu finden.

2.9.2. Standard-Sets

Für die Generierung der Standard-Sets wurden drei Haupteinflussfaktoren gewählt. Durch die Variation der Netzwerkkomplexität (NC), die die durchschnittliche Anzahl von nicht-redundanten Pfeilen einer Aktivität wiederspiegelt, des Ressourcenfaktors (RF), welcher den durchschnittlichen Bedarf an Ressourcen pro Aktivität angibt (RF=1: alle Aktivitäten benötigen Ressourcen, RF=0: keine Aktivität benötigt Ressourcen) und der Ressourcenstärke (RS), welche den Anteil des Ressourcenverbrauchs der Aktivitäten an der gegebenen Ressource wiedergibt, können verschiedenste Ausprägungen für ein Standard-Set erzeugt werden.[141]

Im Bereich des SMRCPSP gibt es vier Standard-Sets, deren Namen die Anzahl der Aktivitäten (ohne Quelle und Senke) beinhaltet: J30, J60, J90, J120

Hierbei gibt es jeweils vier erneuerbare Ressourcen und keine nicht-erneuerbare Ressourcen. Die Werte für NC, RF und RS für die Generierung der J30-J120 SMPCPSP

[139] (Kolisch, et al., 2012b)
[140] (Kolisch, et al., 2012a)
[141] (Kolisch , et al., 1992 S. 8ff)

sind in Tabelle 11 und Tabelle 12 ersichtlich. Dabei wurden jeweils NC * RF * RS * 10 Probleminstanzen gebildet.

Parameter	Level 1	Level 2	Level 3	Level 4
NC	1,5	1,8	2,1	
RF	0,25	0,5	0,75	1
RS	0,2	0,5	0,7	1

Tabelle 11: Parameter für das J30, J60 und J90 SMRCPSP (Kolisch, et al., 1999 S. 206)

Parameter	Level 1	Level 2	Level 3	Level 4	Level 5
NC	1,5	1,8	2,1		
RF	0,25	0,5	0,75	1	
RS	0,1	0,2	0,3	0,4	0,5

Tabelle 12: Parameter für das J120 SMRCPSP (Kolisch, et al., 1999 S. 206)

Es gibt zwei verschiedene Inputdatei-Formate für die Berechnung des SMRCPSP. Die Erweiterung .RCP repräsentiert ein Textfile, in dessen erster Zeile die Anzahl Aktivitäten und erneuerbarer Ressourcen zu finden ist. Die zweite Zeile gibt die verfügbaren Ressourcen an. Danach sind die Aktivitäten zeilenweise aufgeführt, wobei die erste Spalte die Dauer der Aktivität repräsentiert, danach der Verbrauch der vier erneuerbaren Ressourcen folgt und schließlich die Anzahl der Nachfolger und deren Aktivitätsnummern. Alternativ kann auch das .SM File verwendet werden, welches zusätzlich zu den Werten aus dem .RCP File, welche mit Spaltenbezeichnungen tabellenorganisiert angeführt sind, weitere Informationen anbietet. Dazu zählen unter anderem eine Information bezüglich der Parameter bei der Generierung des Testsets und weitere Projektinformationen.

Der Aufbau des Dateinamens lautet wie folgt:

J<PARAMETER>_<REPLIKATION>.SM (.RCP)

Der Parameter ist eine fortlaufende Nummer, aus der die Generierungsparameter NC, RF und RS berechnet werden können. In den Inputdateien selbst, sind diese Parameter nicht angeführt, daher müssen diese für eine Auswertung aus dem Dateinamen errechnet werden. Dazu sind die Parameter in der oben angeführten Reihenfolge zu

fixieren und werden in absteigender Reihenfolge variiert.[142] D.h. beispielsweise beinhaltet der Parameter 1 {NC=1,5;RF=0,25;RS=0,2} , der Parameter 2 {NC=1,5; RF=0,25;RS=0,25} usw.

Für das J30 Standard-Set gibt es .OPT Dateien, die die optimale Lösung des SMRCPSP beinhalten. Die Ergebnisse der besten Heuristiken sind in den .HRS Dateien zu finden. Die J60-120 Standard-Sets beinhalten keine .OPT Dateien. [143]

Tests mit der Anzahl Aktivitäten, Anzahl Ressourcen, NC, RF und RS als Steuerparameter haben gezeigt, dass die Rechenzeit mit der Anzahl Aktivitäten und Ressourcen steigt, eine steigende RS die Rechenzeit verbessert und die Rechenzeit mit RF positiv korreliert. Die NC Variable hat keinen signifikanten Einfluss auf die Rechenzeit gezeigt[144]. Spätere Untersuchungen von verschiedenen (Meta)-Heuristiken haben bei der durchschnittlichen Abweichung des Ergebnisses von der optimalen Lösung des RCPSP analoge Resultate der Parameter wie bei der Rechenzeit ergeben[145], die jedoch in späterer Folge nicht mehr Gegenstand der Untersuchung waren.[146] Die Untersuchungen von Dorndorf et al. (2000) zeigen ebenfalls die gleiche Korrelation. Valls et al. zeigen zudem mit ihrem hybriden genetischen Algorithmus, dass eine Kombination aus niedrigen RS (0,1 und 0,2) und hohem RF (0,75 und 1,0) zu einer noch größeren Abweichung des Ergebnisses führen. Hier wurde jedoch nicht die Abweichung von der optimalen Lösung berücksichtigt, sondern die Abweichung der Gesamtprojektdauer des RCPSP von der Projektdauer ohne Ressourcenbedingungen. Es wurden jedoch bezüglich der kombinierten Auswirkung von niedrigen RS und hohem RF auf die Rechenzeiten keine Dokumentation durchgeführt, doch eine isolierte Betrachtung von RS und RF geht mit den Ergebnissen von Kolisch und Hartmann (2000) einher.[147]

Die Standard-Sets für MMPCPSP gibt es ebenfalls in einer Standardausführung, bei der aufgrund des Namens auf die Anzahl der Aktivitäten geschlossen werden kann: J10, J12, J14, J16, J18, J20, J30.

[142] (Dorndorf, et al., 2000 S. 433)
[143] (Kolisch, et al., 2012b)
[144] (Kolisch, et al., 1996 S. 212)
[145] (Kolisch, et al., 2000 S. 404)
[146] (Kolisch, et al., 2006 S. 10ff)
[147] (Valls, et al., 2008 S. 503)

Diese Sets an Aktivitäten erhalten jeweils zwei erneuerbare und zwei nicht-erneuerbare Ressourcen, wobei es sich um vier unterschiedliche Ressourcen handelt. Zusätzlich sind noch weitere Variationen verfügbar (J16 Instanzen), deren genaue Parametrisierung Kolisch et al. (1992) zu entnehmen ist.

3. Anwendung von genetischen Algorithmen für die Optimierung von ressourcenbeschränkten Netzplänen

Als eine Methode aus dem Bereich der Metaheuristik, bieten genetische Algorithmen die Möglichkeit, Lösungen für ressourcenbeschränkte Netzpläne zu finden. In diesem Teil der Diplomarbeit wird der hybride genetische Algorithmus von Valls et al.(2008), in der Folge kurz HGA genannt, analysiert und auf dessen Basis versucht, mittels Variationen der Inputparameter eine Optimierung der Zielfunktion zu erreichen bzw. werden die Auswirkungen auf die Rechenzeit beobachtet. Als Zielfunktion wird die Minimierung der Gesamtdurchlaufzeit des Netzplans festgelegt.

3.1. Der hybride genetische Algorithmus von Valls et al.[148]

Die Zielsetzung des HGA besteht darin, die Zielfunktion von standardisierten Sets von Projekten[149], wobei immer eine definierte Obergrenze von generierten Netzplänen vorliegt, mit einer möglichst geringen Rechenzeit, zu optimieren.

Grundsätzlich ist der HGA in zwei Phasen unterteilt. Die Anzahl generierter Netzpläne ist in beiden Phasen ident und beträgt exakt die Hälfte der Gesamtanzahl.

In der ersten Phase des Algorithmus wird eine Anfangspopulation erstellt. Diese bildet die Basis für die nächste Generation, welche mithilfe des „Peak Crossover“ Operator und anschließender Mutation gebildet wird. Die Väter der weiteren Generationen werden durch die jeweils verbliebenen besten Lösungen repräsentiert. Zusammen mit zufallsgenerierten Müttern werden die weiteren Generationen gebildet.

Die zweite Phase verwendet das beste Ergebnis der Phase Eins, um in dessen Nachbarschaft nach weiteren Lösungen zu suchen. Hierbei wird wieder eine Anfangspopulation (in diesem Fall Nachbarschaftspopulation genannt) erzeugt und anschließend wieder dem Evolutionsprozess wie in Phase Eins unterworfen.

Der „Double Justification“ Operator wird innerhalb des HGAs sehr umfangreich verwendet. Ein Drittel aller generierten Netzpläne wird durch den Algorithmus selbst erstellt und die restlichen Netzpläne durch den „Double Justification“ Operator. Dies

[148] (Valls, et al., 2008 S. 495ff)
[149] (Kolisch, et al., 2012b)

gilt in beiden Phasen sowohl für die Anfangs- bzw. Nachbarschaftspopulation als auch für die Evolutionsphasen. Durch den Einsatz dieses Operators kann eine beträchtliche Lösungsverbesserung, bei gleichzeitig niedriger Rechenzeit, erreicht werden.[150]

Die besonderen Charakteristika des HGA werden somit durch den „Peak Crossover" Operator, die Zwei-Phasen Strategie, die Verbesserung durch den „Double Justification" Operator und die Elternauswahl festgelegt. Abbildung 30 zeigt die einzelnen Schritte der beiden Phasen.

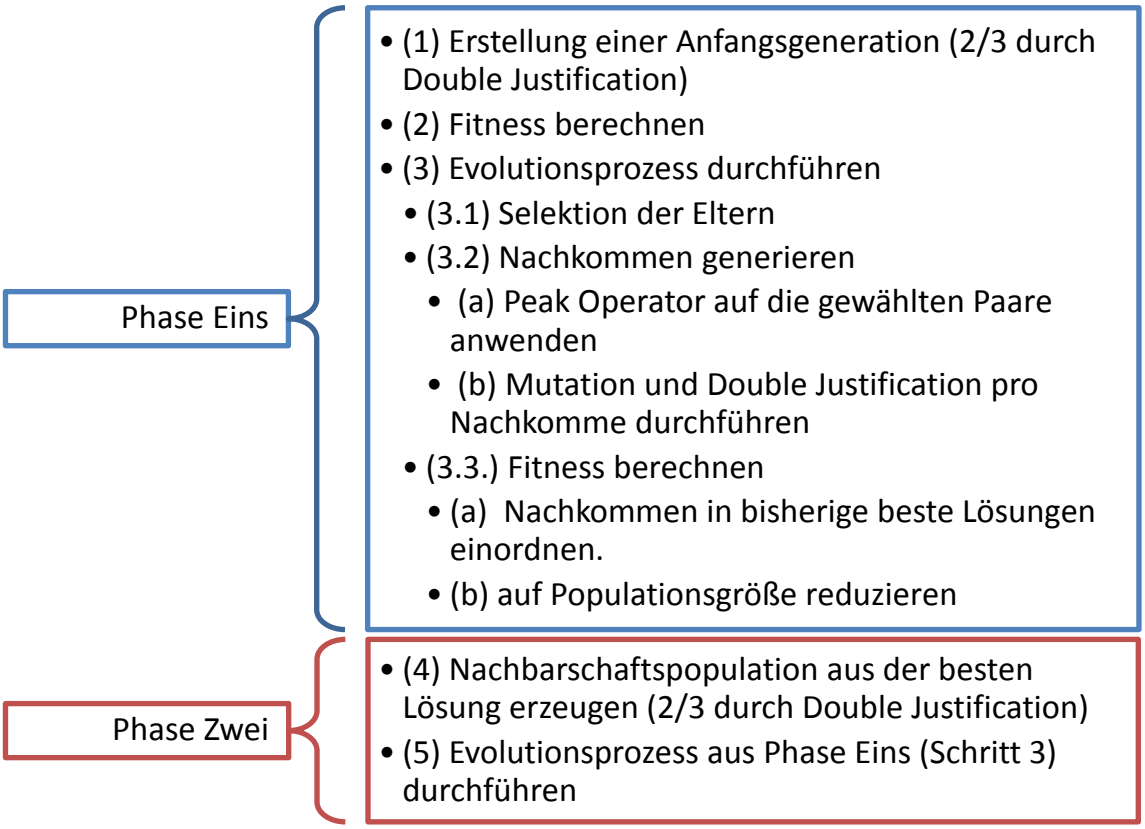

Abbildung 30: Das Modell des HGA (Valls, et al., 2008 S. 497)

3.1.1. Definitionen

Für die Erklärungen des GA beziehen sich alle Ausführungen auf das Beispielprojekt 3 in Abbildung 31.

[150] (Valls, et al., 2005 S. 378ff)

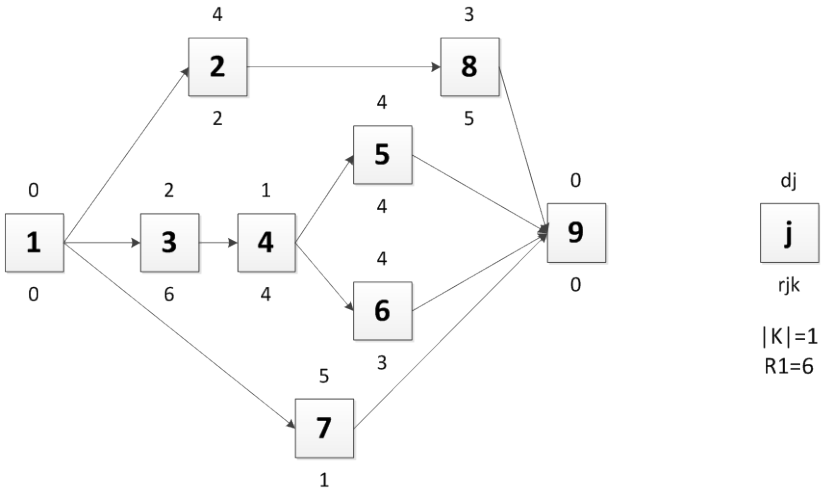

Abbildung 31: Beispielprojekt 3 (Valls, et al., 2008 S. 496)

Die Anwendung eines S-SGS (min. LFT) für die Initialpopulation führt zu einem gültigen Projektplan, der in Abbildung 32 beispielhaft dargestellt ist und als Grundlage für die Erklärung des Peak-Crossover-Operator (Kapitel 3.1.2.3.) verwendet wird.

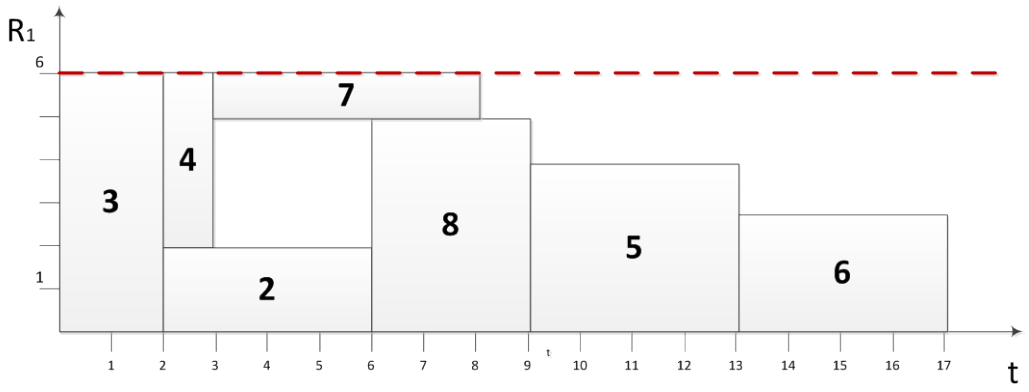

Abbildung 32: Projektplan für das Beispielprojekt 3 (Valls, et al., 2008 S. 496)

Folgende Variablen und Abkürzungen werden bei der Erklärung des Algorithmus verwendet:

S … Gültiger aktiver Projektplan

λ … Aktivitätsliste

V = {1,..,n} … Anzahl Aktivitäten

s_i, f_i … Start- bzw. Endzeitpunkt einer Aktivität

LFT … Spätester Endzeitpunkt (Latest Finish Time)

R-BRS … Regret Based Biased Random Sampling

t … Zeitpunkt in einem Projektplan

ε … Konstante für die R-BRS Berechnung (Positivitätskonstante)

α … Konstante für die R-BRS Berechnung (Exponent)

δ … Untere Grenze für eine Ressourcen-Spitze (Prozent)

$R_{PEAK\ LOWER}$ … Untere Grenze für eine Ressourcen-Spitze (Absolut)

RUR (t) … Durchschnittlicher Ressourcenverbrauch zum Zeitpunkt t

Scheduled(t) … Geplante Aktivitäten zum Zeitpunkt t

I … Zeitraum einer Ressourcenspitze

P … Liste der Aktivitäten einer Ressourcenspitze I

PEAKS … Menge aller P

$p_{mutation}$ … Wahrscheinlichkeit mit der die Mutation ausgeführt werden soll

β … Grad der Veränderung bei der Generierung der Nachbarschaftspopulation

POPsize … Populationsgröße

π … Faktor für die Beeinflussung der Anzahl Kinder einer Generation

SIZE … Größe der aktuellen Population (Phase 1 oder Phase 2)

3.1.2. Phase Eins

3.1.2.1. Erstellung der Anfangsgeneration

Die Anfangsgeneration wird zu einem Drittel durch einen S-SGS erstellt und auf diese Netzpläne wird der Double Justification (siehe Kapitel 2.7) Prozess angewendet. Dadurch wird die generierte Anfangspopulation verdreifacht.

Die Generierung von einem Drittel der **Initialpopulation** basiert auf dem Permutations-GA von Hartmann (Kapitel 2.8.2.1). Hier kommt die LFT Prioritätsregel, basierend auf Kolisch (1996b), zum Einsatz, mit dem Ziel, jeweils die Aktivität mit dem minimalsten Prioritätswert aus der Entscheidungsliste zu selektieren. Die Prioritätsregel ist hierzu in die Regret Based Biased Random Sampling (R-BRS) Methode nach Drexl (1991) integriert, wobei abweichend von Drexl, der den minimalsten Prioritätswert der Entscheidungsliste als Konstante ε für die Positivitätsgarantie des errechneten Wahrscheinlichkeitswertes verwendet ($\varepsilon = \min_{j \in D_j} v(j)$) [151], die Konstante ε mit dem gleichen Wert wie der Parameter α angenommen wird (α=1).

[151] (Kolisch, et al., 1999 S. 157)

Die Berechnung mit den beschriebenen Paramteren ist in Formel 25 bis Formel 27 dargestellt.

$$v'(j) = \max_{j \in D_j} v(i) - v(j)$$

Formel 25: Initialpopulation Valls: minimalsten Prioritätswert aufgrund der Zielfunktion auswählen

$$v''(j) = (v'(j) + 1)^1$$

Formel 26 Initialpopulation Valls: Positivität der Wahrscheinlichkeit und Gewichtung des Wertes (α=ε=1)

$$p(j) = \frac{v''(j)}{\sum_{i \in D_n} v''(i)}$$

Formel 27: Initialpopulation Valls: Wahrscheinlichkeit bestimmen

Anschließend wird jede Aktivität in Reihenfolge der Aktivitätsliste so früh als möglich eingeplant.[152]

3.1.2.2. Selektion

Basierend auf der Populationsgröße SIZE (die in Phase 1 und 2 verschieden ist) und dem Parameter π, der für jede PSPLIB Instanz, abhängig von der Anzahl zu erzeugender Projektpläne (Tab ….), variiert, generiert der Algorithmus π * SIZE/2 Paare PA. Für die Auswahl wird in PA Iterationen jeweils das beste Individuum und ein zufallsgenerierter Partner selektiert und das Paar aus der Selektionsmenge entfernt, damit jedes Individuum nur einmal verwendet werden kann. Nach dem Crossover und Mutationsprozess werden die darin generierten Kinder der Population hinzugefügt und anschließend wird die Population auf die definierte POPsize mittels der Rangmethode (Kapitel 2.8.2.1) reduziert. [153] Der Algorithmus beinhaltet keine explizite Behandlung von Aktivitäts-Klonen, es wird jedoch von eindeutigen Aktivitätslisten ausgegangen.[154] Um diese These zu verfizieren, werden später Tests durchgeführt, welche Klone zulassen.

[152] (Valls, et al., 2008 S. 499)
[153] (Valls, et al., 2008 S. 499f)
[154] (Valls, et al., 2008 S. 498)

3.1.2.3. Crossover

Der Crossover-Prozess ist von der Ressourcenauslastung der Aktivitäten abhängig und dient auch als Basis für Implementationen in anderen genetischen Algorithmen (Debels und Vanhoucke (2007), Fang und Wang (2012)). Da diese Methode die Maximalwerte der Ressoucenauslastung für den Crossover in Betracht zieht, wird sie **Peak-Crossover** genannt.

Ressourcenauslastung

Mit einem Kapazitätsauslastungsdiagramm (Abbildung 33, Beispiel mit einer Ressource) kann die Ausnutzung der Ressource visualisieren werden und die Spitzen und Täler sind erkennbar. Das Ziel eines optimalen Projektplans stellt die maximale Ausnutzung der Ressourcen dar. Für die Klassifizierung einer Kapazitätsspitze muss eine Grenze gezogen werden, deren Überschreitung eine Ressourcenausnutzung als eine Spitze qualiziert. Diese Schranke wird mit der Variable δ festgelegt. Bei Betrachtung von Abbildung 33, welche die Ressourcenauslastung eines gültigen Projektplans (Abbildung 32) wiedergibt, werden durch die Festlegung von δ=0,8 ($R_{PEAK\ LOWER}$ = 6 * 0,8 = 4,6) zwei Ressourcenspitzen P_1 = {3,2,4} im Bereich [0,3] und P_2 = {7,8} im Intervall [6,9] festgestellt. Die Reihenfolge der Aktivitäten innerhalb der Ressourcenspitzen wird durch die permutierte Aktivitätsliste des Projektplans S bestimmt. Ein teilweises Vorkommen der Aktivität im Zeitbereich der Ressourcenspitze ist aussreichend um ein Teil davon zu werden. Die Reihenfolge der Spitzen selbst wird durch ihr Vorkommen auf der Zeitachse festgelegt.

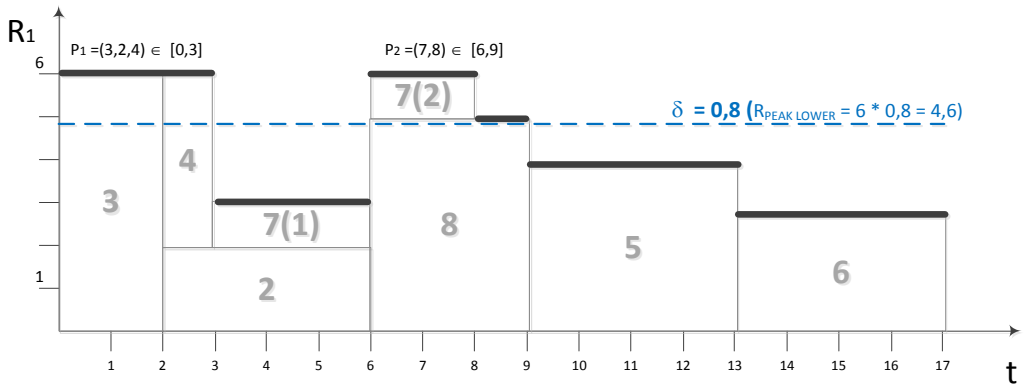

Abbildung 33: Ressourcennutzung (Valls, et al., 2008 S. 496)

Für die Berechnung einer Ressourcenauslastung RUR(t) werden die Ressourcenbelegungen aller, zu diesem Zeitpunkt t geplanten, Aktivitäten $S(A_t)$ herangezogen, ihre Auslastung pro Ressource errechnet und anschließend ein Durchschnitt der Summen aller Ressourcenbelegungen ermittelt.

$$Scheduled(t) = \{i \in V; [t-1, t] \cap [s_i, f_i] \neq 0\}$$

$$RUR(t) = \frac{1}{K} \sum_{j \in Scheduled(t)} \sum_{k=1}^{K} \frac{r_{jk}}{R_k} \qquad 0 \leq RUR(t) \leq 1$$

Formel 28: GA Valls: Durchschnittliche Ressourcenauslastung (Valls, et al., 2008 S. 496)

Sofern nun $RUR(t) \geq \delta$ erfüllt ist, wird ein Zeitpunkt t als Ressourcenspitze definiert. Ein Zeitintervall $I = [I_s, I_f]$ ($I_{s<} I_f$) stellt diese Zeitpunkte der Ressourcenspitzen in einem Projektplan $S(\lambda)$ dar. Alle Aktivitäten der Liste λ die nun in diesem Zeitintervall I geplant sind, werden nun in Reihenfolge der Liste λ Teil der Liste P. Die untere Grenze δ für die Ressourcenspitze des Crossoverprozesses varriert zufallsgeneriert zwischen zwei Grenzen δ_{min} und δ_{max}, die als Ergebnis umfangreicher Tests mit δ_{min}=0,75 und δ_{max}=0,9 festgelegt wurden.

Bildung einer neuen Generation

Bei der Bildung eines Sohns λ^S verwendet der Crossover-Prozess die Aktivitätenliste des Vaters λ^V und der Mutter λ^M. Der Projektplan S^V stellt die Grundlage für die Berechnung der Ressourcenspitzen dar. Die Aktivitäten der Liste λ^V die Teil der Liste P^V sind werden in die Liste λ^S aufgenommen und die Anordnung der restlichen Aktivitäten wird aus der Mutter λ^M übernommen. Bei der Bildung der Tochter λ^T gilt eine analoge Vorgehensweise, wo jedoch die Rollen von Vater und Mutter vertauscht sind.

Die wechselnden Rollen der Eltern in dem Algorithmus können durch die Bezeichnung bestimmendes (Vater bei der Generation eines Sohnes, Mutter bei der Generation einer Tochter) und nichtbestimmendes Elternteil (Mutter bei der Generation eines Sohnes, Vater bei der Generation einer Tochter) allgemein formuliert werden. Bei der Ausführung dieses Prozesses findet eine iterative Abarbeitung aller Ressourcenspitzen statt. Es werden, bis in der Liste des bestimmenden Elternteils die erste Spitze erreicht wird, alle Positionen des Kindes mit den Aktivitäten des nicht bestimmenden

Elternteils aufgefüllt. Dabei ist immer zu prüfen (ISCANDIDATE), ob die Aktivität nicht Teil einer anderen Spitze ist oder bereits Teil der Aktivitätenliste des Kindes ist.

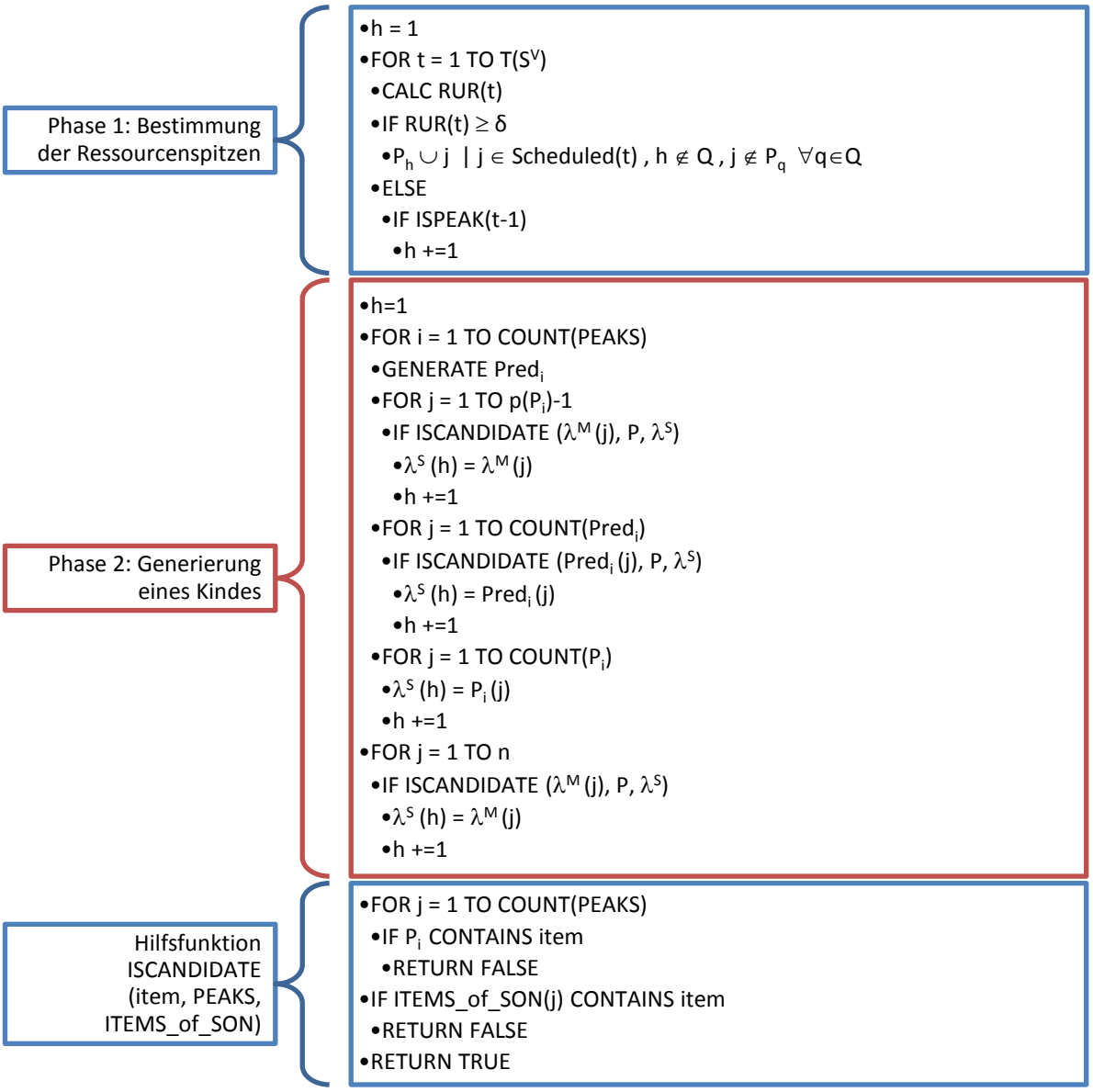

Abbildung 34: GA Valls: Crossover-Programmablauf (Valls, et al., 2008 S. 499)

Danach werden alle Vorgänger der Aktivitäten der Ressourcenspitzenliste der Liste des Kindes zugewiesen. Dabei ist zu beachten, dass die Vorgängeraktivitäten nach dem Vorkommen in der Aktivitätenliste des nicht bestimmenden Elternteils sortiert sind, unter der Berücksichtigung, dass ISCANDIDATE zutrifft. Anschließend werden die Aktivitäten der Ressourcenspitzenliste der Liste des Kindes hinzugefügt.

Sind alle Ressourcenspitzen abgearbeitet, so werden zuletzt die Aktivitäten des nicht bestimmenden Elternteils der Liste des Kindes zugewiesen, wobei wieder ISCANDIDATE zu beachten ist. Eine übersichtliche Darstellung des Programmablaufs ist in Abbildung 34 zu entnehmen.

Es gilt weiters, dass das Auftreten einer Ressourcenspitze nicht zwingend vorkommen muss und in diesem Fall das Kind von dem bestimmenden Elternteil abgeleitet wird. Mit zunehmender Verbesserung der Projektpläne im Laufe des GA ist aber die Wahrscheinlichkeit eines solchen Nicht-Vorkommens sehr gering.[155]

3.1.2.4. Mutation und Double Justification

Die Mutation wird nach der Methode von Hartmann (1998), wie in Kapitel 2.8.2.1 beschrieben, abhängig von einer Mutations-Wahrscheinlichkeit $p_{mutation}$, unter der Berücksichtigung der Vorgängerbeziehungen durch einen paarweisen Tausch der Aktivitäten (Quelle und Senke sind ausgenommen) durchgeführt. Auf den dadurch erhaltenen Projektplan wird abschließend die Double Justification (Kapitel 2.7) Prozedur angewandt.[156]

3.1.3. Phase Zwei

Der hybride GA beinhaltet neben der Mutation noch eine weitere lokale Suche. Die zweite Phase des GA verwendet die beste Aktivitätsliste λ der ersten Phase, um in dessen Nachbarschaft nach einer optimalen Lösung zu suchen.

3.1.3.1. Nachbarschaftspopulation

Die Nachbarschaftspopulation ist vergleichbar mit der Initialpopulation. Es werden allerdings nur halb so viele Projektpläne pro Generation in der zweiten Phase des Algorithmus erzeugt. Wiederum gilt, dass zwei Drittel der Projektpläne durch die Double Justification Methode erzeugt werden.

Auf die Aktivitätsliste λ wird für die Erzeugung von einem Drittel der Nachbarschaftspopulation die **β - Biased Random Sampling Methode**[157] angewandt,

[155] (Valls, et al., 2008 S. 498f)
[156] (Valls, et al., 2008 S. 500)
[157] Siehe Kapitel 2.8.4

in der die Variable β den Grad der Veränderung der Lösung bestimmt und mit β=1-(20/n) festgelegt wird (0 ≤ β ≤ 1). Das verwendete S-SGS bestimmt zum Zeitpunkt der Auswahl einer Aktivität aus der Entscheidungsliste eine reelle zufallsgenerierte Zahl p ∈ (0,1). Unterschreitet p die Variable β, so wird die Aktivität mit der geringsten Anordnung in der Aktivitätsliste ausgewählt. Der Alternativfall verwendet die aus Kolisch und Hartmann (1999) entnommenen Biased Random Sampling Methode, die die Positionsnummer (der besten Lösung) als Prioritätsregel zur Basis hat.[158]

3.1.3.2. Selektion, Crossover und Mutation

Auch die zweite Phase des Algorithmus ist einer genetischen Veränderung unterworfen. Kapitel 3.1.2.1 bis 3.1.2.4 sind sinngemäß wie in Phase Eins zu verwenden, mit der Ergänzung, dass die Initialpopulation durch die neue Nachbarschaftspopulation zu ersetzen ist und die Populationsgröße POPsize, aus der sich auch die Anzahl Elternpaare für die Selektion zusammensetzt, die halbe Größe aufweist. [159]

3.1.4. Parametrisierung

Bei den Berechnung von Valls, Ballestin und Quintanilla (2008) wurde auf Basis der PSPLIB Instanzen J30, J60, J90 und J120 folgende Konfigurationen für die Populationsgröße POPsize und den Fakor π für die Bildung der Generationen vorgenommen.

Die Testergebnisse sind in zwei Rubriken geteilt. Zuerst wurde eine Parametrisierung der Standardtestfälle (Erzeugung von maximalen 1.000, 5.000 und 50.000 Projektplänen) unterstützt durch Testläufe, die mit der jeweils ersten Replikation der 48 bzw. 60 Problemfälle der Testinstanzen durchgeführt wurde, ermittelt. Die Standardtestfälle dienen der Vergleichbarkeit mit anderen Algorithmen

[158] (Valls, et al., 2008 S. 500)
[159] (Valls, et al., 2008 S. 497)

	1.000		5.000		50.000	
	POP_{size}	π	POP_{size}	π	POP_{size}	π
J30	50	1	100	0,8	1000	1
J60	50	0,2	50	0,7	300	1
J90	50	0,4	50	0,9	300	1
J120	24	0,4	50	0,4	400	0,4

Tabelle 13: Parametrisierung J30-J120 (Valls, et al., 2008 S. 501)

Da die Instanzen J30 bis J90 einfachere Probleme darstellen und 2008 bereits 100%, 74% bzw. 73% der optimalen Lösungen bekannt waren, wurden mit diesen Instanzen keine weiteren Untersuchungen vorgenommen. Im Gegensatz dazu stellt die J120 Instanz mit einer Optimalitätsrate von 35% ein noch sehr weites Feld für Verbesserungen dar. Daher wurden für diese Instanz weitere Konfigurationen für 2.500, 10.000, 25.000 und 100.000 Projektplänen festgelegt und getestet (Tabelle 14).

	2.500		10.000		25.000		100.000	
	POP_{size}	π	POP_{size}	π	POP_{size}	π	POP_{size}	π
J120	24	0,6	100	0,4	200	0,4	400	0,9

Tabelle 14: Parametrisierung J120-EXT (Valls, et al., 2008 S. 501)

Die Grenzen für den Crossover-Operator und die Wahrscheinlichkeit der Mutation sind in Tabelle 15 ersichtlich. Es wurden ebenfalls Versuche mit einem fixen Crossover-Operator unternommen, die allerdings schlechtere Ergebnisse lieferten.

Variablen	Wert
δ_{min}	0,75
δ_{max}	0,90
$p_{mutation}$	0,05

Tabelle 15: Crossover- und Mutationsparameter (Valls, et al., 2008 S. 501)

3.2. Implementierung des HGA von Valls et al.

Der in Kapitel 3.1 beschriebene hybride genetische Algorithmus wurde für weitere Analysen nachgebildet. Dazu wurde die Programmiersprache MS Visual Basic 11 (2012 RC [160]) auf Basis des .NET Frameworks 4.0 verwendet. Es wurde in der Entwicklungsumgebung MS Visual Studio 11.0 RC implementiert und als Projekttyp eine Konsolenapplikation gewählt, da diese im Gegensatz zu einer formularbasierenden Lösung, einen geringeren Ressourcenverbrauch aufweist. Die Informations-Ausgabe erfolgt nach jeder Testinstanz in einer Konsole sowie in einem Logfile, das jedes Problem der Testinstanz kommasepariert dokumentiert (CSV Datei). Dadurch kann eine spätere Detailanalyse mit MS Excel durchgeführt werden. Fehlermeldungen werden in der Konsole ausgegeben und in die Logdatei geschrieben.

Das Programm bietet nach dem Start verschiedene Modi an. Es können alle Testkonfigurationen durchlaufen werden ([A] All tests) oder einzeln durchgeführt werden. Die Beschreibung der Konfigurationen und dessen Programmmodus-Kennzeichen ist in Kapitel 3.3 vorzufinden.

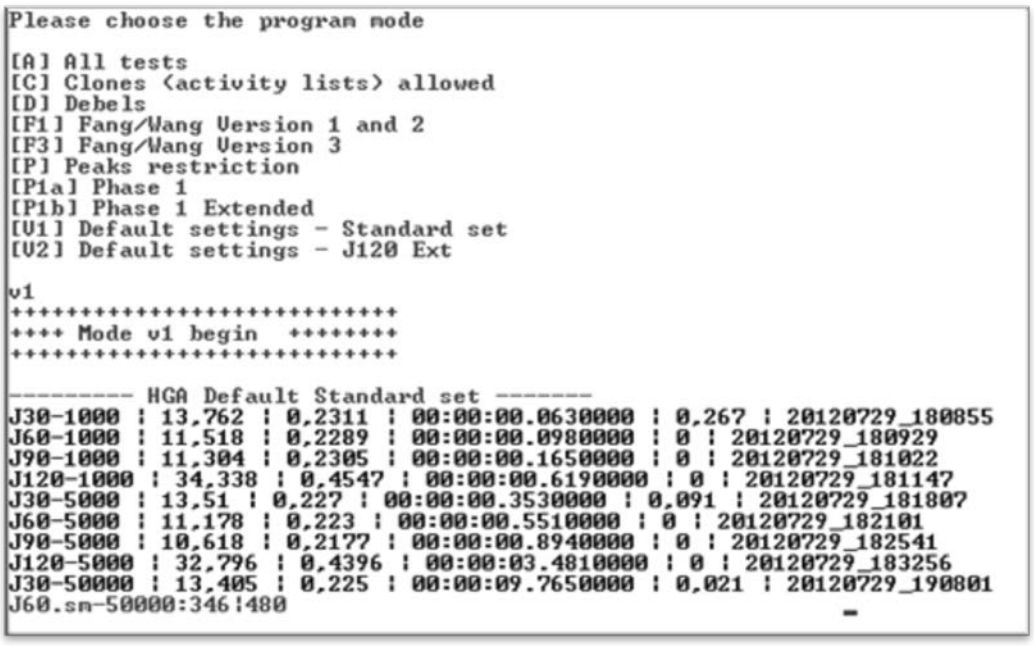

Abbildung 35: Bildschirmkopie der Applikation

[160] Release Candidate

Die Ausgabe in der Konsole erfolgt pro Testinstanz in zwei Schritten. Nach der Beendigung des Teilproblems (z.b. J30-1000) wird die Fortschrittsanzeige durch das Ergebnis überschrieben:

Schritt 1: Fortschrittsanzeige

Schritt 2: Ergebnis (J<Instanz>-<max. Projekte>: <Abweichung CP in %> | <Standardabweichung CP> |<durchschnittliche Rechenzeit> | <Abweichung von der optimalen Lösung, wenn bekannt> | <Referenz auf das Logfileverzeichnis>

3.2.1. Design

Das folgende Kapitel stellt den Aufbau der Implementierung des HGA dar. In Abbildung 36 ist eine Übersicht des Programmes dargestellt, eingeteilt in die zwei Phasen des HGA mit einer weiteren Untergliederung in Initialpopulation und Crossover.

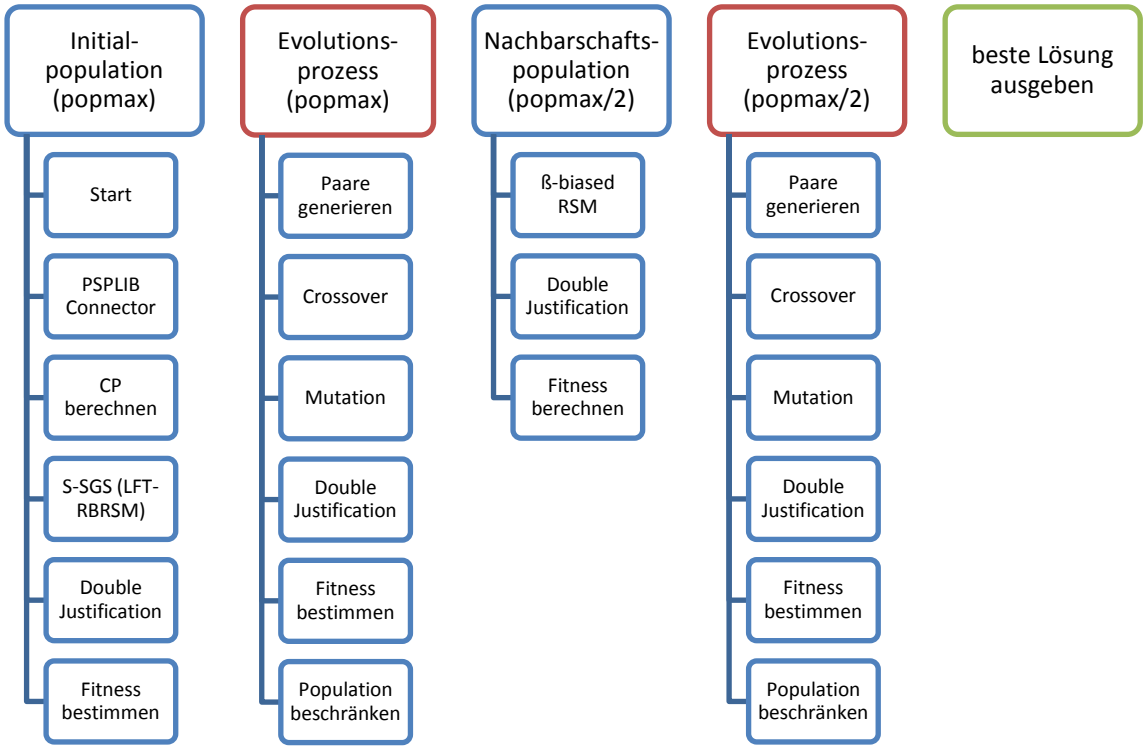

Abbildung 36: Übersicht des Programmablaufes

Der diesem Projekt zugrundenliegende Quellcode und das kompilierte Programm mit der Version **2012.Q3.1.5** kann frei unter http://rcpsp.codeplex.com/ bezogen werden.

Die Klasse *clsTaskLib* ist der Ausgangspunkt der Applikation und koordiniert alle Phasen des Algorithmus. Eine Übersicht über alle Objekte kann folgender Übersicht entnommen werden.

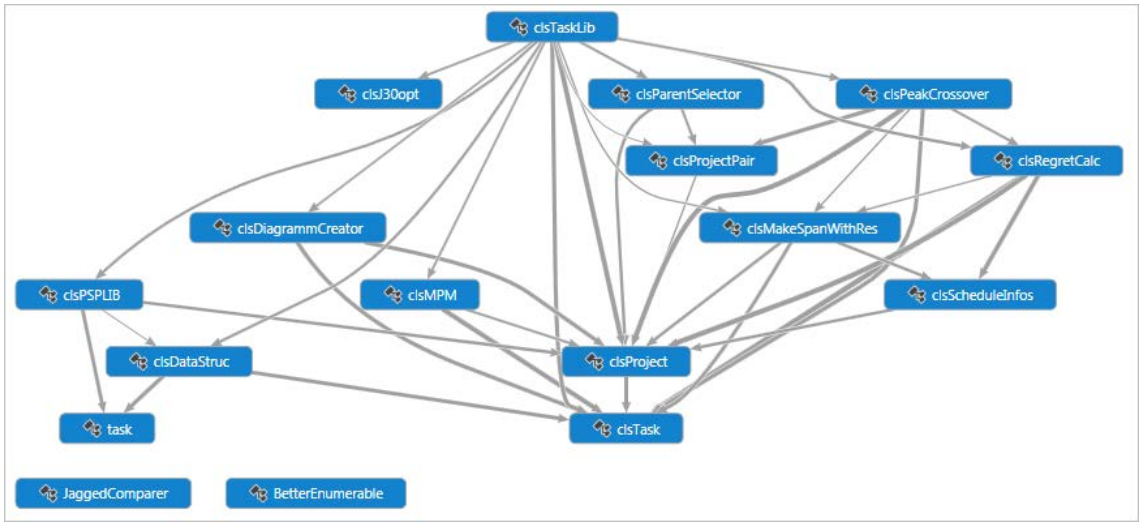

Abbildung 37: Objekte der Applikation

Zudem wird hier auch der Vergleich zur optimalen Lösung vorgenommen und optional die Erstellung eines Kapazitätsdiagramms vorgenommen (*clsDiagrammCreator*). Der Netzplan ist hierzu als Objekt *clsProject* abgebildet, der alle relevanten Informationen des Projektes enthält.

Objekt	**Beschreibung**
Peaks As Int32	Anzahl gefundener Ressourcenspitzen
Generation As Int32	Information wo der Projektplan erzeugt wurde (Initialpopulation, Kindergeneration der Initialpopulation, Nachbarschaft, usw.)
TasksA As List(Of clsTask)	Informationen zu allen Aktivitäten
CriticalPath As New HashSet(Of String)	Kritischer Pfad des Projektplans ohne Ressourcenbeschränkungen
RenewableResources As Dictionary(Of String, Int32)	Erneuerbare Ressourcen des Projekts
Fitness As Int32	Bewertung (=Dauer) des Projekts
ActivtyList As HashSet(Of Int32)	Aktivitätslisten-Repräsentation
PeakResource()() As Int32	Kapazitätsauslastung des Projektes
Diagramm()()() As Int32	Optional (da zeitintensiv): Informationen um ein Kapazitätsbelastungsdiagramm zu erzeugen (Zeit-Ressourcen-Projektplan Vektor)

Tabelle 16: Basiselemente des Aktivitätsobjektes

3.2.1.1. PSPLIB Connector

Als Inputdateien wurden die in Kapitel 2.9 beschriebenen Dateien der PSPLIB Testdatenbank verwendet. Aus den beiden verfügbaren Dateiformaten .SM bzw. .RCP wurde .SM gewählt, da diese mehr Projektinformationen bieten. Damit die Applikation bei Bedarf auch andere Testdatenformate einlesen kann, so wurde für die Verarbeitung in dem Programm eine Zwischenschicht eingeführt. Die Daten werden mit der Klasse *clsPSPLIB* eingelesen und in ein typed DataSet konvertiert, dessen Schema in Abbildung 38 dargestellt ist. Da ein DataSet einfach in ein XML Format speicherbar ist, so kann die Anlieferung auch in diesem Format durchgeführt werden.

Abbildung 38: Schema für Inputdateien auf XML Basis

Mit der Klasse *clsDataStruc* wird diese Datenstruktur eingelesen und für das Hauptprogramm aufgearbeitet. Hierbei wird für jede Aktivität *clsTask* ein Objekt erstellt, das wie folgt aufgebaut ist.

Objekt	Beschreibung
Key As Int32	Nummer der Aktivität
Duration As Int32	Dauer der Aktivität
Pre As HashSet(Of Int32)	Direkte Vorgänger der Aktivität
AggPre As HashSet(Of Int32)	Alle Vorgänger der Aktivität
Post As HashSet(Of Int32)	Direkte Nachfolger der Aktivität
RenewableResources As Dictionary(Of String, Int32)	Ressourcenverbrauch der Aktivität für k Ressourcen

Tabelle 17: Basiselemente des Aktivitätsobjektes

Die Felder Key, Duration, Pre, Post und RenewableResources müssen angeliefert werden, AggPre wird selbst berechnet. Die Aktivitäten werden anschließend der Taskliste *TasksA* (List (of Tasks)) auf dem Objekt *clsProject* hinzugefügt.

3.2.1.2. Kritischer Weg und Startzeiten

Der erste Schritt nach dem Einlesen der Basisdaten besteht darin, den kritischen Weg des Netzplans ohne Ressourcenbeschränkungen zu bestimmen und dabei alle möglichen Start-, End- und Pufferzeiten zu bestimmen. Diese dienen später als Basis für den Einsatz von Prioritätsregeln. Die Berechnung erfolgt in der Klasse *clsMPM*. Durch diesen Vorgang werden die Zeitobjekte Objekte der Klasse *clsTask* befüllt (Tabelle 18Tabelle 18: Zeitobjekte des Aktivitätsobjektes)

Objekte	Beschreibung
FAZ As Int32	FAZ der Aktivität
FEZ As Int32	FEZ der Aktivität
SAZ As Int32	SAZ der Aktivität
SEZ As Int32	SEZ der Aktivität
Puffer As Int32	Puffer der Aktivität

Tabelle 18: Zeitobjekte des Aktivitätsobjektes

Für interne Zwecke ist eine Kopie dieser Zeiten mit dem Prefix MPM (z.b. MPMFAZ) vorhanden.

3.2.1.3. Anfangspopulationen

Die Generierung der Anfangspopulation (Phase 1) und der Nachbarschaftspopulation (Phase 2) wurde in einer gemeinsamen Funktion der Klasse *clsRegretCalc* implementiert.

Diese Funktion produziert eine Aktivitätsliste mit dessen Information die Aktivitäten eingeplant werden. Solange noch Aktivitäten in der Liste fehlen, werden die Aktivitäten bestimmt, deren Vorgänger bereits Teil der Aktivitätsliste sind, aber selbst noch kein Teil dieser Liste sind. Im Falle der **Anfangspopulation** werden die SEZ (LFT) Werte v(i) für die Prioritätswertbestimmung herangezogen, das Maximum daraus bestimmt, von jedem Wert abgezogen (v‘(i)) und dann von dem verteilten Wert v‘‘(j) die Auswahlwahrscheinlichkeit berechnet. Die **Nachbarschaftspopulation** bekommt zusätzlich eine geordnete Aktivitätsliste übergeben, die als Basis für die β - Biased Random Sampling Methode verwendet wird. Sofern eine Zufallszahl geringer als der β Wert ist, wird die Ordnung der Aktivitätsliste beibehalten (Aktivität geringster Ordnung

zuerst in die Liste aufnehmen), andernfalls mit der Biased Random Sampling (Positionsnummer der geordneten Aktivitätsliste) ein S-SGS durchgeführt.

In der Klasse *clsMakeSpanWithRes* wird, basierend auf der Aktivitätsliste, der ehestmögliche **Einplanvorgang** durchgeführt. Hier wird zeitorientiert geprüft, ob die Ressourcen an dieser Stelle noch frei sind. Andernfalls wird die Aktivität so weit nach rechts verschoben, bis ein Platz gefunden wurde. Vorgängerbeziehungen sind bereits durch die Aktivitätsliste gegeben.

Dabei wird das Aktivitätsobjekt wieder um eine neue Information erweitert:

Objekt	Beschreibung
PlanedAt Int32	Nummer der Aktivität

Tabelle 19: Planobjekte des Aktivitätsobjektes

Die Double Justification findet ebenfalls auf der Klasse *clsRegretCalc* ihre Anwendung. Das Konzept ist ähnlich dem Einplanvorgang gestaltet, allerdings ist die Basis die nach Endzeitpunkten (Rechtsanordnung) bzw. Startzeitpunkten (Linksanordnung) absteigend bzw. aufsteigend geordnete Liste des Ausgangsprojektplans.

3.2.1.4. Paargenerierung

Die Paargenerierung in *clsParentSelector* geht nach der in Kapitel 3.1.2.2 beschriebenen Methode vor und kann gleichermaßen in Phase Eins und Zwei verwendet werden. Es wird lediglich die halbe Anzahl Paare in der zweiten Phase verwendet.

3.2.1.5. Crossover und Mutation

Die in Kapitel 3.1.2.3 ausführlich beschriebene Crossover-Methode ist in der Klasse *clsPeakCrossover* abgebildet. Der Crossover Prozess ist für Vater und Mutter gleichermaßen gestaltet, mit jeweils wechselnden Rollen (bestimmendes und nicht bestimmendes Elternteil). Die Grenze für die Ressourcenspitzen wird zufallsgeneriert in einem vordefinierten Bereich [0,75 – 0,9] bestimmt. Danach wird der Algorithmus aus Abbildung 34 angewandt. Die Mutation ist ein einfacher Algorithmus, der bei Eintreten der Mutations-Wahrscheinlichkeit Element x mit Element x+1 vertauscht

(Quelle und Senke ausgenommen) und deren Vorgängerbeziehungen vor dem Tausch prüft. Die Double Justification Methode wird wie in der Anfangspopulation aufgerufen.

3.2.1.6. Kapazitätsbelastungsdiagramm

Für Dokumentationszwecke kann pro gelöstem Problem ein Kapazitätsbelastungsdiagramm für jede Ressource erzeugt werden. Hierzu wird an dem Speicherort der Logdatei pro Problem eine HTML-Datei erzeugt. Diese repräsentiert die beste Lösung des Problems und visualisiert durch ein Kapazitätsbelastungsdiagramm die Einplanzeiten der Aktivitäten und deren Ressourcenverteilung. Weiters ist eine tabellarische Übersicht aller Aktivitäten und deren Planzeiten gegeben mit einer Farb-Legendeninformation der Aktivität.

Key	Duration	Pre	Resource	MPM-FAZ	MPM-FEZ	MPM-SAZ	MPM-SEZ	MPM-Puffer	FAZ	FEZ	SAZ	SEZ	Puffer	PlanedAt
1	0	0|	0|0|0|0|	0	0	0	0	0	0	0	0	0	0	0
2	8	1|	0|0|2|1|	0	8	15	23	15	0	8	15	23	15	0
3	9	1|	1|8|1|6|	0	9	0	9	0	0	9	0	9	0	0
4	1	1|	2|1|4|6|	0	1	39	40	39	0	1	39	40	39	0
5	4	3|	4|6|0|9|	9	13	26	30	17	9	13	26	30	17	9
6	4	2|	7|9|0|0|	8	12	23	27	15	8	12	23	27	15	8
7	8	3|	8|4|7|0|	9	17	9	17	0	9	17	9	17	0	10
8	3	7|	5|6|2|6|	17	20	17	20	0	17	20	17	20	0	18

Abbildung 39: Dokumentation der Aktivitäten

Die Generierung wird standardmäßig nicht durchgeführt, da die Aufzeichnung der Zusatzinformationen die Rechenzeit verdoppelt und der Speicherbedarf pro HTML-Datei ca. 1 MB beträgt, was bei der J120 Instanz einen Dokumentationsumfang von ca. 600 MB bedeutet.

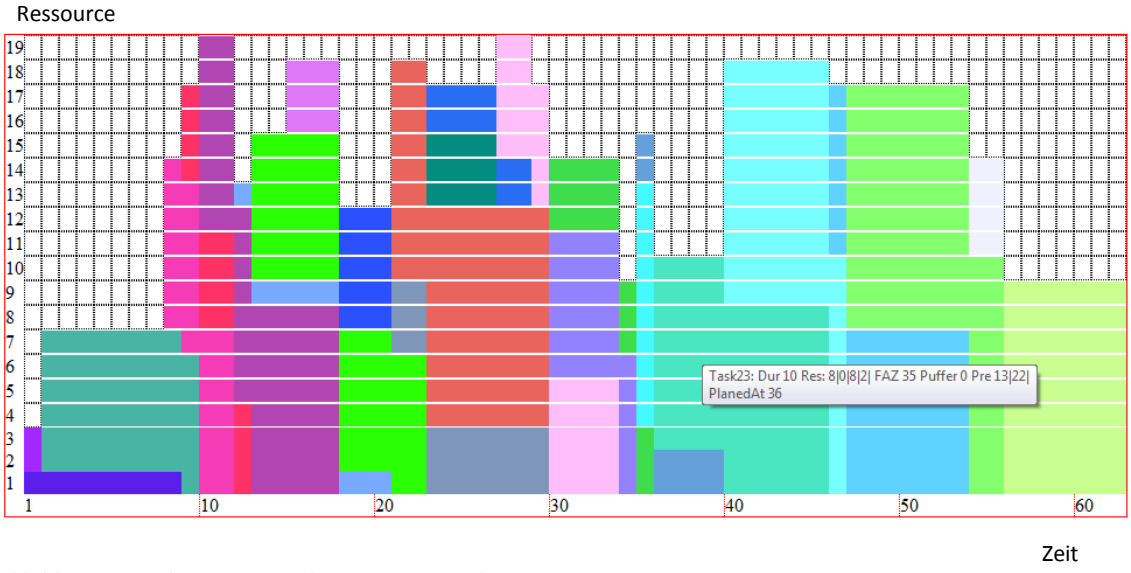

Abbildung 40: Dokumentation der Kapazitätsauslastung

Ein Tooltip zeigt alle relevanten Informationen der fokusierten Aktivität an.

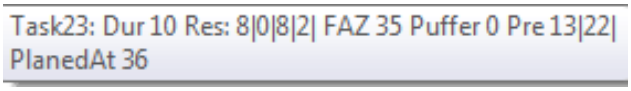

Abbildung 41: Tooltip für die Kapazitätsauslastung

Das Kapazitätsbelastungsdiagramm wird in Form einer Ressourcenauslastung aufgezeichnet. Eine geblockte Darstellung der Aktivitäten ist nicht vorgesehen, ist aber leicht in diese überführbar.

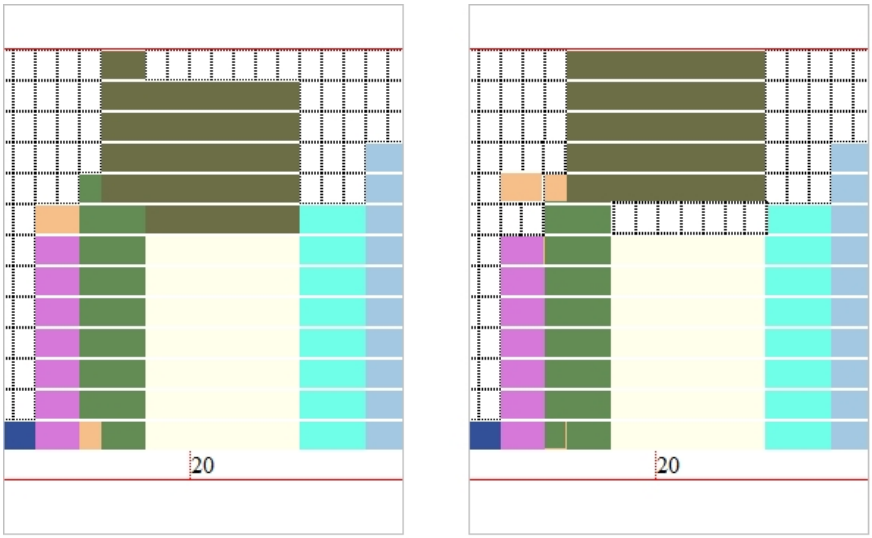

Abbildung 42: Kapazitätsauslastung in geblockter Form

3.3. Analyse

Die Implementierung des hybriden genetischen Algorithmus wurde nach der programmatischen Umsetzung einer umfassenden Analyse unterzogen. Dabei wurden ausgehend von den Testergebnisdaten von Valls, et al. (2008), welche in Kapitel 3.3.1 angeführt sind, eine Verfikation des Programmes durchgeführt. Nachdem sichergestellt war, dass die Ergebnisse des Algorithmus reproduzierbar sind, wurde versucht, mit einigen Änderungen die Applikation zu verbessern. Dazu wurden parallel zum bestehenden Standardset die Möglichkeit vorgesehen, den Algorithmus mit alternativen Einstellungen durchzuführen. Diese Konfigurationen (in Klammer die Bezeichnung des Programmmodus der Applikation) sind:

- Statische Begrenzung der zu ermittelten Ressourcenspitzen (P):
  - J30: 2
  - J60: 4
  - J90: 6
  - J120: 8
- Dynamische Begrenzung der zu ermittelten Ressourcenspitzen: Nur jede zweite Ressourcenspitze wird für den Crossover verwendet (P)
- Nur Phase Eins ausführen (zu Kontrollzwecken)
  - $\pi = 1$, kein Crossover (P1)
  - $\pi = 1$, kein Crossover, keine Mutation (P1)
- Doppelte Aktivitätslisten in der Population zulassen (C)

Die hier verwendete Kapazitätsauslastungsberechung wurde alternativ auch in zwei anderen Algorithmen verwendet: Debels und Vanhoucke (2007) und Fang und Wang (2012). Der Einsatz dieser Methoden wurde an den Crossover Prozess von Valls et al. (2008) angepasst und daher wurden diese modifziert verwendet (Originalversion siehe Kapitel 2.8.2.4 bzw. 2.8.2.5). Durch eine minimale Programmanpassung war es möglich, den Ressourcen-Spitzen Selektor dahingehend zu modifizieren, um auch die beiden anderen Methode zu unterstützen. Die Crossover-Methode selbst wurde nicht verändert, um einen direkten Vergleich zwischen den drei Ressourcen-Spitzen Methoden zu erhalten.

Dadurch sind folgende weitere Konfigurationen entstanden:

- Peak-Operator Debels et al. (D)
- Peak-Operator Fang & Wang
    - Variante 1 (F1):
        - 0,15 * fitness (ft) ≤ Startposition (SP) ≤ 0,35 * ft
        - Suchbereich für das minimale RUR(t):
            - SP ≤ Endposition ≤ (0,65 * ft ≤ SP ≤ 0,75 * ft)
    - Variante 2 (F1):
        - 0,25 * fitness (ft) ≤ Startposition (SP) ≤ 0,65 * ft
        - Suchbereich für das minimale RUR(t):
            - SP ≤ Endposition ≤ (0,75 * ft ≤ SP ≤ 0,85 * ft)
    - Varianten 3 (F3):
        - Suchbereich wie Variante 2
        - Erweiterung auf n Resourcenspitzen
            - n = Anzahl Aktivitäten (ohne Quelle, Senke) / 30
            - (Fitness des Vaters / n) bestimmt die Abstände zwischen den Spitzen
            - Die Länge der Spitzen werden wie in Variante 2 berechnet, allerdings durch n geteilt

Die Versuche wurden zuerst mit einem begrenzten Testset durchgeführt, welches nur die erste Replikation der 48 bzw. 60 Probleminstanzen der J30-J120 Instanzen beinhaltet hat. Dadurch wurden Konfigurationen für die Varianten ermittelt, um damit alle Problemfälle zu berechnen.

In weiterer Folge werden für die Darstellung der Datenanlyse folgende Abkürzungen verwendet:

CP_dev : durchschnittliche Abweichung vom kritischen Pfad (ohne Resourcen-beschränkung), in Klammer jeweils die Standardabweichung.
Time: durchschnittliche Rechenzeit in Sekunden

3.3.1. Ergebnisse Applikation.

Alle Versuche wurden auf Workstations mit einer Intel Core Duo CPU (3 GHz) und 4 GB RAM durchgeführt.

3.3.1.1. Standardeinstellungen

Für die Standardtestfälle (Erzeugung von maximalen 1.000, 5.000 und 50.000 Projektplänen) wurden folgende Referenzwerte der Originalstudie entnommen (fehlende Werte in der Tabelle sind nicht in der Referenzstudie enthalten, J30opt gibt zusätzlich die Abweichung von der optimalen Lösung an).

	1.000		5.000		50.000	
	CP_dev	time	CP_dev	time	CP_dev	time
J30	n.v.	n.v.	13,47 (0,22)	0,31	n.v.	n.v.
J30-opt	0,27	n.v.	0,6 (0,22)	0,31	0,02	n.v.
J60	11,56	n.v.	11,10 (0,22)	0,46	10,73	n.v.
J90	n.v.	n.v.	10,46 (0,21)	0,61	n.v.	n.v.
J120	34,07	n.v.	32,54 (0,44)	2,03	31,24	n.v.

Tabelle 20: Referenzwerte Standardsets (Valls, et al., 2008)

Mit den Standardeinstellungen des HGA Valls et al. (2008) wurden mit der erstellten Applikation für die in Kapitel 3.3.1 angeführten Referenzwerte folgende Ergebnisse erreicht:

	1.000		5.000		50.000	
	CP_dev	time	CP_dev	time	CP_dev	time
J30	13,81 (0,23)	0,055	13,56 (0,23)	0,28	13,41 (0,225)	9,48
J30-opt	0,30 (0,23)	0,055	0,126 (0,23)	0,28	0,026 (0,225)	9,48
J60	11,55 (0,23)	0,089	11,12 (0,22)	0,45	10,76 (0,214)	5,47
J90	11,29(0,229)	0,148	10,58 (0,22)	0,77	10,14 (0,208)	8,78
J120	34,28 (0,45)	0,570	32,86 (0,44)	3,06	31,53 (0,428)	34,4

Tabelle 21: Applikation Standardset: Einstellungen Standard

Weitere Ergebnisse für das Standardset sind in dem nächsten Kapitel vorzufinden.

Der erweiterte J120 Test von Valls et al. (2008) für 2.500, 10.000, 25.000 und 100.000 Projektpläne hat folgende Referenz-Ergebnisse errechnet:

	2.500		10.000		25.000		100.000	
	CP_dev	time	CP_dev	time	CP_dev	time	CP_dev	time
J120	33,18 (0,45)	1,02	32,04 (0,43)	4,01	31,52 (0,43)	10,02	30,95 (0,42)	39,51

Tabelle 22: Referenzwerte J120-EXT (Valls, et al., 2008)

Bei einem eindeutigen Vorkommen der Aktivitätslisten in der Population, werden geringere Abweichungen CP_dev erreicht. Die Klon-Prüfung (Modus C) führt zu keiner signifikanten Verschlechterung der Durchlaufzeit.

	2.500		10.000		25.000		100.000	
	CP_dev	time	CP_dev	time	CP_dev	time	CP_dev	time
J120	33,44 (0,446)	1,5	32,34 (0,435)	6,3	31,84 (0,43)	16,4	31,25 (0,425)	68,6
J120 Clone	33,55 (0,447)	1,44	32,41 (0,435)	6,48	31,92 (0,43)	17,0	31,46 (0,426)	71,7

Tabelle 23: Applikation J120-EXT : Einstellungen Standard (keine Klone)

Für die J120 Instanz mit 5000 Projektplänen wurde analog zu der Studie eine erweiterte Datenerhebung für die Ressourcenstärke RS, den Ressourcenfaktor (RF) und die Netzwerkkomplexität (NC) durchgeführt. Eine Beschreibung dieser Faktoren ist in Kapitel 2.9.2 zu finden.

Die Aussage von Valls et al. (2008) der Nichtbeeinflussung der Resultate durch NC wurde ebenfalls verifiziert und das gleiche Verhältnis der Abweichungen bei J120 (5000) vorgefunden (1,5: 32,04 ; 1,8 : 30,71 ; 2,1: 35,54).

Auch die in Tabelle 24 dargestellte Pivot-Tabelle zeigt die gleichen Zusammenhänge wie in Valls et al. (2008). Zusätzlich zu deren Ergebnissen wurden auch die J120 Instanzen untersucht, die mit 100.000 und 2.500 Projektplänen gerechnet wurden. Dabei hat die Analyse auch die NC in Betracht gezogen. Somit konnte bei beiden Ergebnissen erkannt werden, dass die geringere CP_dev bei NC=1,8 gegenüber NC=2,1 durch RS=0,1 und RS=0,2 bestimmt wird. Bei allen anderen RS Werten ist die CP_dev bei NC=1,8 höher als bei NC=2,1.

CP-dev	RF				
RS/NC	0,25	0,5	0,75	1	Gesamt
0,5	**0,31**	**0,36**	**0,33**	**2,09**	**0,77**
2,1	0,18	0,71	0,64	5,27	1,70
1,8	0,75	0,37	0,35	0,47	0,48
1,5	0,00	0,00	0,00	0,54	0,14
0,4	**1,56**	**3,01**	**7,05**	**11,56**	**5,80**
2,1	1,91	4,86	9,47	13,75	7,50
1,8	1,17	3,14	6,39	10,87	5,40
1,5	1,60	1,04	5,29	10,05	4,50
0,3	**2,13**	**10,16**	**19,95**	**30,25**	**15,62**
2,1	4,72	14,51	21,63	34,69	18,89
1,8	1,19	6,92	21,65	27,56	14,33
1,5	0,50	9,05	16,56	28,51	13,65
0,2	**12,21**	**34,61**	**53,19**	**68,02**	**42,01**
2,1	11,76	35,84	57,80	69,75	43,79
1,8	11,52	34,72	43,83	70,14	**40,05**
1,5	13,34	33,28	57,94	64,17	42,18
0,1	**24,48**	**76,00**	**118,93**	**148,71**	**92,03**
2,1	25,38	76,02	129,35	160,21	97,74
1,8	22,04	71,61	115,64	135,25	**86,13**
1,5	26,02	80,37	111,80	150,66	92,21
Gesamt	**8,14**	**24,83**	**39,89**	**52,13**	**31,25**

Tabelle 24: Applikation: CP-dev J120-100.000

Eine graphische Auswertung von Tabelle 24 ist in Abbildung 43 zu sehen. Im Vergleich zu Abbildung 44 (2.500 Projektpläne gerechnet) lässt sich gut erkennen, dass der Verlauf der Diagramme fast ident ist und das J120-2.500 Diagramm einer Vertikalverschiebung der J120-100.000 entspricht.

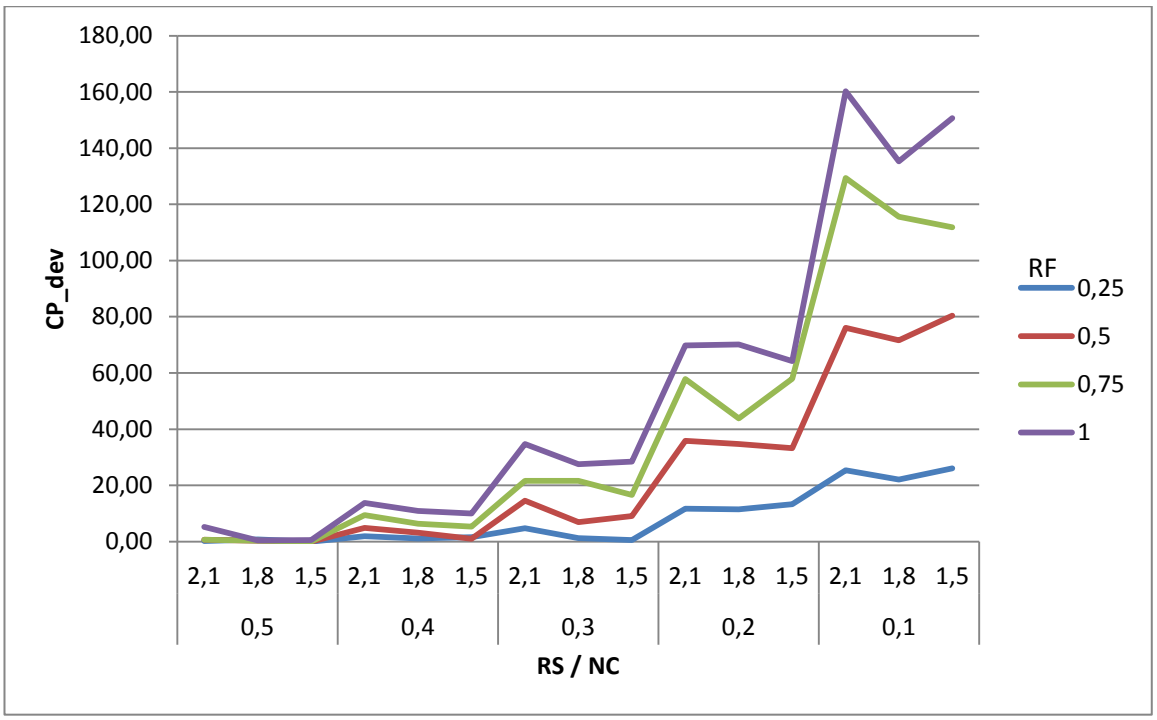

Abbildung 43: Applikation: CP-dev J120-100.000

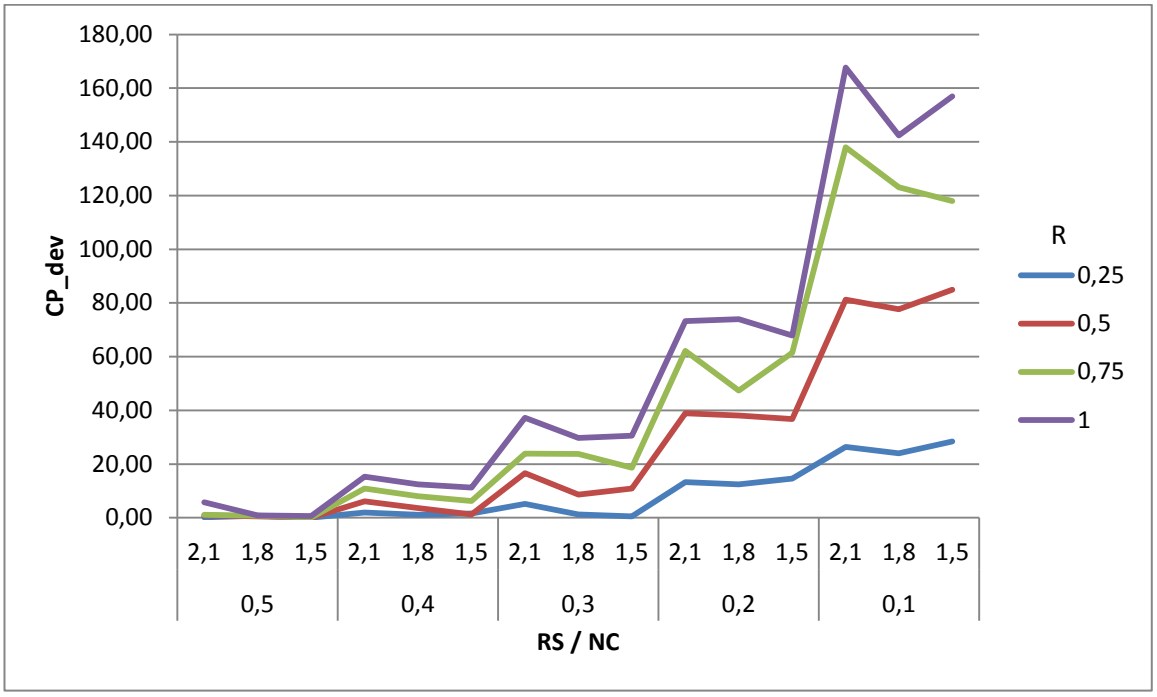

Abbildung 44: Applikation: CP-dev J120-2.500

3.3.1.2. Alternative Konfigurationen

Der Einsatz von alternativen Crossover-Operatoren (Debels, Fang und Wang) zeigt eine minimale Verbesserung der J30 Instanz und eine Reduktion der Rechenzeit, jedoch ist die Begrenzung der Anzahl von Ressourcenspitzen weder dynamisch noch statisch erfolgreich. Eine bessere Diversität der Population durch die Vermeidung von Aktivitätslisten-Klonen ist bei dieser Instanz nicht gegeben. Anhand der Kontrollergebnisse Phase 1 C und CM können die Mutations- bzw. Crossover-Effekte des GA erkannt werden.

J30	1.000			5.000		
	CP_dev	+ / -	time	CP_dev	+ / -	time
Standard	13,82		0,06	13,56		0,31
Peak-S	13,91	0,09	0,05	13,61	0,05	0,27
Peak-D	13,93	0,12	0,06	13,58	0,02	0,31
Clones	13,84	0,03	0,06	13,59	0,03	0,36
Phase 1 C	14,61	0,80	0,05	14,04	0,48	0,26
Phase 1 CM	15,40	1,59	0,05	14,97	1,41	0,22
Peak Debels	13,82	0,00	0,06	13,55	**-0,01**	0,31
Peak FW V1	13,90	0,08	0,05	13,57	0,01	0,26
Peak FW V2	13,79	**-0,03**	0,05	13,53	**-0,03**	0,27
Peak FW V3	13,78	**-0,03**	0,05	13,53	**-0,03**	0,23

Tabelle 25: Applikation Standardset: J30 Alternative Einst.

J30 opt	1.000			5.000		
	CP_dev	+ / -	time	CP_dev	+ / -	time
Standard	0,31		0,06	0,13		0,31
Peak-S	0,37	0,06	0,05	0,16	0,03	0,27
Peak-D	0,39	0,09	0,06	0,14	0,01	0,31
Clones	0,32	0,02	0,06	0,15	0,02	0,36
Phase 1 C	0,84	0,54	0,05	0,45	0,33	0,26
Phase 1 CM	1,41	1,11	0,05	1,09	0,97	0,22
Peak Debels	0,30	**0,00**	0,06	0,11	**-0,01**	0,31
Peak FW V1	0,36	0,05	0,05	0,13	0,00	0,26
Peak FW V2	0,28	**-0,03**	0,05	0,10	**-0,03**	0,27
Peak FW V3	0,27	**-0,03**	0,05	0,10	**-0,02**	0,23

Tabelle 26: Applikation Standardset: J30opt Alternative Einst.

Die beste Alternativkonfiguration wurde auch mit 50.000 Projektplänen getestet. Hierbei konnte keine Verbesserung bei CP_dev festgestellt werden.

J30	50.000		
	CP_dev	+ / -	time
Standard	13,41		9,5
Standard Opt	0,03		9,5
Peak FW V3	13,46	0,05	2,6
Peak FW V3 Opt	0,06	0,03	2,6

J120	50.000		
	CP_dev	+ / -	time
Standard	31,53		34,4
Peak FW V3	32,44	0,91	18,1

Tabelle 27: Applikation J30/J120-50000 : Einstellungen Fang/Wang V3

Die alternativen Konfigurationen der Instanzen J60 und J90 bieten keine Verbesserungsmöglichkeiten in der Abweichung CP_dev. Die Alternativkonfiguration Fang/Wang Version 3 besitzt die größte Nähe zu dem HGA mit Standardeinstellungen, in Bezug auf die Abweichung CP_dev, und benötigt weniger Rechenzeit (44 Prozent).

J60	1.000			5.000		
	CP_dev	+ / -	time	CP_dev	+ / -	time
Standard	11,55		0,09	11,12		0,45
Peak-S	11,70	0,15	0,07	11,24	0,12	0,37
Peak-D	11,66	0,12	0,08	11,13	0,01	0,44
Clones	11,64	0,09	0,09	11,27	0,15	0,46
Phase 1 C	12,93	1,38	0,08	12,25	1,13	0,36
Phase 1 CM	13,88	2,33	0,06	13,84	2,72	0,27
Peak Debels	11,77	0,22	0,09	11,31	0,19	0,43
Peak FW V1	11,85	0,30	0,07	11,34	0,22	0,35
Peak FW V2	11,78	0,24	0,07	11,23	0,11	0,34
Peak FW V3	11,70	0,16	0,07	11,17	0,05	0,31

Tabelle 28: Applikation Standardset: J60 Alternative Einst.

J90	1.000			5.000		
	CP_dev	+ / -	time	CP_dev	+ / -	time
Standard	11,29		0,15	10,58		0,75
Peak-S	11,44	0,14	0,11	10,73	0,14	0,51
Peak-D	11,55	0,26	0,13	10,61	0,03	0,63
Clones	11,32	0,03	0,15	10,69	0,11	0,75
Phase 1 C	12,39	1,09	0,1	11,70	1,12	0,47
Phase 1 CM	13,23	1,94	0,08	13,22	2,63	0,35
Peak Debels	11,49	0,19	0,13	10,76	0,18	0,55
Peak FW V1	11,61	0,32	0,1	10,81	0,23	0,46
Peak FW V2	11,53	0,23	0,11	10,73	0,15	0,45
Peak FW V3	11,58	0,29	0,1	10,66	0,07	0,42

Tabelle 29: Applikation Standardset: J90 Alternative Einst.

Die schwierigste Testinstanz J120 zeigt die Leistungsfähigkeit des Peak Crossover-Operators. Keine Alternativkonfiguration ist in der Lage ein besseres Ergebnis zu errechnen, allerdings ist dadurch eine höhere Rechenleistung erforderlich.

J120	1.000			5.000		
	CP_dev	+ / -	time	CP_dev	+ / -	time
Standard	34,29		0,57	32,84		2,97
Peak-S	34,80	0,52	0,32	33,23	0,39	1,59
Peak-D	34,79	0,50	0,41	33,18	0,34	2,01
Clones	34,33	0,04	0,56	32,97	0,13	2,98
Phase 1 C	37,71	3,42	0,3	36,01	3,17	1,54
Phase 1 CM	40,61	6,32	0,23	39,82	6,98	1,16
Peak Debels	35,08	0,80	0,35	33,45	0,61	1,65
Peak FW V1	35,29	1,00	0,27	33,58	0,75	1,47
Peak FW V2	35,01	0,72	0,29	33,33	0,49	1,39
Peak FW V3	34,89	0,60	0,26	33,22	0,39	1,32

Tabelle 30: Applikation Standardset: J120 Alternative Einst.

Zusammenfassend kann festgestellt werden, dass die alternativen Peak Operatoren nur in der J30 Instanz minimale Verbesserungen hervorbringen und die Klon-Ermittlung in allen Instanzen einen Erfolg erkennen lässt. Die Detailergebnisse der Untersuchung ist unter

http://rcpsp.codeplex.com/SourceControl/changeset/view/15389#

abrufbar.

Conclusio

Metaheuristiken können mit einer akzeptablen Rechenzeit gute Näherungslösungen für RCPSP erzeugen. Klassische genetische Algorithmen treten zunehmend in den Hintergrund, deren Grundbausteine in neueren genetischen Algorithmen Anwendung finden. Hartmann (1998) hat durch die Entwicklung eines genetischen Algorithmus für das RCPSP eine sehr gute Grundlage geschaffen. Die Generierung von Initialpopulationen, die Selektion, der Crossover und die Mutation findet abgewandelt in vielen späteren genetischen Algorithmen ihre Anwendung.

Moderne genetische Algorithmen unterscheiden sich vor allem durch die Wahl des Crossover-Operators, der meist die Ressourcenausnutzung eines Projektplans zur Basis hat. Die Wahl der Partner vor der Evolution und eine geeignete Methode zur anschließenden Reduktion auf die Populationsgröße bieten weitere Möglichkeiten für die Optimierung der Zielfunktion des RCPSP. Lokale Suchmethoden wie die Links- oder Rechtsausrichtung (Double Justification) eines Projektes sind längst Standardmethoden des GA geworden, auf dessen Verbesserungspotential nicht verzichtet werden kann. Eindeutige Repräsentationen in der Population tragen zu einer besseren Diversität der Population bei und bieten die Möglichkeit, dass Raum für andere Schemata bleibt, welche die Lösung verbessern können.

Das Lösen von besonders schwierigen Probleminstanzen, die einen hohen Ressourcenfaktor und eine geringe Ressourcenstärke aufweisen, ist nach wie vor eine große Herausforderung und zeigt Abweichungen von der ressourcenlosen Projektlänge von über hundert Prozent. Diese Probleminstanzen sind sehr bestimmend für das Gesamtergebnis. Verbesserungen in diesem Bereich haben daher eine große Auswirkung auf die gesamte Performance des Algorithmus.

Die Weiterentwicklung im Bereich der genetischen Algorithmen ist durch die Hinzunahme von Bausteinen aus anderen Methoden gekennzeichnet. Durch diese Erweiterung sind viele hybride genetische Algorithmen mit sehr vielen Performance-Verbesserungsmöglichkeiten entstanden. Der gezielte Einsatz von intelligenten Selektions- und Evolutionsverfahren kann hier weitere Fortschritte für die Zukunft bringen.

Literaturverzeichnis

Al-Momani, A. H. 2000. A Linear Programming Algorithm for Least Cost Scheduling. *Journal of the Institution of Engineers.* 2000, 81, pp. 79-82.

Alvarez-Valdes, R. and Tamarit, J. M. 1989. Heuristic Algorithms for Resource-Contrained Project Scheduling. [book auth.] R. Slowinski and J. Weglarz. *Advances in Project Scheduling.* Amsterdam : Elsevier, 1989.

Barták, R. 1999. Constraint Programming: In Pursuit of the Holy Grail . *Proceedings of WDS99* . Prag : s.n., 1999, pp. 555-564.

Blazewicz, J., Lenstra, J.K. and Rinooy Kan, A.H.G. 1983. Scheduling Subject to Resource Constraints: Classification and Complexity. *Discrete Applied Mathematits.* 1983, 5, pp. 11-24.

Bouleimen, K. and Lecocq, H. 2003. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European Journal of Operational Research.* 2003, 149, pp. 268-281.

Bradke, T. 2003. *Grundlagen in Operations Research für Ökonomen.* s.l. : Oldenbourg Verlag, 2003.

Davis, E. W. and Patterson, J.H. 1975. A Comparison of Heuristic and Optimum Solutions in Resource-Constrained Project Scheduling. *Management Sience.* 1975, 17, pp. 944-955.

Debels, D. and Vanhoucke, M. 2007. Decomposition-Based Genetic Algorithm for the Resource-Constrained Project-Scheduling Problem. *Operations Research.* 2007, Vol. 55, 3, pp. 457-469.

Dorigo, M., Maniezzo, V. and Colorni, A. 1996. The ant system: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man and Cybernetics, Part B.* 1996, 26, pp. 29-41.

Dorndorf, U., Pesch, E. and Phan-Huy, T. 2000. A Branch-and-Bound Algorithm for the Resource-Constrained Project Scheduling Problem. *Mathematical Methods of Operations Research.* 2000, 52, pp. 413-439.

Drexl, A. 1991. Scheduling of Project Networks by Job Assignments. *Management Science.* 1991, 37, pp. 1590-1602.

Eglese, R.W. 1990. Simulated Annealing: A tool for Operational Research. *European Journal of Operational Research.* 1990, 46, pp. 271-281.

Fang, C. and Wang, L. 2012. An Effective Shuffled Frog-leaping Algorithm for Resource-Constrained Project Scheduling Problem. *Computers &OperationsResearch.* 2012, 39, pp. 890-901.

Goldberg, D. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning.* s.l. : Addison-Wesley Publishing Company, Inc., 1989.

Hansen, H.R.. and Neumann G. 2001. *Wirtschaftsinformatik I.* 8.Auflage. Tübingen : UTB, 2001. [Home].

Hartmann, S. 1998. A Competitive Genetic Algorithm for Resource-Constrained Project Scheduling. *Naval Reserach Logistics.* 1998, 45, pp. 733-750.

Hartmann, S. and Briskorn, D. 2010. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research.* 2010, 207, pp. 1-14.

Heizer, J. and Render, B. 1990. *Production and Operations Management.* s.l. : Allyn and Bacon, 1990.

Herroelen, W., de Reyck, B. and Demeulemeester, E. 1998. Resource-Constrained Project Scheduling: A Survey of Recent Developments. *Computers and Operation Researchs.* 1998, Vol. 25, 4, pp. S.279-302.

Improved Tabu Search Algorithm Application in RCPSP. **Pan, N., Lee, M. and Chen, K. 2009.** Hong Kong : s.n., 2009. Proceedings of the International MultiConference of Engineers and Computer Scientists.

Kerzner, H. 2009. *Project Management: A Systems Approach to Planning, Scheduling, and Controlling.* s.l. : John Wiley & Sons, 2009.

Kolisch , R., Sprecher, A. and Drexl, A. 1992. *Characterization and Generation of a Gerneral Class of Resource-Constrained Project Scheduling Problems.* Kiel : s.n., 1992. Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel. 301.

Kolisch, R. and Drexl, A. 1996. Adaptive Search for Solving Hard Project Scheduling Problems. *Naval Research Logistics.* 1996, 43, pp. 23-40.

Kolisch, R. and Hartmann, S. 2000. Experimental evaluation of state-of-the-art Heuristics for the Resource-constrained Project scheduling problem. *European Journal of Operational Research (127).* 2000, pp. 394-407.

—. 2006. Experimental Investigation of Heuristics for Resource-Constrained Project Scheduling: An Update. *European Journal of Operational Research (147).* Oktober 2006, pp. 23-37.

—. 1999. Heuristic Algorithms for Solving the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis. [ed.] J. Weglarz. *Project Scheduling, Recent Models, Algorithms and Applications.* Online: http://129.187.106.231/psplib/files/handbook1.pdf : Kluwer Academic Publishers, 1999, pp. 147-178.

Kolisch, R. and Sprecher, A. 2012b. Project Scheduling Problem Library - PSPLIB. [Online] 2012b. [Cited: 06 25, 2012.] http://129.187.106.231/psplib/.

—. 1996. PSPLIB – A project scheduling problem library. *European Journal of Operational Research 96.* 1996, pp. 205-216.

—. 2012a. PSPLIB - ProGen - Generate New Data Sets. *ProGen - Project Scheduling Problem Instance Generator.* [Online] 2012a. [Cited: 06 29, 2012.] http://129.187.106.231/psplib/files/progen-sfx.exe.

Kolisch, R. 1996a. Efficient Priority Rules for the Resource-Constrained Project Scheduling Problem. *Journal of Operiations Management.* 1996a, 14, pp. 179-192.

—. 1995. *Project Scheduling under Rsource Constraints - Efficient Heuristics for Several Problem Classes.* Heidelberg : Springer, 1995.

—. 1996b. Serial and Parallel Resouce-Constrained Project Scheduling Mehtods revisited: Theory and Computation. *European Journal of Operational Research.* 1996b, 90, pp. 320-333.

Kolisch, R., Schwindt, C. and Sprecher, A. 1999. Benchmark Instances for Project Scheduling Problems. *Project Scheduling, Recent Models, Algorithms and Applications.* s.l. : Kluwer Academic Publishers, 1999, pp. 197-212.

Kolisch, R., Sprecher, A. and Drexl, A. 1992. *Characterization and Generation of a General Class of Resource-Constrained Project Scheduling Problems.* Insitut für Betriebswirtschaftslehre, Universität Kiel. Kiel : s.n., 1992. Manuskript. 301.

Li, K.Y. and Willis, R.J. 1992. An Iterative Scheduling Technique for Resource-Constrained Project Scheduling und General Resource Contrained. *European Journal of Operational Research.* 1992, 56, pp. 370-379.

Merkle, D, Middendorf, M. and Schmeck, H. 2002. Ant Colony Optimization for Resource-Constrained Project Scheduling. *IEEE Transactions on Evolutionary Computation.* 2002, 6, pp. 333-346.

Neumann, K. and Morlock, M. 2002. *Operations Research.* 2. Auflage. s.l. : Hanser Verlag, 2002.

Nonobe, K. and Ibaraki, T. 1999. *Formulation and Tabu Search Alogrithm for the Resource Constrained Project Scheduling Problem.* Kyoto : s.n., 1999. Technical Report #99010.

Olfert, K. 2010. *Projektmanagement.* 7. Auflage. Neckargemünd : Kiehl Verlag, 2010. [WU QP 360 O45(7).

Özdamar, L. and Ulusoy, G. 1996. A Note on an Iterative Forward/Backward Scheduling Technique with Reference to a Procedure by Li and Willis. *European Journal of Operational Research.* 1996, 89, pp. 400-407.

Patzak, G. and Rattay, G. 2009. *Projektmanagement.* 5. Auflage. Wien : Linde Verlag, 2009. [WU: QP360 P322(5)].

Runzheimer, B., Cleff, T. and Schäffer, W. 2005. *Operations Research 1, Lineare Planungsrechnung und Netzplantechnik.* 8. Auflage. s.l. : Gabler, 2005. [WU, QH 411 R943 (8)].

Schirmer, A. and Riesenberg, S. 1997. *Parameterized Heuristics for Project Scheduling - Biased Random Sampling Methods.* Kiel : s.n., 1997. Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel. 456.

Schwarze, J. 2010. *Projektmanagement mit Netzplantechnik.* 10. Auflage. s.l. : nwb Studium, 2010. [WU, QP 360 S411 (10)].

Shaffer, L. R., Ritter, J. B. and W.L., Meyer. 1965. *The Critical Path Method.* New York : McGraw-Hill, 1965.

Thiruvady, D., et al. 2012. Constraint-based ACO for a Shared Resource Constrained Scheduling Problem. *International Journal of Production Economics.* 2012, p. http://dx.doi.org/10.1016/j.ijpe.2012.06.012.

Valls, V., Ballestin, F. and Quintanilla, S. 2003b. *A Hybrid Genetic Algorithm for the RCPSP. .* s.l. : Departamento de Estadistica e Investigacion Operativa, Universidad de Valencia, 2003b. Technical Report.

—. 2008. A hybrid genetic algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research.* 2008, 185, pp. 495-508.

—. 2005. Justification and RCPSP: A technique that pays. *European Journal of Operational Research.* 2005, Vol. 165, pp. 375-386.

—. 2006. Justification Technique Generalizations. [ed.] J. Józefowska and J. Weglarz. *Perspectives in Modern Project Scheduling.* s.l. : Springer, 2006.

—. 2003a. Resource-Constrained Project Scheduling: A Critical Activity Reordering Heuristic. *European Journal of Operational Research.* 2003a, 149, pp. 282-301.

Wiest, J.D. 1964. Some Properties of Schedules for Large Projects with Limited Resources. *Operations Reserach.* 1964, Vol. 12, 3, pp. 395-418.

Xiao, J. and Tang, Y. 2012. Solving Software Project Scheduling Problems with Ant Colony Optimization. *Computers & Operations Research.* 2012, p. http://dx.doi.org/10.1016/j.cor.2012.05.007.

Yang, X. S., Deb, S. and and Fong, S. 2011. Accelerated Particle Swarm Optimization and Support Vector Machine for Business Optimization and Applications. *NDT.* 2011, pp. 53-66.

Zäpfel, G., Braune, R. and Bögl, M. 2010. *Metaheuristic Search Concepts: A Tutorial With Applications to Production and Logistics.* Heidelberg : Springer-Verlag, 2010.

Zimmermann, H-J. 2008. *Operations Research, Methoden und Modelle.* 2. Auflage. s.l. : Vieweg, 2008. [WU: QH 400, Z74(2)].

„Ich habe mich bemüht, sämtliche Inhaber der Bildrechte ausfindig zu machen und ihre Zustimmung zur Verwendung der Bilder in dieser Arbeit eingeholt. Sollte dennoch eine Urheberrechtsverletzung bekannt werden, ersuche ich um Meldung bei mir.“

Anhang

I. Zusammenfassung

Diese Arbeit beschreibt den Einsatz von genetischen Algorithmen für die optimale Nutzung von Ressourcen in kapazitätsbeschränkten Netzplänen. Nach einer Einführung in die Netzplantechnik und dessen Stellenwert im Projektmanagement, wird das kapazitätsbeschränkte Projektplan Problem (Resource-Constrained Project Scheduling Problem – RCPSP) definiert und es werden Lösungsmöglichkeiten dafür aufgezeigt, deren Fokus auf den genetischen Algorithmen, aus dem Bereich der Metaheuristiken, liegen. Die detaillierte Beschreibung der aktuell besten Verfahren wird durch die Vorstellung einer Testdatenbank (PSPLIB), deren Problemfälle verschiedenste Verfahren vergleichbar machen, abgeschlossen. Der zweite Teil dieser Arbeit analysiert den genetischen Algorithmus von Valls et al. (2008) und beschreibt eine programmatische Umsetzung dieses Verfahrens. Des Weiteren wurden Verbesserungsversuche an dieser Metaheurisitik vorgenommen und dokumentiert. Der Einsatz von alternativen Crossover-Operatoren, die anderen Methoden entnommen sind, zeigen nur bei den kleinen Probleminstanzen minimale Wirkung. Mechanismen die Klone in der Population unterbinden, zeigen in allen Instanzen Erfolge.

II. Abstract

This diploma thesis describes the application of genetic algorithms to use resources optimal in a resource-constrained project. After an introduction into project planning and its significant value for project management, the resource-constrained project scheduling problem (RCPSP) is defined. Focused on genetic algorithms, possible solutions are shown, which are part of the meta-heuristics. After showing the best known genetic algorithms in detail, a presentation of a representative test data library (PSPLIB) completes the theoretical part of this thesis. The second part of this paper analysis the hybrid genetic algorithm of Valls et al. (2008) and shows a self-developed application which is based on this meta-heuristic. Furthermore, improvement attempts were made and documented. Using crossover-operators from other solutions, show minimal effects only on short instances (J30). Avoiding clone in the population improved all instances.

III. Curriculum Vitae

Markus Lopin

Persönliches

Geboren am 01. Juli 1977, österreichischer Staatsbürger

Studium und Ausbildung

Seit 10/1997 — **Universität Wien, Betriebswirtschaftszentrum**

Studienrichtung: Internationale Betriebswirtschaft

Kernfachkombinationen

- Produktionsmanagement
  o. Univ.-Prof. Dipl.-Ing. Dr. Richard F. Hartl
- Internationales Management
  Ao. Univ.-Prof. Mag. Dr. Josef Windsperger

1991 – 1996 — **Technologisches Gewerbemuseum, HTL, Wien**

Ausbildungszweig: Biomedizintechnik

Berufserfahrung

Seit 12/2000 — **SmartStream Technologies GmbH**

Abteilung SVENSON
Software für das österr. Bankenmeldewesen

Projektleitung
Software Engineer

Kenntnisse

Sprachen — **Deutsch:** Muttersprache
Englisch: Verhandlungssicher in Wort und Schrift
Französisch: Gute Kenntnisse in Wort und Schrift

EDV — **Software Development** (VB.NET, C#.NET, ASP.NET AJAX, jQuery, VB 6), **Application Lifecycle Management** (TFS), **Database Programming** (Oracle, SQL Server), **Software Deployment** (Wise, MSI)FF