Thema:

Entwicklung von Algorithmen zu dem „Resource Constraint Project Scheduling Problem" auf der Basis eines definierten Projektmodells

Diplomarbeit

Arbeitsgruppe Wirtschaftsinformatik

Themensteller:

Betreuer: Prof. Dr. Hans-Knud Arndt

Vorgelegt von: Maik Gruhne

Abgabetermin:

Inhaltsverzeichnis

Verzeichnis der Abkürzungen und Akronyme

Ajax	Asynchronous Javascript And XML
B&B	Branch-and-Bound
CAL	Client Access Lizenz
CPM	Critical Path Method
DIN	Deutsche Industrie Norm
EDV	Elektronische Datenverarbeitung
FAZ	frühster Anfangstermin
FEZ	frühster Endtermin
GA	Genetischer Algorithmus
GRD	Greatest-Resource-Demand
GRPW	Greatest-Rank-Poritional-Weight
ISO	International Organization for Standardization
LST	Latest-Start-Time
MIS	Most-Immediate-Successors
MPM	Metra Potential Methode
MPS	Multi Mode RCPSP
MS	Microsoft
NPT	Netzplantechnik
PERT	Program Evaluation and Review Technique
PM	Projekt-Management
POP	Population

RAM	Random Access Memory
RCPSP	Resource Constraint Project Scheduling Problem
SAZ	spätester Anfangstermin
SEZ	spätester Endtermin
SGS	Schedule Generation Scheme
SPT	Shortest-Processing-Time
XML	Extensible Markup Language

Abbildungsverzeichnis

Tabellenverzeichnis

Einleitung

0.1 Motivation

Zu der Bedeutung von Projektmanagementsoftware in der heutigen Zeit tragen Ahlemann und Backhaus [AhlBa] in ihrer Veröffentlichung zahlreiche aktuelle Fakten zusammen, worauf im Folgenden eingegangen werden soll. Die folgenden Ausführungen werden im Laufe der Arbeit ausführlich behandelt und sollen vorerst einen Einstieg in das Problemthema geben.

In den letzten Jahren sind **P**rojekt-**M**anagement-**S**ysteme (kurz PM-Systeme) mehr und mehr zu den am meisten genutzten Tools im Projektgeschäft geworden. So wurde laut Untersuchungen in 75% aller durchgeführten Projekte PM-Software eingesetzt. Ursprünglich wurden diese Systeme laut Ahlemann und Backhaus [AhlBa] für das „network planning“ und „scheduling“ ausgelegt. Mit „network planning“ ist vor allem die Erstellung, Modellierung und Planung von Projektmodellen gemeint. Der Begriff „scheduling“ hat sich, auch in der deutschsprachigen Literatur, zu dem hier vorliegenden Problem als eigenständig etabliert und soll aus diesem Grund näher erläutert werden. „Scheduling“ kann mit den Wörtern „Ablaufkoordinierung“, „Terminierung“, „Zeitplanung“ oder „Ablaufplanung“ übersetzt werden. Daraus ergibt sich der Begriff Schedule, welcher einen berechneten Netzplan oder eine nach bestimmten Zielvorgaben errechnete Lösung wiedergibt (optimierter Netzplan).

Ein PM-System konzentriert sich in seinen Grundfunktionen immer auf die grafische Darstellung und Modellierung von Projekten. Eine solche definierte Projektinstanz beinhaltet in vielen Fällen kritische Situationen wie beispielsweise das Vorhandensein von Überlastbereichen von Ressourcen. Kommt es zu einem solchen Fall, ist es Aufgabe des Tools, derartige Ressourcenkonflikte mithilfe von geeigneten Algorithmen auszugleichen. Die Qualität der Algorithmen schlägt sich dabei zuerst auf die niedrigeren Projektkosten nieder, welche durch eine verkürzte Projektdauer verhindert werden können. Bei einer höheren Dauer kann es beispielsweise zu Überschneidungen kommen, wenn ein Projekt in den zeitlichen Verlauf eines Anderen hineinragt – Konkurrenzsituationen um Ressourcen sind vorprogrammiert. Die Literatur gibt zu diesem Thema eine Fülle von Lösungsmöglichkeiten vor. So ist es interessant, inwiefern es möglich ist, diese theoretischen Verfahren auf konkrete Projektmodelle in die Praxis zu überführen und mit bestehenden Lösungen zu vergleichen. Das zugrunde liegende Optimierungsproblem wird in der Literatur als „Resource Constraint Project Scheduling Proplem“ - kurz RCPSP – bezeichnet.

Es sei an dieser Stelle vorweggenommen, damit soll auf das Kapitel 4 verwiesen werden, dass ein Einzug von wirklichen Optimierungsalgorithmen zum RCPSP in PM-Anwendungen nicht nachvollziehbar stattgefunden hat. Im Zuge der Recherchen zu der vorliegenden Arbeit ist der Autor auf kein System mit einer solchen Funktion gestoßen.

Die zentrale Frage dieser Diplomarbeit lässt sich also wie folgt formulieren: Ist es möglich, gegenüber etablierten PM-Systemen Verbesserungspotenziale umzusetzen und welche Vorteile ergeben sich daraus für den Nutzer einer solchen Software?

0.2 PM-Systeme im Wandel

Die angesprochenen rudimentären Funktionen zur Darstellung und Modellierung von Projekten sind in der heutigen Zeit längst nicht mehr ausreichend. Ein zeitgemäßes PM-System muss mittlerweile alle Aspekte des Projektmanagements abdecken. Auch hat sich die Nutzerklientel erheblich gewandelt – wonach es in den frühen Anfangsjahren vorrangig den Projekt-Managern vorbehalten war, mit dieser Software zu arbeiten, so nutzen heute alle am Projekt beteiligten Personen ein solches System: Projektteam, Kunde, Leitungsausschüsse und das Management des Unternehmens.

Auch hinsichtlich der technischen Basis haben sich PM-Systeme über die Zeit deutlich verändert: Monolithische Desktop-Applikationen aus der Anfangszeit haben sich mittlerweile zu hoch skalierbaren Unternehmenslösungen entwickelt.

Parallel zu dieser Entwicklung hat sich eine große Anzahl von Systemangeboten vom Single-Projektmanagement zum Multiprojektmanagement zugewandt. Die Weiterentwicklung der Internettechnologie trägt ebenfalls zu einem großen Umbruch bei und beeinflusst moderne Lösungen maßgeblich.

Projektbeteiligte können heute unter Nutzung Groupware-spezifischer Technologien auch über geografische und zeitliche Grenzen hinweg miteinander kommunizieren und Probleme lösen. Eine große Gruppe von Software-Systemen dieser Kategorie wird mittlerweile als unternehmensweite Lösung eingesetzt.

In solchen Fällen erleichtert die Projektmanagementsoftware den rationalisierten und standardisierten Ablauf von Projekten und gibt Unterstützung in der ganzheitlichen Sicht auf das Projektportfolio. Sind gemäße Schnittstellen vorhanden und das System offen konzipiert, so lässt sich im Idealfall das System auf die individuellen Wünsche und Anforderungen eines Unternehmens übertragen.

Die Auswahl der geeigneten Software gestaltet sich mehr und mehr komplexer (Quelle: Ahlemann und Backhaus [AhlBa]):

- Die Anzahl und die funktionale Vielfalt der zur Zeit verfügbaren Systeme sind in den letzten Jahren enorm angestiegen, was eine Auswahl nicht unbedingt erleichtert - 220 Systeme von 200 Herstellern sind derzeit auf den Markt.

- Die Anforderungen der Nutzergruppen im Unternehmen müssen berücksichtigt werden; ein solches System muss auf allen Ebenen akzeptiert werden, sonst kann es nicht erfolgreich in das Projektgeschäft integriert werden.

- Die Unternehmens- und Projektphilosophie muss aufgegriffen und vom gewählten System widergespiegelt werden.

- Schnittstellen zu bestehenden Systemen müssen vorhanden sein.

So ist zu sehen, dass in früheren Tagen vor allem die Beschaffung und Installation das Hauptproblem bei der Einführung von PM-Software im Unternehmen darstellte. Heutzutage hat sich dies vor allem zu einem organisatorischen Problem gewandelt.

0.3 Zielsetzung der Arbeit

In der vorliegenden Arbeit sollen zunächst alle theoretischen Modelle des „Resource Constraint Project Scheduling“ RCPSP dargestellt und diskutiert werden. In diesem Zusammenhang ist es wichtig zu klären, ob diese Modelle den praktischen Anforderungen von PM-Software-Systemen vollständig genügen und sich in diesen Kontext übertragen lassen.

Unabhängig davon ist klar, dass die etablierten Systeme bereits einen Weg gefunden haben, Praxisanforderungen zu überführen und Algorithmen zu implementieren, die das RCPSP – Problem zu lösen in der Lage sind. Dabei ist interessant, welche Formen von Algorithmen zum Einsatz kommen und ob diese in der Lage sind, gute Lösungen zu finden. Wenn aus der Diplomarbeit hervorgeht, dass dies nicht der Fall ist, so ist zu untersuchen, welche Heuristiken darüber hinaus existieren und ob diese implementierbar sind. Auch ist deren Komplexität und Performance ausschlaggebend für einen möglichen praktischen Einsatz. Ein Algorithmus ist, auch im Falle einer optimalen Lösungsfindung, uninteressant, wenn Performancekriterien (Rechenzeit, benötigter Arbeitsspeicher) nicht erfüllt werden können.

Es ist also nicht ausschließlicher Gegenstand der Arbeit, eine theoretische Betrachtung zum RCPSP vorzulegen, vielmehr soll durch praktische Tests nachgewiesen werden, dass es durchaus Verbesserungspotenzial der Lösungen in der praktischen Umsetzung des RCPSP gibt. Es ist zu klären, welche Algorithmen die qualitativ besten Ergebnisse bringen und wie deren Performance ist. So ist die Zielsetzung dieser Arbeit vor allem

darin zu sehen, qualitative Verbesserungspotenziale praktischer Lösungen zum RCPSP aufzudecken, indem theoretische Verfahren auf ihre Praxiseignung untersucht werden.

Dem Autor ist im Verlauf seiner Arbeit und v.a. während der Literaturrecherche aufgefallen, dass zum Einen die theoretische Betrachtung auf das RCPSP bereits weit fortgeschritten und ausreichend beschrieben ist. Auf der anderen Seite ist jedoch eine deutliche Kluft zwischen theoretischer Betrachtung und praktischer Umsetzung innerhalb der Projektmanagementsoftware zu verzeichnen. Es wird zunächst Aufgabe dieser Diplomarbeit sein, diese Lücke ein Stück weit zu schließen.

0.4 Aufbau der Arbeit

Die Diplomarbeit unterteilt sich in 5 Hauptkapitel und dem Anhang.
Angefangen mit einführenden Worten zum Themengebiet des Projektmanagement, welche für diese Arbeit von Bedeutung sind, stellt Kapitel 2 überdies die PM-Software SimProQ und deren Projektmodell vor. Das Modell von SimProQ soll Ausgangspunkt und Plattform der Implementierung der in der Diplomarbeit zu behandelnden Algorithmen sein. Auch spielt der Begriff Konsistenz in Verbindung mit dem vorzustellenden System eine wichtige Rolle.
Das Kapitel 3 stellt im Anschluss die zentralen Themen zur theoretischen Erläuterung des RCPSP heraus. Es werden die bedeutendsten Modellvarianten des RCPSP vorgestellt und welche Ansätze zu deren Lösung derzeit vorhanden sind. Dabei soll ein grober Überblick der in der Literatur behandelten Algorithmen zusammengefasst werden. So sollen exakte Algorithmen, einfache Prioritätsregeln und Metaheuristiken zur Sprache kommen. Des Weiteren soll im Rahmen einer Komplexitätsbetrachtung veranschaulicht werden, welche Heuristiken für die praktische Anwendung in Frage kommen.
Nach der vorwiegend theoretischen Betrachtung sollen unter Kapitel 4 die praktischen Anforderungen an Modelle des RCPSP formuliert werden. Welche realweltlichen Probleme rechtfertigen diese Anforderung und können diese ganzheitlich in einem PM-System nachgebildet werden? Aufgrund der Komplexität und der Fülle von Ausprägungen aller theoretischen Modelle wird es nicht möglich sein, diese Frage innerhalb der Diplomarbeit allumfassend zu klären. Der Autor widmet sich daher, auch in Abhängigkeit von dem Projektmodell aus SimProQ, verstärkt einem einzelnen konkreten Modell. Die Testergebnisse, welche unter Kapitel 4 präsentiert werden, sollen erste Rückschlüsse auf die praktische Eignung der vorgestellten Algorithmen geben. Daneben geben Ergebnisse aus der Literatur einen Überblick von der qualitativen Umsetzung des RCPSP in aktuellen kommerziellen PM-Systemen. Im Anschluss soll verdeutlicht werden, welche Vorteile sich aus der Verbesserung der Qualität der Lösung des RCPSP in diesen Systemen ergeben.

Im letzten Kapitel 5 wird analysiert, welche Fragen in Zusammenhang mit einer vollständigen praktischen Umsetzung der bestehenden Modelle beantwortet werden müssen. Welche Verbesserungen in der Implementierung der Algorithmen sind denkbar und wie können diese zukünftig umgesetzt werden?
Im Anhang wird ein konkretes Projektbeispiel behandelt und eine Lösung durch die hier vorgestellten Algorithmen und aus MS Project gegenübergestellt.

1 Projektmanagement

1.1 Zentrale Begriffe und Definitionen

Bevor die Aufgaben und vielfältigen Aspekte des Projektmanagements dargestellt werden, ist es nötig, die Basis und damit zentrale Begriffe, welche in diesem Zusammenhang von Bedeutung sind, zu beleuchten und deren Besonderheiten aufzuzeigen.

Der Begriff „Projekt“ ist nach DIN 69901 charakterisiert durch eine eindeutige Zielvorgabe, welche nach zeitlichen, finanziellen, personellen oder weiterer Begrenzungen und unter Abgrenzung zu anderen Vorhaben in einer spezifischen Organisation durchgeführt wird.

Ein Projekt ist vor Allem durch seine Einmaligkeit hinsichtlich der Umstände und Bedingungen gekennzeichnet, wobei der Aspekt der Ganzheitlichkeit hervorzuheben ist.

Aufgrund der Komplexität und der Vielfalt der Aufgaben, die mit der Durchführung eines Projektes auftreten, ist es nötig, herkömmliche Führungskonzepte auf die Anforderungen des Projektmanagements zu übertragen und zu erweitern. So ergibt sich eine ganzheitliche Sicht auf entsprechende Parameter wie Kosten, Termine und Qualität der Projektdurchführung im Allgemeinen und des zu entwickelnden Produktes bzw. Projektergebnisses im Besonderen.

Burghardt [Burgh] nennt hierzu fünf Merkmale, die in Verbindung der Führung von Projekten von Bedeutung sind:

- Projektadäquate Organisation
- Exakte Entwicklungsvorgaben
- Projektbezogene Planung
- Laufender Soll-/Ist-Vergleich
- Definiertes Entwicklungsende

Neben diesen Aspekten ist die Personalbereitstellung und Organisation der Beteiligten in Abhängigkeit von der Komplexität des Projektvorhabens von großer Bedeutung. In einem festgeschriebenen zeitlichen Rahmen müssen für das Projektvorhaben geeignete Mitarbeiter eingesetzt und ggf. neben ihrer Funktion und Verantwortung in der Linienorganisation in einem leistungsfähigen Projektteam zusammengesetzt werden. An dieser Stelle existiert ein Grundkonflikt zwischen Linien- und Projektorganisation, da qualifizierte und zentrale Know-How-Träger meist rar sind und nicht in allen parallelen Projekten mitarbeiten können.

Um ein Entwicklungsvorhaben erfolgreich durchführen zu können, sind exakte Vorgaben hinsichtlich der Leistungsmerkmale des zu entwickelnden Produkts, des einzusetzenden Personals, der benötigten Sach- und Geldmittel sowie der Zeit, in der das Projekt durchgeführt werden soll, von entscheidender Wichtigkeit. Diese Vorgaben sind der Ausgangspunkt für eine projektbezogene Planung, welche sowohl die Projektstruktur als auch die Prozessstruktur enthält.

Eine zentrale Bedeutung kommt laut Burghardt [Burgh S. 10] der Projektkontrolle zu, da hier frühzeitig Abweichungen von Planungsvorgaben erkannt werden müssen, um den Korrekturaufwand überschaubar zu halten. Dieser beschriebene ganzheitliche, moderne Ansatz des Projektmanagement sichert schließlich zum Ende des Projektes einen kontrollierten Abschluss, wonach das gesamte Projekt zusammenfassend analysiert wird, um die so gesammelten Erkenntnisse für zukünftige Projektvorhaben zu nutzen.

Breitenfeld [Breiten] gibt zusammenfassend zu den vorangegangenen Erläuterungen die folgende Definition aus der DIN 69901 an:

„Unter Projektmanagement versteht man organisatorische Verfahren und Techniken, mit der die erfolgreiche Abwicklung eines Projektes verbunden ist. Die Norm DIN 69901 definiert Projektmanagement als die ‚Gesamtheit Führungsaufgaben, -organisation, -techniken -mittel für die Abwicklung eines Projektes.' “

Innerhalb des Projektmanagements müssen alle Aktivitäten für die Definition, Planung, Kontrolle und Abschluss eines Projektes unter dem Fokus einer zielgerichteten Abwicklung koordiniert werden. Dabei ist zu beachten, dass ein Projekt nie isoliert betrachtet werden kann, sondern in den gesamten Entwicklungsbereich eines Unternehmens eingegliedert werden muss. Im Allgemeinen gibt es neben dem Projektmanagement eine weitere Ebene, welche diesem übergeordnet ist - das Entwicklungsmanagement. Dieses nimmt projektübergreifend Planungs- und Steuerungsaufgaben wahr und kontrolliert entsprechende Eckparameter. Dabei stehen den Managementebenen unterschiedliche Entwicklungsdienste wie Qualitätssicherung, Bauunterlagenerstellung, Konfigurations- und Dokumentations-verwaltung zur Verfügung, die i. Allg. projektübergreifend von Bedeutung sind. Im Folgenden soll anhand der Darstellung des Projektablaufes verdeutlicht werden, wie umfangreich sich das Aufgabenspektrum des Projektmanagements darstellt.

1.2 Projektablauf

Bevor in Kapitel 2 moderne PM-Systeme hinsichtlich ihres Funktionsumfangs und ihrer Kernintention untersucht werden können, müssen zunächst die einzelnen Projektphasen vorgestellt werden. Die Verbindung der unter diesem Kapitel erläuterten Problemstellungen im Rahmen des Projektmanagements und der Intention von PM-Systemen kann erst hergestellt werden, wenn geklärt ist, welche Anforderungen bestehen und wo eine

Abgrenzung zu anderen Werkzeugen sinnvoll ist. Unter Berücksichtigung des zu begrenzenden Umfangs dieser Arbeit soll an dieser Stelle nur kurz auf die einzelnen Phasen eingegangen und Schlagworte genannt werden. Die nachfolgenden Aussagen beziehen sich auf die Publikation von Burghardt [Burgh], der die einzelnen Phasen und die verbundenen Techniken, Werkzeuge und Problemstellungen ausführlich darlegt.

Um eine sach-, termin- und kostengerechte Abwicklung eines Projektes zu gewährleisten, ist es Aufgabe des Projektmanagements, durch geeignete Regelungs- und Steuerungsmaßnahmen auf den Projektverlauf einzuwirken. Der Projektablauf unterteilt sich in folgende Phasen:

- Projektdefinition
- Projektplanung
- Projektkontrolle
- Projektabschluss

Die Projektdefinition bildet die Grundlage für alle späteren Phasen und damit die Basis für eine erfolgreiche Durchführung des Projektes. Diese Phase beinhaltet die formale Gründung des Projektes, die Definition des Projektzieles sowie die Organisation des Projektes und der Prozesse. Das Projektziel ist in Zusammenarbeit mit dem Auftraggeber zu definieren und bringt als Ergebnis den Aufgabenkatalog bzw. das Pflichtenheft hervor. Unter Analyse des Problemumfeldes ist die zu erwartende Wirtschaftlichkeit des zu entwickelnden Produktes zu ermitteln. Bevor das Projekt begonnen werden kann, sollten alle organisatorischen, fachlichen und wirtschaftlichen Aspekte des Vorhabens diskutiert und abgesichert werden.
Im nächsten Schritt ist die gesamte Ablauforganisation zu bestimmen. Dies beinhaltet das Festlegen der Entwicklungsphasen, die Definition von Pflichtmeilensteinen, Entwicklungslinien und Tätigkeitsarten.

Die anschließende Phase der Projektplanung beginnt, aufbauend auf den Anforderungskatalog, mit der Strukturplanung. Diese beinhaltet eine technische, aufgabenbezogene und kaufmännische Gliederung des Entwicklungsvorhabens. Die sich daraus ergebenden Inhalte – Produktstruktur, Projektstruktur und Kontenstruktur – bilden die Ausgangsbasis für alle weiteren Planungsschritte.
Aus dem Projektstrukturplan können die einzelnen Aufgabenpakete, basierend auf einer Aufwandsschätzung, abgeleitet werden. Eine Darstellung der daraus hervorgehenden Arbeitspakete und Teilaufgaben mithilfe von Netzplänen eröffnet eine bessere Übersichtlichkeit und bessere Kontrollmöglichkeiten. Ergänzend soll mithilfe der Einsatzmittelplanung ein optimaler Einsatz des vorhandenen Personals und der Betriebs- und Sachmittel erreicht werden.

Zum Schluss ist es unerlässlich, in Form einer ganzheitlichen Kostenplanung eine detaillierte Übersicht über Kostenarten und Kostenträger zu erarbeiten. Dies bildet die Grundlage für einen wirtschaftlichen Projektablauf und nur so ist eine Preisbildung möglich.
Diese vorangegangenen Aspekte sollen im nächsten Kapitel als Kernaufgabenfeld heutiger PM-Systeme näher erläutert werden.

Mit Beginn der eigentlichen Projektdurchführung setzt die Phase der Projektkontrolle ein. Im Vordergrund steht hierbei der ständige Soll-/Ist-Vergleich. Das Abweichen von Planvorgaben kann darauf basierend effizient erkannt und frühzeitig geregelt werden. Die Projektkontrolle umfasst folgende Bereiche:

- Terminkontrolle
- Aufwands- und Kostenkontrolle
- Sachfortschrittskontrolle
- Qualitätssicherung
- Projektdokumentation

Bei großen Projekten ist, wie bereits beschrieben, eine effiziente Terminkontrolle nur mit der Netzplantechnik praktikabel durchzuführen. Nur durch Unterstützung von geeigneten Tools ist ein Gesamtüberblick über Abhängigkeiten zwischen den einzelnen Aufgabenpaketen und ggf. gesetzten Terminen möglich. Der Einsatz von Trendanalysen ist sowohl für die Terminkontrolle als auch die Aufwands- und Kostenkontrolle ein leistungsfähiges Werkzeug, um Rückschlüsse auf Planabweichungen und geeignete Maßnahmen daraus abzuleiten.
Neben der Kontrolle von Parametern wie Termine, Kosten und Aufwand hält die Sachfortschrittskontrolle den Grad der Fertigstellung des zu entwickelnden Produktes fest. Dies zählt zu den schwer zu kontrollierenden Parametern, da es hierfür i. Allg. keine Messgrößen gibt und lediglich auf Schätzungen zurückgegriffen werden kann.
Die Qualitätssicherung kann hingegen in vielen Fällen auf detaillierte Normen und Richtlinien wie ISO-Standards und Modelle zurückgreifen. Das Ziel der Qualitätssicherung ist das Hervorbringen eines qualitativ hochwertigen Produktes unter Verwendung minimaler Entwicklungskosten. Ergänzend sichert die Projektdokumentation Erkenntnisse über die Durchführung, hält Erfahrungen und Kenntnisse für spätere Projektvorhaben fest und gewährleistet die Weiterentwicklung eines Unternehmens und die Pflege eines leistungsfähigen Wissensmanagements. Diese Daten sind eine wichtige Grundlage für die Abschlussphase des Projektes, welche auch die Erfahrungssicherung zum Inhalt hat. Das Sammeln von Daten bildet hierbei die Basis zur Erstellung und Nutzung von Kennzahlensystemen.
Darüber hinaus ist die Phase des Projektabschlusses durch die Schritte der Produktabnahme, der Projektabschlussanalyse und der Projektauflösung gekennzeichnet. Diese Begriffe sollen

an dieser Stelle für sich selbst stehen, die ausführliche Darstellung – Problembereiche, Methoden und Werkzeuge - dieser Projektphase ist nachzulesen in Burghardt [Burgh S. 237 ff].
Nachdem die einzelnen Projektphasen und deren inhaltlichen Abhängigkeiten kurz umrissen und schematisch dargestellt wurden, sollen nun zwei Elemente herausgegriffen werden, die nach Ansicht des Autors für die vorliegende Arbeit von entscheidender Bedeutung sind und die Verknüpfung zu den in Kapitel 2 vorgestellten PM-Systemen und in Kapitel 3 diskutierten „Resource Constraint Project Scheduling Problem“ verdeutlichen. Dies ist zum Einen die Ablauf- und Terminplanung und zum Anderen die Ressourcenplanung. Der Autor stützt sich dabei auf die Inhalte aus Patzack und Rattay [PatzRat].

1.3 Ablauf und Terminplanung

Die Grundlage der Ablauf- und Terminplanung besteht aus der Kenntnis des Probleminhalts. Damit werden die zu erledigenden Arbeitspakete verstanden, welche im Verlauf der Ablauf- und Terminplanung in Abhängigkeit ihrer technologischen und zeitlichen Kriterien angeordnet werden. Das erzielte Ergebnis bildet die Grundlage für terminliche Aussagen und weiteren Planungsschritten, wie des Ressourceneinsatzes, des Kostenverlaufes und des Finanzierungsbedarfes. Die Ablaufplanung legt somit alle logischen Anordnungsbeziehungen der Aufgabenpakete fest und beschreibt deren vollständige Vernetzung von Projektbeginn bis Projektende. Darauf aufbauend kann der erstellte Projektplan durch den Parameter Zeit erweitert werden, sodass eine Fristen- und Terminplanung durchgeführt werden kann.

Zunächst sollen die wesentlichen Bestandteile und Begriffe, die für die Erstellung eines Projektplanes nötig sind, zusammengefasst werden:

- Arbeitspaket bzw. Vorgang: dadurch wird ein bestimmtes Geschehen repräsentiert; Synonym sind die Begriffe Job, Activity, Tätigkeit oder - wie es im Laufe der Arbeit noch stellenweise gebraucht wird – Task zu verwenden.
- Ereignis oder Meilenstein: geben einen bestimmten Zustand eines Vorgangs, Teilprojekts oder Projekts wieder. Die Dauer eines Start- oder Endereignisses ist immer 0.
- Abhängigkeit: beschreibt die Beziehungen unter den Vorgängen, welche sich aus den technologischen Erfordernissen oder organisatorischen Randbedingungen ergeben.

Nachdem grundlegende Elemente definiert wurden, sollen im nächsten Schritt Methoden und Technologien der Ablauf- und Terminplanung vorgestellt werden. Die Autoren Patzack und Rattay [PatzRat] stellen an dieser Stelle 5 Methoden vor – Geschwindigkeitsdiagramm, Terminliste, zeitfixierter Balkenplan, vernetzter Balkenplan und Netzplan. In der nachfolgenden Darstellung wird ausschließlich auf den vernetzten Balkenplan, der, ebenfalls

ausführlich nachzulesen in Patzack und Rattay [PatzRat S. 201], gegenüber den restlichen Methoden entscheidende Vorteile aufweisen kann und deshalb auch in dem Simulationstool zur Anwendung kommt.
Patzack und Rattay [PatzRat] verwenden den Begriff Balkenplan synonym zu Gantt-Diagramm, welcher im Folgenden verwendet werden soll.
Der Zweck eines Gantt-Diagramms ergibt sich in der grafischen Darstellung von Aufgaben und ihrer Ablauflogik. Die zeitliche Einordnung, kritische Wege und vorhandene Pufferzeiten der Arbeitspakete können so sichtbar gemacht werden. Ein weiterer wesentlicher Aspekt besteht in der starken Kommunikationskraft, die sich im Austausch mit Auftraggeber und innerhalb der Projektorganisation ergibt.
Der Ablauf zur Erstellung eines Gantt-Diagramms beginnt mit der Auflistung aller Aufgaben und evtl. deren Zerlegung nach Maßstab einer sinnvollen inhaltlichen Größe. Im Anschluss muss die Dauer bzw. Bearbeitungszeit geschätzt und die Abhängigkeiten zwischen Aufgaben und Prozessen definiert werden. Anhand der sich daraus ergebenden Ablaufplanung können Termine fixiert und bestehende Meilensteine auf ihre Durchführbarkeit untersucht werden.
Die Erstellung des Diagramms sollte immer unter dem Aspekt der Übersichtlichkeit und Lesbarkeit erfolgen. Wird die grafisch dargestellte Vernetzung der Balken untereinander zu dicht, so kann sich die eigentliche Intention dieser Methode nicht durchsetzen.

Die grafische Darstellung des Arbeitablaufs des Projektes wurde damit festgehalten. Im nächsten Schritt erfolgt nunmehr die Planung der zur Verfügung stehenden Ressourcen.

1.4 Ressourcenplanung

Im Rahmen mehrerer Befragungen von Projektmanagern zu Ursachen von Schwierigkeiten in Projekten ergab sich auf Platz eins das Problem der Ressourcenzuordnung. Die Auflistung der zehn meistgenannten Gründe für Schwierigkeiten in Projekten ist nachzulesen in Patzack und Rattay [PatzRat S. 203]. Die Autoren geben auf der Seite 205 einen Einstieg in das Verständnis und die Bedeutung der Einsatzmittelplanung:

„Ziel der Einsatzmittelplanung oder auch Ressourcenplanung ist die Planung und Darstellung des Bedarfs an Einsatzmitteln im Zeitverlauf. Dabei werden die erforderlichen ausgewählten Einsatzmittel den Vorgängen und Arbeitpaketen zugeordnet.“

Die Ressourcenzuteilung kann dabei auf globaler oder vorgangsbasierter Ebene erfolgen: Global bedeutet, dass für das gesamte Projekt der Bedarf an Personentagen geschätzt wird. Im Gegenzug bezieht sich die vorgangsbasierte Planung auf den bereits besprochenen Terminplan, sodass Ressourcen in Bezug auf die zeitliche Lage der Aufgabenpakete und der Verfügbarkeit zugeordnet werden.

Nach Ermittlung des Vorrats und des Bedarfs an Mitarbeitern für ein Projekt ist der letzte Schritt im Rahmen der Einsatzmittelplanung die Optimierung der Personalauslastung - ein zentraler Punkt im Zusammenhang mit der Optimierung des Projektplanes. Nach Burghardt [Burgh] gestaltet sich diese Maßnahme wie nachfolgend beschrieben.

Es wird versucht, Arbeitspakete aus Überlastbereichen in Bereiche mit geringerer Auslastung zu verlegen. Fachliche, technische, organisatorische und personelle Abhängigkeiten regeln dabei die Möglichkeiten eines Ausgleichs.

Es wird an dieser Stelle zwischen terminlicher und kapazitätstreuer Einsatzmittelplanung unterschieden. Bei dem terminlichen Ansatz werden Vorgänge so gelegt, dass ohne ein Überschreiten des Endtermins ein gleichmäßiger Verlauf erreicht wird. Maßgeblich bei der kapazitätsgesteuerten Methode dagegen ist, dass sie die Summenkurve unter einen vorgegebenen Vorratswert drückt. Übertragen auf das Modell ist die kapazitätsgetreue Vorgehensweise an dieser Stelle entscheidend: Mitarbeiter können zunächst nur bis 100% belastet werden. Alle Vorgänge, die zusammengenommen diese Grenze überschreiten, müssen ausgeglichen werden. Dabei ist es möglich, dass ein gegebener Endtermin oder Meilensteine überschritten werden müssen, um die Konsistenz des Projektplanes zu gewährleisten.

Durch das systematische Aufteilen des Personals wird ein gleichmäßiger Netzplan angestrebt. So ist zum Einen das Ziel, die Mitarbeiter wo möglich zu 100% auszulasten, also weder Über- noch Unterbelastung, und zum Anderen den Personalbedarf der Aufgaben zu decken.
Patzack und Rattay [PatzRat S. 212] unterscheiden an dieser Stelle zwischen Einsatzmittelbedarfsglättung und Einsatzmittelbedarfsbeschränkung. Die zuerst genannte Methode verfolgt einen „weichen“ Abgleich, was bedeutet, dass durch Verschiebung von Vorgängen unter Verwendung möglicher Pufferzeiten systematisch Spitzen und Täler im Bedarfsprofil ausgeglichen werden, sodass die Projektdauer eingehalten wird. Nach dieser Maßnahme kann es, und wird es in vielen Fällen auch, zu weiteren Überschreitungen der Verfügbarkeit kommen, wobei dieses Problem punktuell durch Aufstockung und technologischen Maßnahmen eliminiert werden kann. Daraus ergeben sich jedoch Mehrkosten, welche den vermiedenen Pönalkosten gegenübergestellt werden müssen.

Im Rahmen der Diplomarbeit soll jedoch die zweite Vorgehensweise in den Vordergrund gerückt werden – die Einsatzmittelbedarfsbeschränkung. Diese wird auch als „harter“ Abgleich bezeichnet, da die Verfügbarkeit der Ressourcen in keinem Fall überschritten werden darf. Die Tatsache, dass dadurch die Projektdauer in vielen Fällen nicht beibehalten werden kann, stellt das Hauptproblem dar. Kann eine Überschreitung nicht akzeptiert werden, muss wiederum durch punktuelle Maßnahmen entgegengesteuert werden. Das Ergebnis ist somit ein kapazitätstreuer Terminplan, wobei die Minimierung der Projektdauer im zentralen Blickfeld steht. Patzack und Rattay [PatzRat S. 213] kritisieren diese Vorgehensweise

energisch: „Die unbedingte Einhaltung der Verfügbarkeit der Ressource liefert bei Anwendung der diversen EDV-Software Programme praktisch immer unvertretbare Projektverlängerungen ... Die Anwendung dieses Optimierungsalgorithmus ist daher in der Praxis nur sehr eingeschränkt Nutzen bringend.“

Es muss an dieser Stelle verdeutlicht werden, dass die vorangegangenen Aussagen nicht vollständig zutreffend sind. Im Zuge dieser Diplomarbeit - und damit soll auf das Kapitel 3 verwiesen werden – wird veranschaulicht, dass gegenwärtige Software-Systeme gar keine Optimierungsalgorithmen anwenden. Somit ist der Begriff „Optimierungsalgorithmus“ nicht zutreffend und die Aussage inkorrekt.

Im Kapitel 2 sollen zunächst eine Auswahl der soeben angesprochenen Software-Systeme vorgestellt werden. Im Anschluss werden Methoden und Algorithmen erläutert, die die Probleme der Einsatzmittelbedarfsbeschränkung eindämmen können.

2 Projektmanagement- Softwaresysteme

2.1 Multiprojektmanagement

Über die Einprojektplanung hinaus treten im Rahmen der simultanen Planung mehrerer Projekte weitere Szenarien auf, welche in diesem Kapitel zur Sprache kommen sollen. Ist es der Fall, dass mehrere Projekte zu einer Zeit innerhalb eines Unternehmens oder einer Organisation auftreten, erfordert dies einen höheren Planungs- und Lenkungsbedarf. Neben sachlichen und terminlichen Abhängigkeiten sind Mitarbeiter oftmals in mehrere Projekte eingebunden (siehe Schwab [SchwTool]). Auch muss anhand eines Projektportfolios entschieden werden, welche Projekte im Sinne der Unternehmensphilosophie durchgeführt werden sollten oder nicht. Als Eingangsdaten zur Erfassung sachlicher und terminlicher Abhängigkeiten dienen die Terminplanungen der betroffenen Projekte. In einem projektübergreifenden Metaplan müssen demnach Abhängigkeiten definiert werden, um einen umfassenden Terminplan zu erhalten. Im Sinne der Einsatzmittelplanung muss die Arbeitskraft der Mitarbeiter gemäß ihrer Verfügbarkeit erfasst und gesteuert werden. Dabei ist darauf zu achten, dass Mitarbeiter weder über- noch unterbelastet werden.

Ein PM-System muss im Hintergrund die Konsistenz über alle Projekte und Zeiträume gewährleisten und durch Algorithmen testen. Im Backend entsteht somit ein höherer Rechenaufwand. Im Rahmen des Projekt-Portfolio-Managements, der inhaltlichen Kontrolle und dem Treffen von strategischen Entscheidungen, stoßen Projektplanungsinstrumente an ihre Grenzen. So können beispielsweise nur Plan-Ist-Abweichungen von Terminen, Arbeitsaufwand und Kosten errechnet werden [SchwTool].

Auch die Lösung von Konflikten gestaltet sich über mehrere Projekte hinweg komplexer. Die folgenden Punkte sind dabei zu beachten:

- bei mehreren voneinander mehr oder weniger autonomen Projekten sind viele leitende Personen beteiligt, was zur Folge hat, dass Konfliktsituationen, v.a. im Zuge der Zuteilung von Ressourcen und Budget, entstehen können.
- ein unübersichtliches Projektportfolio kann zum Scheitern der einzelnen Projekte führen.
- der Anwender muss beim Zuteilen von Ressourcen deren bisherige/derzeitige zeitliche Belastung beachten.
- es ist oft nicht sinnvoll, eine einzelne Person in mehreren Projekten einzusetzen → der Stressfaktor steigt und die Leistungsfähigkeit sinkt; Koordinationsprobleme kommen hinzu.

Arbeitspakete lassen sich nunmehr auch mit projektexternen Vorgängen verknüpfen. So kann ein Vorgang beispielsweise erst beginnen, wenn ein bestimmtes Arbeitsergebnis aus einem anderen Projekt gegeben oder ein Prozessbaustein bereits abgeschlossen ist.

In diesem Punkt gelten alle Modellierungsregeln wie bei der Verknüpfung der Arbeitspakete mit internen Vorgängen. Der Unterschied besteht in der Berechnung des bisherigen Netzplans und die Erweiterung der bisher dargestellten Algorithmen. Nunmehr fließen auch projektexterne Arbeitsdauern in die Berechnung mit ein.

2.2 MS-Project – ein typischer Vertreter für PM-Systeme

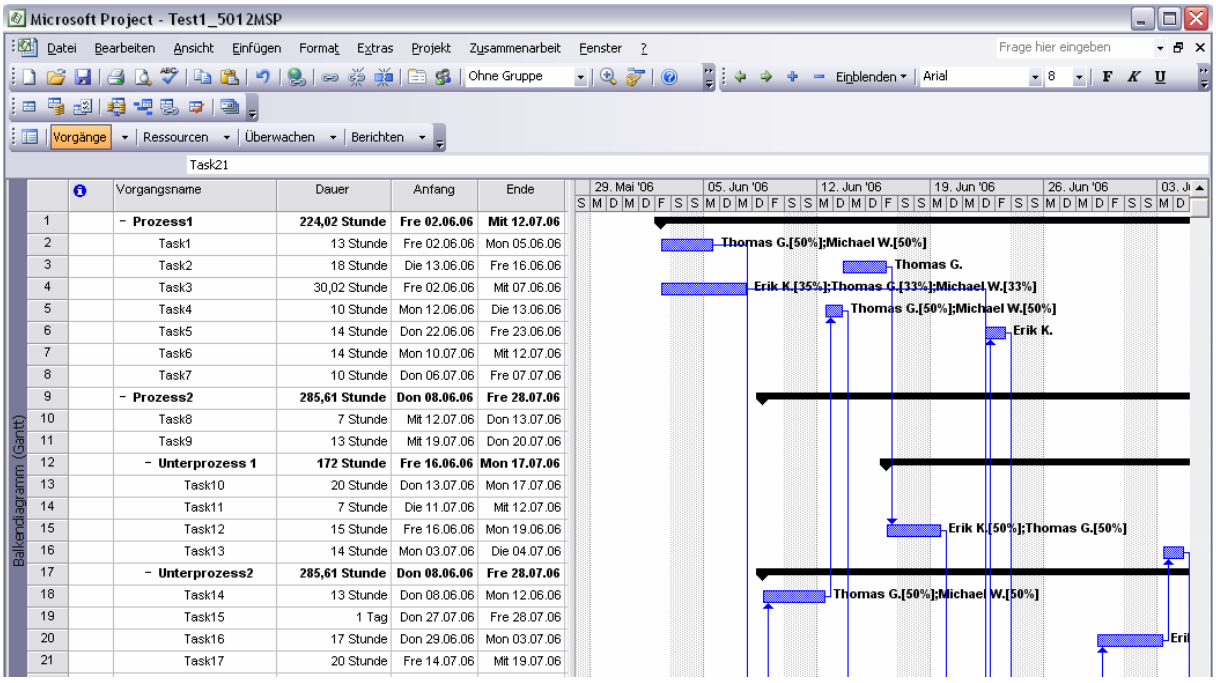
In diesem Kapitel soll ein typischer Vertreter für PM-Software Systeme vorgestellt werden – MS-Project in der Version 2003 zusammen mit Project Server. Um auch nur einen kleinen Einblick von den Features zu geben, welche diese Software in sich vereint, reicht dieser Abschnitt bei Weitem nicht aus. Dennoch bemüht sich der Autor, die wesentlichen Aspekte hervorzuheben, welche als Kernfeatures von PM-Software Systemen essentiell erscheinen. Darüber hinaus soll der Blick auf aktuelle Trends und Entwicklungen nicht verdeckt werden. Zu einer ausführlichen Studie sei an dieser Stelle die Publikation „Projektplanung realisieren mit MS Project 2003 und Project Server 2003“ von Schwab [SchwMSP] empfohlen.

2.2.1 Allgemeine Leistungsmerkmale

Gegenwärtige PM-Systeme unterscheiden sich in ihrem Umfang und der angebotenen Features ganz erheblich untereinander. Bei, wie eingangs beschrieben, 220 Systemen von 200 Herstellern erscheint diese Feststellung nicht verwunderlich.
MS Project ist das am Markt am meisten eingesetzte Tool zur Unterstützung des Projektmanagements (siehe Schwab [SchwTool]), was die Auswahl dieser Software als typischen Vertreter für PM-Tools begründet.
Die folgende Beschreibung stammt von Burghardt [Burgh S. 282] und fasst die wesentlichen Punkte der Software zusammen:
„MS Project ist ein Projektsteuerungsverfahren auf Basis der Netzplanmethode; es ermöglicht die Planung und Steuerung des Arbeits- und Zeitablaufs eines Projekts, den Einsatz von Personal und Betriebsmitteln (Ressourcen) sowie die Kostenbudgetierung für große und kleine Projekte“
Modellierte Projekte können in mehrere Unterprojekte unterteilt und durch deren Teilnetzpläne miteinander verknüpft werden. Nach der von Burghardt beschriebenen Version kann ein Netzplan 200 Vorgänge aufnehmen, wobei für jeden Vorgang 16 Vorgänger bzw. Nachfolger bestimmt werden können. Des Weiteren können Vorgänge, welche in regelmäßigen Abständen auftauchen, als standardisierte Arbeitspakete hinterlegt werden.
Der Modellierer kann in der Ressourcensicht praktisch beliebig viele Ressourcen wie Personal, Maschinen und Räumlichkeiten definieren und in eine Kostenplanung einbeziehen.

Diese Ressourcen werden in der Folge, im Zuge einer vollständigen Einsatzmittelplanung (siehe Kapitel 1.4 - Ressourcenplanung), in den zeitlichen Verlauf – in der Form eines Gantt-Diagramms und eines Netzplanes – eingeordnet.
Im praktischen Einsatz ist es hilfreich, neben den drei Standardkalendern weitere benutzerdefinierte Kalender zu hinterlegen, die außer der Wochenenden weitere arbeitsfreie Zeiten abspeichern können (Feiertage, Betriebsurlaub). Auch ist die Möglichkeit eines ressourcenbezogenen Kalenders ein starkes Feature, welches erlaubt, individuelle Arbeitszeiten zu definieren, wie z.B. Urlaubstage, Weiterbildungsmaßnahmen, Mutterschaftsurlaub oder Teilzeitarbeitszeiten.
Bei der Netzplanberechnung wird sowohl die Vorwärts- als auch die Rückwärtsrechnung berücksichtigt, wodurch eine Darstellung der kritischen Wege möglich ist.

Das zentrale Planungsinstrument, das Gantt-Diagramm, ist in der folgenden Abbildung nochmals dargestellt.



	Vorgangsname	Dauer	Anfang	Ende
1	**Prozess1**	**224,02 Stunde**	**Fre 02.06.06**	**Mit 12.07.06**
2	Task1	13 Stunde	Fre 02.06.06	Mon 05.06.06
3	Task2	18 Stunde	Die 13.06.06	Fre 16.06.06
4	Task3	30,02 Stunde	Fre 02.06.06	Mit 07.06.06
5	Task4	10 Stunde	Mon 12.06.06	Die 13.06.06
6	Task5	14 Stunde	Don 22.06.06	Fre 23.06.06
7	Task6	14 Stunde	Mon 10.07.06	Mit 12.07.06
8	Task7	10 Stunde	Don 06.07.06	Fre 07.07.06
9	**Prozess2**	**285,61 Stunde**	**Don 08.06.06**	**Fre 28.07.06**
10	Task8	7 Stunde	Mit 12.07.06	Don 13.07.06
11	Task9	13 Stunde	Mit 19.07.06	Don 20.07.06
12	**Unterprozess 1**	**172 Stunde**	**Fre 16.06.06**	**Mon 17.07.06**
13	Task10	20 Stunde	Don 13.07.06	Mon 17.07.06
14	Task11	7 Stunde	Die 11.07.06	Mit 12.07.06
15	Task12	15 Stunde	Fre 16.06.06	Mon 19.06.06
16	Task13	14 Stunde	Mon 03.07.06	Die 04.07.06
17	**Unterprozess2**	**285,61 Stunde**	**Don 08.06.06**	**Fre 28.07.06**
18	Task14	13 Stunde	Don 08.06.06	Mon 12.06.06
19	Task15	1 Tag	Don 27.07.06	Fre 28.07.06
20	Task16	17 Stunde	Don 29.06.06	Mon 03.07.06
21	Task17	20 Stunde	Fre 14.07.06	Mit 19.07.06

Abbildung 1: MS-Projekt-Hauptansicht

Der Screenshot zeigt alle für die Planung des Projektes wesentlichen Module. In der Tabelle findet der Nutzer eine vollständige Auflistung der bisher modellierten Vorgänge und Prozesse vor. Diese sind versehen mit allen relevanten Informationen wie Dauer, Anfang, Ende, Vorgänger und Beteiligte. Auf der rechten Seite ist das bereits angesprochene Gantt-Diagramm dargestellt. Hier kann eine Bearbeitung des Projektplanes mittels manueller Verschiebung der Vorgänge erfolgen.
Als weiteres Instrument steht dem Anwender ein Netzplan zur Verfügung. Der erstellte Balkenplan und die damit verknüpften Informationen können in Diesen übernommen werden.

2.2.2 Project Server, Web Acces

In MS Project gab es bis zu der Version 2000 nur dateibasierte Lösungen (Quelle [Schwab S. 7] und [SchnHieb S. 55]), wonach Projekte unter einer Datei zusammengefügt werden konnten. Wurden von mehreren Projekten dieselben Ressourcen genutzt, war es möglich, einen gemeinsamen Ressourcenpool zu definieren. Ab der Version 2002 steht mit dem Project Server eine datenbankbasierte Lösung bereit.

Der Project Server erlaubt durch ein vorgegebenes und auf individuelle Vorstellungen anpassbares Berechtigungskonzept die Steuerung des Zugriffs auf Informationen bis zu den einzelnen Benutzern. Mitglieder dieses Systems sind der Administrator, die Projektmanager und die Teammitglieder. Führungskräften, Ressourcen- und Portfoliomanagern können bestimmte Ansichten und Funktionen zur Verfügung gestellt werden. Die Benutzer können mehreren Gruppen angehören und besitzen individuelle Rechte.

Projekte werden zentral auf dem Server gespeichert und veröffentlicht, um die geänderten Daten allen Teammitgliedern innerhalb von Web Access und damit zur Kommunikation bereitzustellen. Das Projektcenter, ein Modul auf dem Server, zeigt eine Zusammenfassung aller Projektdaten, das Ressourcencenter die der Ressourcen. Auch hier kann der Anwender die einzelnen Sichten individuell auf seinen Informationsbedarf anpassen.

Darüber hinaus hat sich die Teamkommunikation mit den Versionen von MS-Project immer weiterentwickelt. Hier stellt die Server-Lösung einen weiteren bedeutenden Fortschritt dar. Die Daten werden, wie bereits erwähnt, auf dem Server veröffentlicht und können je nach Zugriffsrechten mit allen Beteiligten diskutiert werden. Dazu bedarf es lediglich einer Client-Access-Lizenz (CAL) – eine Projekt-Lizenz wird nicht benötigt. Benutzer können somit über den Browser, nach Angabe von Schwab kann dafür leider nur der Internet-Explorer genutzt werden, auf die Projektdaten zugreifen. Rückmeldungen der Beteiligten hinsichtlich des Fortschritts ihrer Tätigkeiten, das ständige Erfassen der Ist-Situation im Projekt, ist dabei sicher ein wichtiges Feature für die Verantwortlichen und Entscheidungsträger. Auch können Beteiligte eigene Vorschläge anbringen und sich eigene Vorgänge zuweisen.

Schwab betont darüber hinaus: „Grundsätzlich bietet die Einrichtung des Project Servers die Möglichkeit, echtes Multiprojektmanagement in dem Sinne zu betreiben, dass die Projektdaten zur differenzierten Auswertung je nach Bedarf zur Verfügung stehen und über den Web Access, nur mittels des Internet Explorers kommuniziert werden können“

2.3 SimProQ

Im derzeitigen Entwicklungszustand kann das PM-Tool SimProQ im Funktionsumfang noch nicht mit großen Namen wie MS-Project verglichen werden. Dennoch gibt es Ansätze, nicht zuletzt durch die Implementierung von Algorithmen zum Ressourcenausgleich im Rahmen

dieser Diplomarbeit, welche eine Abgrenzung zu etablierten Software-Systemen aufzeigt und auch Vorteile erkennen lässt.
Ursprünglich als Desktop-Anwendung konzipiert, haben sich die Entwickler von den neuesten Trends in der Web-Entwicklung inspirieren lassen und die Software komplett in die Browser-Umgebung verlagert. Dies ist vor allem auf die noch sehr junge Technologie Ajax (Asynchronous Javascript And eXtensible Markup Language), zurückzuführen (der Begriff taucht erstmals Anfang 2005 auf). Leider reicht der Platz auch an dieser Stelle nicht aus, um diese neue Entwicklung ausreichend zu beschreiben – dazu bedarf es einer genauen Analyse der zugrunde liegenden Technologie (siehe Denny Carl [Carl]). Festzuhalten ist, dass sich eine in Ajax entwickelte Software von einer konventionellen Desktop-Anwendung kaum mehr unterscheiden lässt. Anders als in der Vergangenheit, in der eine Internetseite komplett neu geladen werden muss, wenn neue Inhalte dargestellt werden sollen, kann mithilfe einer asynchronen Datenübertragung dieser Vorgang erheblich vereinfacht werden. Der Nutzer wird diesen neu gewonnenen Komfort in den meisten Fällen nicht bemerken, da er gemäße Features bereits aus Desktop-Anwendungen gewohnt ist. Für die hier zugrunde liegende Software eröffnen sich daraus jedoch viele Vorteile. Carl [Carl S. 9f] benennt unter Anderem die Folgenden:

- Eine Installation ist nicht mehr notwendig. Es muss lediglich eine Web-Adresse aufgerufen werden.
- Updates werden auf dem Server hinterlegt und stehen den Anwender sofort zur Verfügung
- Die Nutzung der Anwendung ist nicht länger auf einen Computer fixiert und kann von jedem Ort der Welt aus erfolgen.

Demnach lassen sich nicht nur die unter 2.2.2 beschriebenen Vorteile nutzen, sondern die gesamte Anwendung in den Browser übertragen werden.

2.3.1 Allgemeine Leistungsmerkmale

Mit dem zugrunde liegenden System, der PM-Software SimProQ, ist es möglich, ein Projekt auf allen Ebenen nachzubilden und zu modellieren. In Form einer Baumstruktur wird zunächst das Projekt selbst als Wurzelknoten betrachtet, wonach sich alle Aufgabenpakete oder Prozesse auf der nächsten Ebene verzweigen. Diese können wiederum als Container für weitere Arbeitspakete dienen. Die Verzweigung wird solange weiterbetrieben, bis auf der untersten Ebene des Projektbaumes die kleinsten unteilbaren Aufgabenpakete übrig bleiben, die meist nur einem Mitarbeiter zugeordnet werden können.

Die beschriebene Baumstruktur ist in der folgenden Abbildung dargestellt.

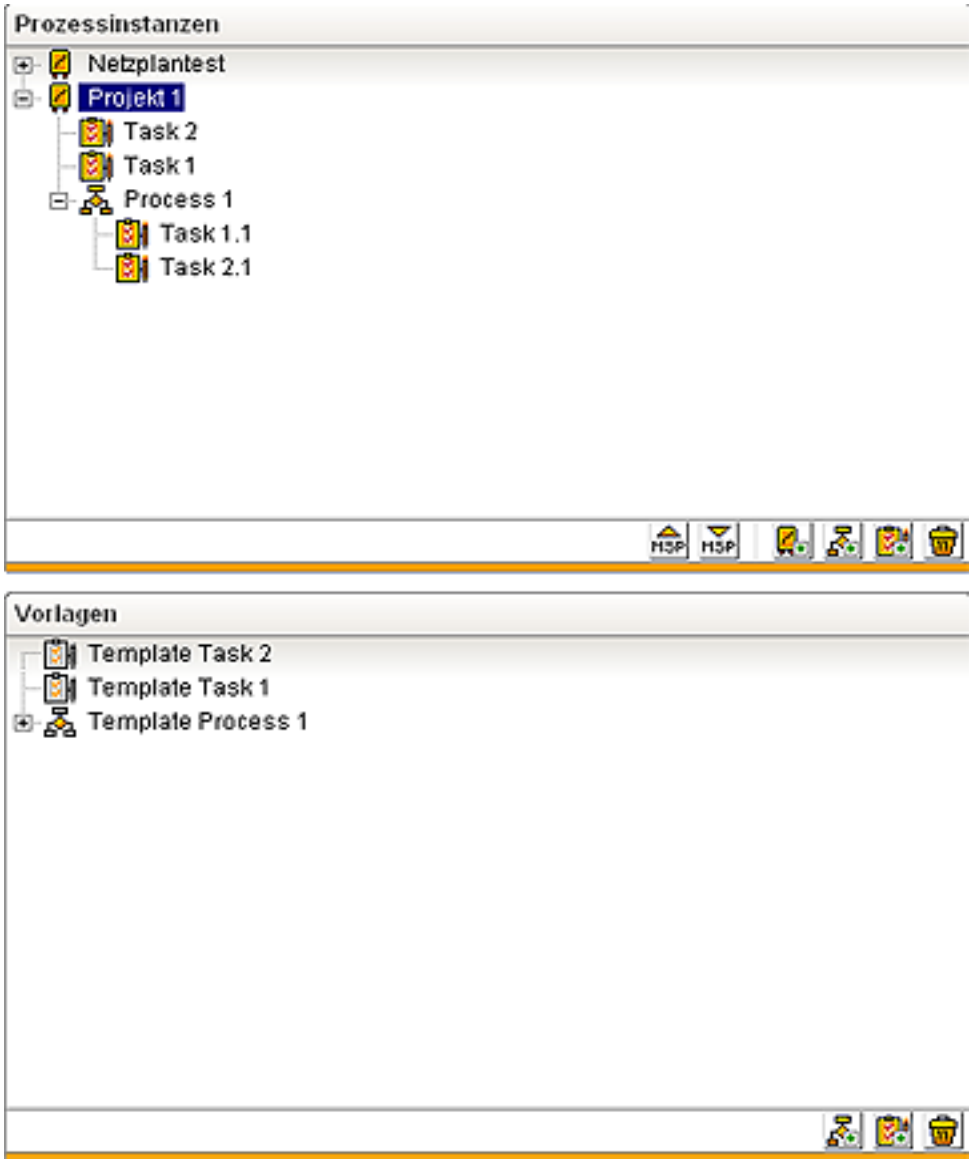


Abbildung 2: SimProQ-Modeller

Im unteren Bereich der Abbildung 2 ist die Modellierungssicht für Projektvorlagen zu sehen, im oberen Teil ist die Modellierung konkreter Projektinstanzen abgebildet. Um nach einer Projektdurchführung erlangtes Wissen und Erfahrungen zum abgeschlossenen Projekt zu sichern, ist es in jedem Falle ratsam, diese in Form von Projektvorlagen zu speichern. In der Praxis ist es oft der Fall, dass bestimmte Prozesse innerhalb von Projekten oder Projekte im Gesamten in ähnlicher Form immer wieder auftauchen. Hierbei ist es von großem Vorteil, auf gesichertes Wissen zurückzugreifen und dieses gegebenenfalls zu erweitern - dieser Hintergrund bildet die Intention der in der Grafik wiedergegebenen Sichten.

Für die sachgerechte Abwicklung eines Projektvorhabens ist es notwendig, eine umfangreiche und detaillierte Produktstruktur zu erstellen. Diese stellt einen Architekturplan des Produkts bzw. des Systems dar.

Eine Kontenstrukturierung sorgt demgegenüber für eine kostengerechte Projektabwicklung.

Für die termin- und aufwandsgerechten Projektabwicklung hingegen muss eine vollständige Strukturierung des Projektes in geeignete Prozessbausteine und Aufgabenpakete erfolgen. Die damit angestrebte inhaltlich ausgeglichene Projektstruktur ist Gegenstand der Software und steht in dieser Diplomarbeit im Vordergrund.

2.3.2 Reihenfolgebeziehungen

Ein wichtiger Punkt in Verbindung mit der Modellierung innerhalb von SimProQ ist die simultane Planung mehrer Projektinstanzen. Das bedeutet, dass dementsprechend auch parallel mehrere Projekte auf ihre Konsistenz geprüft werden müssen. In einem späteren Kapitel wird zu diesem Punkt vertiefend erläutert, welche Probleme damit einhergehen.

Die Modellierung der Vorbedingungen stellt eine weitere Besonderheit dar. Tasks werden im Netzplan durch AND/OR Bedingungen miteinander verknüpft. Auch können alternative Task-Zustände als Vorbedingung einer Nachfolger-Task definiert werden. So kann sich für jede Aufgabe in dem Projekt ein Baum aus Vorbedingungen ergeben, welche durch logische Verknüpfungen miteinander verbunden sind. Die folgende grafische Darstellung soll diesen Sachverhalt noch einmal beleuchten:

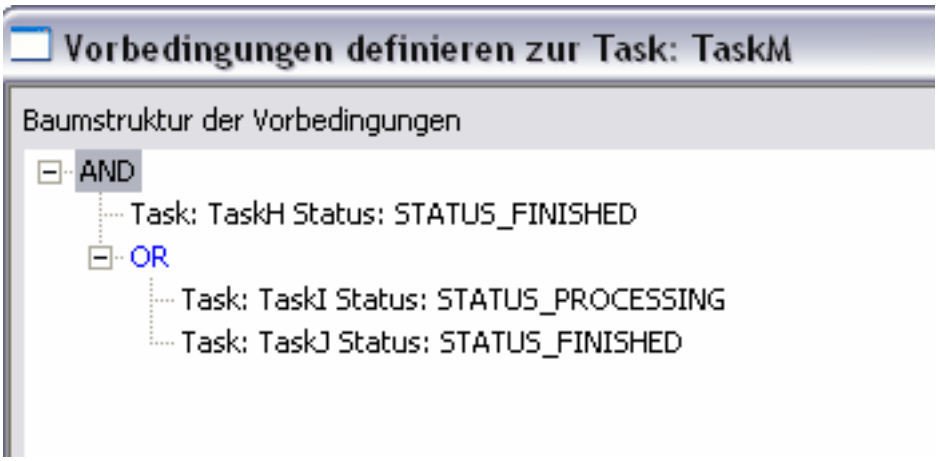


Abbildung 3: Modellierung der Vorbedingungen

Die Grafik zeigt die Modellierung der Vorbedingungen zu einer TaskM. Zu erkennen ist zum Einen die baumförmige AND/OR- Verknüpfungen. Des Weiteren sind Vorbedingungen

immer verbunden mit einem bestimmten Status der jeweiligen Vorgängeraufgabe. Das ermöglicht dem Nutzer komplexere Projekt-Strukturen zu definieren.

Bereits während der Definition der Vorbedingungen durch den Nutzer, demzufolge während der Modellierung, ist es angebracht, inkonsistente Projektmodelle zu vermeiden. So lassen sich beispielsweise Zirkelbezüge verhindern, die eine Netzplanberechnung und somit einen Ressourcenausgleich unmöglich machen.

Durch das Zulassen von OR-Verknüpfungen ergibt sich ein geänderter Netzplanaufbau, der bei der Berechnung berücksichtigt werden muss. So können beispielsweise Vorgänger und Nachfolger durch eine faktische „Anfang-Anfang“ Definition auch parallel ablaufen, insofern sich dadurch keine Ressourcenüberbelastungen ergeben.

Die OR-Verknüpfung eröffnet dem Nutzer weitere Modellierungsmöglichkeiten und einen flexibleren Netzplan. So kann ein Aufgabenpaket durch mehrere alternative Vorbedingungen angestoßen werden, d.h. unterschiedliche Projektzustände können bewirken, dass eine bestimmte Aufgabe gestartet werden kann.

2.4 Vergleich der Projektmodelle

Das Modell von MS Projekt bildet keine alternativen OR-Vorbedingungen ab: Arbeitspakete werden unter eindeutigen definierten Vorbedingungen angestoßen. Die Verknüpfung kann durch die vier Beziehungsarten Ende-Anfang, Anfang-Anfang, Ende-Ende und Anfang-Ende vorgenommen werden. In Kapitel 4.2 werden diese Beziehungsarten genauer unter die Lupe genommen.

Durch die OR-Darstellung ergeben sich weitere Möglichkeiten zur Modellierung des Projektablaufes, der in dieser Form in MS Project nicht möglich ist: Ein Vorgang kann alternativ von mehreren Vorgängern angestoßen werden. Für eine Übertragung des vorliegenden Modells in MS Project durch einen Export ist dieser Sachverhalt nicht umzusetzen, d.h. die Projekte müssen in vereinfachter Struktur übertragen werden. Abgesehen von diesem Punkt ist es jedoch möglich, beide Systeme gegeneinander auf die Qualität ihrer verwendeten Algorithmen zu testen.

Das Multiprojektmanagement ist aufgrund des hohen Informations- und Lenkungsbedarfs zum unverzichtbaren Element moderner PM-Systeme geworden. Neben der dateibasierten Lösung wird dieser Aspekt durch den Project Server von Microsoft behandelt. Die zweite Variante ist zu vergleichen mit dem SimProQ-Modell, welches Projektpläne ebenfalls auf eine Datenbank abbildet und darauf basierend projektübergreifende Abhängigkeiten und Ressourcenkapazitäten überprüfen kann.

Neben notwendigen funktionellen und grafischen Features, wie der Darstellung des Projektes in einem Netzplan oder Gantt-Diagramm, fehlt es SimProQ überdies an modellspezifischen Grundlagen, welche in zukünftigen Versionen berücksichtigt werden müssen. So ist es derzeit nicht möglich, Aufgabenunterbrechungen zu modellieren. Dies wird notwendig sein, wenn ein Mitarbeiter beispielsweise seinen Urlaub in Anspruch nimmt und Aufgaben über eine bestimmte Zeit ruhen müssen.

3 Das „Resource Constraint Project Scheduling Problem“

3.1 Der Begriff Konsistenz

Konsistenz ist der zentrale Begriff im Rahmen dieser Arbeit. Zunächst einmal sollen allgemeine Definitionen zu dessen Bedeutung führen. Im Folgenden soll ein Auszug der Definition aus der Web-Bibliothek „Wikipedia“ wiedergegeben werden. Der Auszug enthält jene Passagen, welche nach Ansicht des Autors in Zusammenhang mit dem zu erörternden Problemfall von Bedeutung sind.

Definition Konsistenz [Wiki]:

„Das Wort Konsistenz (v. lat.: con = zusammen + sistere = halten) bedeutet Bestand, Zusammenhalt, Geschlossenheit und In-sich-Ruhen.

Konsistenz im weiteren Sinn bezieht sich auf den logischen, hermeneutischen oder ästhetischen Bestand eines Gegenstandes (z. B. eines Textes, eines Begriffs, einer Methode oder Technik, eines Werkzeugs oder eines Kunstwerks), der ‚in sich stimmig’ ist, ‚Sinn ergibt’ und keine inneren Widersprüche oder Spannungen aufweist, die seine Einheit gefährden.“

Im Sinne des hier zu betrachteten Konsistenzbegriffs bewegt sich diese Definition in der Nähe des vorliegenden Projektmodells.

Das Projektmodell kann also in diesem Sinne als konsistent betrachtet werden, wenn ein Modellzustand erreicht ist, der in der Realität, d.h. in der praktischen Durchführung, in Form eines konkreten Projektes auch durchgeführt werden kann. Die gegenteilige Ausprägung, wenn der Projektzustand demzufolge in sich nicht stimmig ist, hat zur Folge, dass das Projekt in der modellierten Form nicht durchführbar ist.

Im weiteren Sinne kann ein Projekt in sich selbst konsistent sein; in Zusammenhang mit bereits bestehenden Projekten hingegen kann es wiederum zu Unstimmigkeiten kommen. Wenn beispielsweise zwei Projekte ressourcenbezogen ausgeglichen sind, somit keine Überbelastung vorhanden ist, kann eine zeitliche Überschneidung wiederum zu Ressourcenüberlastungen führen.

Neben ressourcenbezogenen Aspekten kann ein fehlerhafter Zustand ebenfalls entstehen, wenn der geplante und modellierte Ablauf in sich unlogisch ist, d.h. Arbeitsabläufe sind falsch geplant bzw. in der definierten Reihenfolge nicht durchführbar. Diese Form der Inkonsistenz kann in der Regel von einer Software nur schwer erkannt werden und soll auch nicht Gegenstand der Betrachtung sein.

Vielmehr kann die Modellierung der Vorbedingung ebenfalls fehlerhaft sein, indem redundante Verknüpfungen erstellt werden. Bezogen auf die bereits verwendete Grafik zur Definition der Vorbedingungen zu „TaskM“ wäre folgende Definition nicht sinnvoll.

TaskH Status: Finished AND TaskH Status: Started AND TaskH Status: Aborted

Diese Form der Inkonsistenz kann jedoch leicht vom System erkannt und beseitigt werden. Bei umfangreichen Baumstrukturen wird es jedoch zunehmend komplizierter, Unstimmigkeiten wie Redundanzen zu erkennen.

Nach der im vorangegangenen Kapitel erläuterten Berechnung des Netzplans liegen nun folgende Informationen zu jedem Vorgang vor (Quelle Burghardt[Burgh]):

- Frühester Anfangszeitpunkt (FAZ)
- Spätester Anfangszeitpunkt (SAZ)
- Frühester Endzeitpunkt (FEZ)
- Spätester Endzeitpunkt (SEZ)

Eine Zeitkonsistenz ist demnach gegeben, wenn keine negativen Puffer auftreten. So lassen sich folgende Beziehungen für einen zeitkonsistenten Netzplan herausstellen:

$SAZ \geq FAZ$

$SEZ \geq FEZ$

Im Falle einer Inkonsistenz bezüglich dieser Zeitangaben muss der Netzplan nochmals überarbeitet werden. Der Nutzer sollte innerhalb der Modellierung darauf hingewiesen werden und zeitinkonsistente Tasks bzw. Projektbausteine müssen hinsichtlich der Vorgangsdauer, der Fixtermine oder hinsichtlich der Anordnungsbeziehungen bearbeitet werden.

3.2 Notation

Seit Beginn der 1990er Jahre und dem Zusammenschluss von Autoren zu der „Research Group Constraint Scheduling“, wurde die Forschung auf dem Gebiet des RCPSP entschieden vorangetrieben. Die vorrangigen Ziele dieser Vereinigung sind im Folgenden dargestellt:

Zum Einen ist es vor allem das Anliegen für Project Scheduling, nach intensiven Forschungen in den 1990er Jahren bis in die heutige Zeit, eine einheitliche Notation für Project Scheduling (PS) zu formulieren. Deren Fehlen hatte bisher bewirkt, dass bei der Fülle von Veröffentlichungen und den verbunden Modellvarianten, der Überblick bisweilen verloren

ging und eine Einordnung der Begriffe und Modelle schwer fiel. Kuhlmann [Kuhl] verweist an dieser Stelle auf Brucker et al., der diesen Mangel beseitigen wollte und mit der $\alpha \mid \beta \mid \gamma$ – Notation sich vor allem an der Maschinenbelegungsplanung orientierte. Die Grundsituation der Maschinenbelegungsplanung gestaltet sich folgendermaßen: Eine bestimmte Anzahl von Aufgaben bzw. Aufträgen muss auf einer oder mehreren Maschine bearbeitet werden. Die Optimierungsaufgabe besteht nun darin, einen, in Abhängigkeit der Forderungen aus der Zielfunktion, bestmöglichen Belegungsplan zu erstellen, sodass alle Maschinen einerseits gut ausgelastet sind und andererseits alle Aufträge in schnellstmöglicher Zeit abgearbeitet werden. Einen genaueren Erblick in das Thema „Maschine Scheduling“ soll das Kapitel 4.1.2 geben; dem soll nicht weiter vorgegriffen werden.

Mit α werden die Ressourcencharakteristika beschrieben, Eine Erweiterung zur Maschine Scheduling Notation ist hingegen unerlässlich, wobei α als Platzhalter für die folgenden Ausprägungen fungiert.

- $PS_{m,\sigma,\rho}$ steht für das PSP mit m Ressourcen, dabei stehen pro Zeitperiode σ Einheiten der Ressourcen zur Verfügung und eine Aktivität benötigt maximal ρ Einheiten. Wenn Ressourceninformationen im eigentlichen Probleminput spezifiziert sind, können m, σ, ρ entfallen. So handelt es sich um ein Ein-Modus-PSP.

- $MPS_{m,\sigma,\rho,\mu,\tau,\omega}$ bezeichnet hingegen ein Mehr-Modus-PSP mit m erneuerbaren und μ nicht erneuerbaren Ressourcen. Pro Zeitperiode sind demnach σ Einheiten der erneuerbaren und insgesamt τ Einheiten der nichterneuerbaren Ressourcen verfügbar.

Der Begriff Modus gibt die möglichen Ausprägungen des zu betrachtenden Vorgangs wieder. Im Ein-Modus-Fall ist klar vorgeschrieben in welcher Form Dieser durchzuführen ist während der Mehr-Modus-Fall unterschiedliche Varianten zulässt. So ist es denkbar, um ein einfaches Beispiel zu nennen, dass im Modus 1 eine Aufgabe 20 Zeitperioden bei einem Ressourcenverbrauch von 2 Einheiten benötigt. Im Gegenzug schreibt Modus 2 die doppelte Dauer mit nur einem Mitarbeiter vor. Das $MPS_{m,\sigma,\rho,\mu,\tau,\omega}$ ist somit eine Erweiterung des Standardfalls und somit komplexer bei der Berechnung.

Die Aktivitäten benötigen pro Zeitperiode insgesamt höchstens $\rho(\omega)$-Einheiten beider Ressourcentypen. Es können ebenfalls die Indizes entfallen, um den allgemeinen Mehr-Modus-Fall zu kennzeichnen.

Die Aktivitätscharakteristika werden nach der vorliegenden Notation mit griechisch β bezeichnet. Die folgenden Einträge können ggf. auch in Kombination betrachtet werden – nach Kuhlmann [Kuhl]:

- $p_j = 1$: Alle Aktivitäten haben eine Durchführungsdauer von einer Zeitperiode.

- p_j = sto: Alle Aktivitäten haben eine stochastische Durchführungsdauer, die um den Erwartungswert $E(p_j)$ schwankt.
- $\overline{d}$: Das Projekt muss bis zu dem Stichtag $\overline{d}$ fertig gestellt sein.
- *Prec*: Zwischen den Aktivitäten herrschen Nachfolgebeziehungen. So kann eine Aufgabe erst beginnen, wenn die definierten Vorbedingungen erfüllt sind. Die Aufgabe selber kann mit dem jeweiligen Status wiederum selbst Vorbedingung für eine andere Aufgabe sein.

Mit γ wird die Zielfunktion des Problems beschrieben. Die folgenden Funktionen standen bisher im Fokus der Veröffentlichungen zu diesem Problem:

Die Zielfunktion zur Minimierung der Gesamtprojektdauer C_{max} steht in der vorliegenden Arbeit im Vordergrund. Die Aktivitäten werden im Falle eines Ressourcenkonfliktes so verzögert und verschoben, dass die Gesamtdauer des Projektes minimiert wird. Nach Kuhlmann [Kuhl] ist die Optimierung nach der Gesamtdauer des Projektes das sinnvollste Vorgehen, da Zielwerte wie ein ausgeglichener Ressourcenverbrauch und eine frühzeitige Fertigstellung der Aktivitäten automatisch positiv beeinflusst werden. Im Kapitel 4.2 sind die Aussagen von Kuhlmann nochmals zusammengefasst.

Der nächste mögliche Zielfunktionswert bezieht sich auf den Kapitalwert oder „Net Present Value“, welcher durch das Investitionsvorhaben ‚Projekt’ verwirklichen lässt. Die formelle Beschreibung kann wie folgt zusammengefasst werden: $\sum_{j=0}^{n+1} c_j^F \beta^{C_j}$ - Der Cash-Flow jeder Aktivität j, j = 0,1,..., n+1, wird mit c_i^i bezeichnet. Der Abzinsungsfaktor wird durch β dargestellt und der Zeitpunkt der Fertigstellung einer Aktivität j wird durch C_j ausgedrückt. Der Kapitalwert ist somit definiert „… als die Summe der Barwerte der durch eine Investition hervorgerufenen, künftigen Einzahlungsüberschüsse abzüglich der Anfangsauszahlung für die Investition“ (Neus [Neus S. 271]). So wird der Kapitalwert als eines der entscheidenden Kriterien für Investitionsentscheidungen angesehen.

Ein weiterer wichtiger Gesichtspunkt ist der gleichmäßige Verbrauch von Ressourcen über die gesamte Projektlänge. Wenn beispielsweise Fixkosten für eine Ressource anfallen, welche unabhängig von der Beanspruchung berechnet werden, ist eine Unterbeanspruchung, v.a. bei hoch spezialisierten Maschinen oder Projektbeteiligten, immer als inakzeptabler Zustand und Verschwendung anzusehen. Dieser Sachverhalt kann mit der Formel $\sum_{i=1}^{m+\mu} c_i f(r_i(S,t))$ zusammengefasst werden und ist in der Literatur auch unter dem Begriff Resource-Levelling bekannt. Die Kosten einer Ressource i; i = 1,..., m+ μ pro Zeitperiode werden mit c_i ausgedrückt. Der Verbrauch von Ressource i zum Zeitpunkt t bei zugrunde liegendem Schedule S wird mit $r_i(S,t)$ angegeben und f stellt eine geeignete Funktion auf r_i dar.

Das „Resource Investment Problem“ beschreibt einen weiteren Aspekt als Ergänzung zu den vorangegangenen Erläuterungen. Diese Funktion hat zum Ziel, den Verbrauch besonders teurer Ressourcen so niedrig wie möglich zu halten. Es handelt sich somit um ein Minimierungsproblem und wird durch die Formel $\sum_{i=1}^{m+\mu} c_i \max_t (r_i(S,t))$ repräsentiert.

Zum Schluss soll das Minimierungsproblem „Weighted Earliness Tardiness“ (WET) zur Sprache kommen. $\sum_{j=0}^{n+1}(e_j E_j + t_j T_j)$: Eine Aktivität j, j= 0,1,..., n+1 besitzt einen Stichtag für ihre Fertigstellung. Mit E_j sind nunmehr die Anzahl der Zeitperioden wiedergegeben, die diese Aufgabe vor diesem Stichtag beendet werden kann. Falls es hingegen eine Verspätung einer Aktivität j gibt, wird diese mit T_j festgehalten. Mit ej und tj werden die jeweiligen Strafkosten einer Verfrühung bzw. Verspätung von einer Zeitperiode bezeichnet. An dieser Stelle spielen die in der Ablaufplanung hinterlegten Fixtermine für die Aufgabenpakete bzw. Prozesse die zentrale Rolle.

Vor- und Nachteile dieser Notation wurden in der Literatur bereits diskutiert. Zum Einen sind die Vorteile nach Kuhlmann [Kuhl] vor allem in der Einfachheit, Verständlichkeit und Nachvollziehbarkeit zu sehen. Zum Anderen können eine große Menge von Modellvarianten und Problembeschreibungen damit abgedeckt werden. Kuhlmann verweist in der Aufzählung der Nachteile auf Heroelen, der diese wie folgt zusammenfasst:

Einerseits ist der Zusammenhang mit dem Maschine Scheduling nur unvollständig wiedergegeben. Des Weiteren können Modellierungserweiterungen nicht abgebildet werden, da hierfür noch keine Bezeichnungen gegeben sind.

Zuletzt wird die Darstellung expliziter Zielkriterien im γ-Feld bemängelt, da dadurch eine Unterscheidung zwischen „regulären“ und „nicht regulären“ Zielkriterien nicht möglich ist.

3.2.1 Das RCPSP-Grundmodell

Im Folgenden definiert Kuhlmann [Kuhl] vier Grundtypen von Project Scheduling Modellen. Der Großteil der bisherigen Veröffentlichungen zum Project Scheduling bezieht sich auf diese Typen.

PS |*prec*| *: PS-Probleme mit allgemeinen Fertigstellungs-Start-Beziehungen und einem beliebigen Zielkriterium

Ein Projektmodell besteht aus mehreren Aktivitäten 1,..., n sowie einer Dummy-Start- und einer Dummy-Endaktivität n+1. Jede Aktivität besitzt eine Durchführungszeit $pj \in Z_{\geq 0}$ und soll zunächst nicht unterbrochen werden können. Start und Ziel haben eine Dauer von null Zeitperioden. V bezeichnet die Menge aller Aktivitäten in einem Projektmodell.

Zwischen den Aktivitäten können Vorrangbeziehungen <i,j> bestehen, wonach die Aktivität j erst beginnen kann, wenn Aktivität i beendet wurde. Die Menge *PREDj* bezeichnet in Folge dessen alle direkten Vorgänger und *SUCC j* alle direkten Nachfolger einer Aktivität j ∈ V. Die Projektzeit wird in t = 1,...,T diskrete Zeitperioden unterteilt, in denen die Aktivitäten angeordnet und durchgeführt werden. Es sind ausschließlich k = 1,..., R^ρ erneuerbare Ressourcen vorhanden, die unter der Menge R^ρ zusammengefasst werden.

„Erneuerbar" bedeutet in diesem Zusammenhang, dass eine Ressource zu jedem Zeitpunkt stets mit derselben Kapazität R^ρ_k zur Verfügung steht.

Eine Aktivität j ∈ V benötigt zu einer definierten Zeitperiode die Kapazität einer Ressource. Dieser Aspekt wird mit der Variable r^ρ_{jk} ausgedrückt. Bei der Durchführung einer Aktivität können beliebig viele Ressourcen beteiligt sein.

Der Startzeitpunkt einer Aktivität j ∈ V wird mit S_j, die Fertigstellung mit F_j bezeichnet. Gesucht wird demnach der Vektor (S_0, S_1,..., S_{n+1}), auch als Schedule bezeichnet, der alle angesprochenen Restriktionen beachtet und eine noch zu ermittelnde Zielfunktion optimiert.

Da sich die praktische Umsetzung, also Implementierung und Testung, ausschließlich auf die beschriebene Modellvariante beziehen, soll nachfolgend weitere Modelltypen nur kurz beschrieben werden.

3.2.2 Weitere Modelltypen

Neben den vorangegangen beschriebenen Erläuterungen zum Grundmodell des RCPSP sollen an dieser Stelle weitere Varianten und Ausprägungen zur Sprache kommen. Begonnen werden soll mit dem PS | *temp* | *:

PS | *temp* | *: PS-Probleme mit planungsabhängigen Zeitfenstern und einem beliebigen Zielkriterium

Diese Problemklassifikation wird in der Literatur auch als „RCPSP with Generalized Precedence Relations" (RCPSP-GPR) oder „RCPSP with minimal and maximal time lags" (RCPSP/max) bezeichnet.

Zu dem Grundmodell des RCPSP werden weitere Prämissen definiert. Es wird eine minimale bzw. maximale Zeitverzögerung zwischen zwei Aktivitäten i und j definiert. Diese Verzögerungen können von unterschiedlicher Ausprägung sein:

- SS_{ij}^{min} / SS_{ij}^{max} : min/max Zeitverzögerung zwischen dem Start von i und j

- SF_{ij}^{min} / SF_{ij}^{max} : min/max Zeitverzögerung zwischen dem Start von i und der Fertigstellungsperiode von j
- FS_{ij}^{min} / FS_{ij}^{max} : min/max Zeitverzögerung zwischen dem Fertigstellungsperiode von i und dem Start von j
- FF_{ij}^{min} / FF_{ij}^{max} : min/max Zeitverzögerung zwischen der Fertigstellungsperioden von i und j

MPS | *prec* | *: Mehr-Modus- PS-Probleme mit allg. Fertigstellungs-Start-Beziehungen und einem beliebigen Zielkriterium.

Die Einführung mehrerer Durchführungsmodi für eine Aktivität stellt für das Modell eine bedeutende Erweiterung dar. Das MPS |*prec*|* ist auch unter der Bezeichnung „Multi-Mode RCPSP“ (MRCPSP) bekannt. Neben den im Grundmodell vorgestellten Bedingungen werden die folgenden Prämissen aufgestellt:

Für jede Aktivität $j \in V$ werden mehrere Durchführungsmodi m_j, die in der Menge M_j festgehalten werden, eingeführt. Diese Modi unterscheiden sich durch die unterschiedliche Inanspruchnahme von Ressourcen und damit verbundenen Aktivitätsdauern. Ein Beispiel verdeutlicht die nahe liegenden praktische Relevanz des Problems: Im Laufe eines Softwareprojektes soll ein Modul entwickelt werden und mehrere Programmierer können den Auftrag dafür erhalten. Diese unterscheiden sich in Ihrer Qualifikation zu den sich aus der Aufgabe ergebenen Anforderungen. Auf der einen Seite steht der erfahrene und stark beanspruchte Entwickler zur Verfügung, der weit reichende Erfahrungen auf dem Gebiet vorzuweisen hat, aber in viele Projekte parallel eingebunden ist. Auf der anderen Seite müsste der erst vor kurzem zu dem Unternehmen gestoßene Praktikant sich erst in das Thema einarbeiten und Literatur studieren. Nun muss der verantwortliche Projektleiter unter Beachtung der Dringlichkeit und Wichtigkeit entscheiden, welche Lösung die beste für die jeweilige Situation ist. Neben der Dauer der Aktivität spielt auch die Entwicklungskosten für diesen spezifischen Fall eine maßgebliche Rolle.

Vor dem Einlasten einer Aktivität muss ein Durchführungsmodus gewählt werden, der während der gesamten Bearbeitung bestehen bleibt. Neben den erneuerbaren Ressourcen R^{ρ} steht die Menge R^{υ} der nicht erneuerbaren Ressourcen zur Verfügung. Mit Hilfe der nicht erneuerbaren Ressourcen lassen sich beispielsweise auch monetäre Restriktionen wie ein begrenztes Budget nachbilden.

MPS | *temp* | *: Mehr-Modus PSP mit planungsabhängigen Zeitfenstern und einem beliebigen Zielkriterium.

Werden die verschiedenen Durchführungsmodi aus dem vorangegangenen Modell mit den planungsabhängigen Zeitfenstern kombiniert, so ergibt sich das MPS | *temp* | * - auch als

„Multi – Mode RCPSP with minimal and maximal tine lags“ (MRCPSP/max) bezeichnet. Zu den bereits definierten Bedingungen aus den beschriebenen Modellvarianten ergibt sich die folgende Erweiterung: Die maximalen und minimalen Zeitverzögerungen hängen vom gewählten Modus der jeweiligen Aktivitäten ab.

3.3 Komplexitätsbetrachtung

Im Rahmen der Betrachtung von kombinatorischen Optimierungsproblemen, zu denen auch das RCPSP zu zählen ist, fällt der Komplexitätsbetrachtung des zu untersuchenden Problems eine entscheidende Bedeutung zu. Brucker und Knust [BruKnu] stellen die Fakten zusammen, wonach das RCPSP in dieser Hinsicht einzuordnen ist.

Wenn kein effizienter Algorithmus gefunden werden kann, so muss zunächst überprüft werden, ob das Problem *NP*-hart ist. Das bedeutet, dass hierzu mit hoher Wahrscheinlichkeit kein Algorithmus existiert, der das Problem auf effektive Weise optimal lösen kann.

Im Folgenden soll nun geklärt werden, in welchen Fällen von einem „leichten“ oder „schweren“ Problem gesprochen werden kann.

An erster Stelle müssen einige Begriffe erläutert werden, um das RCPSP in seiner Komplexität einzuordnen.

Es wird zunächst eine Inputgröße für den Algorithmus mit n definiert. Ein Algorithmus wird demnach als polynomial bezeichnet, wenn gilt: $O(n^k)$ für eine beliebige Konstante $k \in \mathbb{N}$.

So ist festzuhalten, dass exponentiale Funktionen mit steigendem Input sehr viel schneller an ihre Grenzen stoßen als polynomiale, sodass diese Algorithmentypen für größere Probleminstanzen nicht in Frage kommen.

Ein Problem kann somit als „leicht“ klassifiziert werden, wenn es in einer polynomialen Zeit gelöst werden kann.

Es handelt sich um ein sogenanntes Entscheidungsproblem, wenn die Antwort nur „ja“ oder „nein“ sein kann. Auf der anderen Seite gehören die meisten Scheduling-Aufgaben zu der Klasse der Optimierungsprobleme. In diesem Fall ist die Lösung eine gültige Berechnung, welche eine objektive Funktion minimiert oder maximiert. Für jedes Scheduling kann jedoch ein Entscheidungsmodell definiert werden, in dem geprüft wird, ob eine gültige Lösung existiert, die den Zielfunktionswert um einen bestimmten Wert y unter- bzw. überschreitet:

$$c(S) \leq y.$$

Die Klasse der polynomialen Entscheidungsprobleme wird zusammengefasst unter *P*. Eine andere wichtige Klasse ist zusammengefasst unter *NP*, was bedeutet dass diese Probleme

nicht deterministisch polynomial lösbar sind. Ein sogenanntes „Zertifikat“ für eine „ja“-Instanz ist eine Information, die überprüft, ob eine „ja“-Antwort für diese Instanz existiert. Wenn ein solches Zertifikat in polynomialer Zeit ausgestellt werden kann, so ist das Entscheidungsproblem *NP* zuzuordnen.

Brucker und Knust (S. 26) definieren den Begriff *NP*-schwer wie folgt:

„The central issue for the definition of NP-hardness is the notation of a polynomial reduction. A decision problems P is said to be polynomially reducible to an other decision problem Q (denoted by P α Q) if a polynomial-time computable function g exists which transforms every instance I for P into an instance I’ = g(I) for Q such that I is a ‘yes’-instance for P if and only if I’is a ‘yes’ instance for Q. “

Eine polynomiale Reduktion zwischen zwei Entscheidungsproblemen bedeutet dann, dass Q die gleiche Komplexität besitzt wie P - wenn Q polynomial lösbar ist, trifft dies auch auf P zu (und umgekehrt). So lässt sich weiter definieren, dass eine Transitivität besteht: wenn P α Q dann muss auch gelten Q α P.

Ein Entscheidungsproblem wird als NP-vollständig bezeichnet, wenn:

1. P zu der Klasse *NP* gehört und

2. jedes andere Entscheidungsproblem Q $\in$ *NP* polynomial reduzierbar zu P ist.

Ein Problem P wird nun als *NP*-schwer definiert, wenn nur 2. zutrifft. Speziell ein Optimierungsproblem wird als *NP*-schwer angesehen, wenn das korrespondierende Entscheidungsproblem *NP*-vollständig ist. In Anbetracht der Tatsache, dass bisher kein Algorithmus gefunden wurde, welcher in polynomialer Zeit eine Lösung für ein NP-vollständiges Problem errechnen kann, ist mit hoher Wahrscheinlichkeit davon auszugehen, dass kein solcher Algorithmus existiert (Quelle Brucker/Knust S. 26).

Brucker und Knust (S. 27f) zeigen, dass die Minimierung der Projektlaufdauer als Zielfunktion des RCPSP *NP*-schwer ist. So sind komplexere, aber auch vereinfachte Modelle des RCPSP nur mit unverhältnismäßig hohem Aufwand zu lösen. Bei einem RCPSP mit minimalen und maximalen Zeitverzögerungen und allgemeinen Vorrangbeziehungen ist die Lösung der Frage, ob es einen gültigen Schedule gibt, bereits *NP*- vollständig.

3.4 Das Auftreten von Ressourcenkonflikten

Ein Ressourcenkonflikt kann aus vielen Modelliervorgängen resultieren. Es soll im Folgenden zunächst festgehalten werden, in welchen Fällen dies geschehen kann, bevor Lösungsansätze dazu diskutiert werden.

Der einfachste und nahe liegende Fall ist die Zuordnung einer Ressource zu einem Vorgang. In der folgenden Abbildung wird dieser Sachverhalt dargestellt.

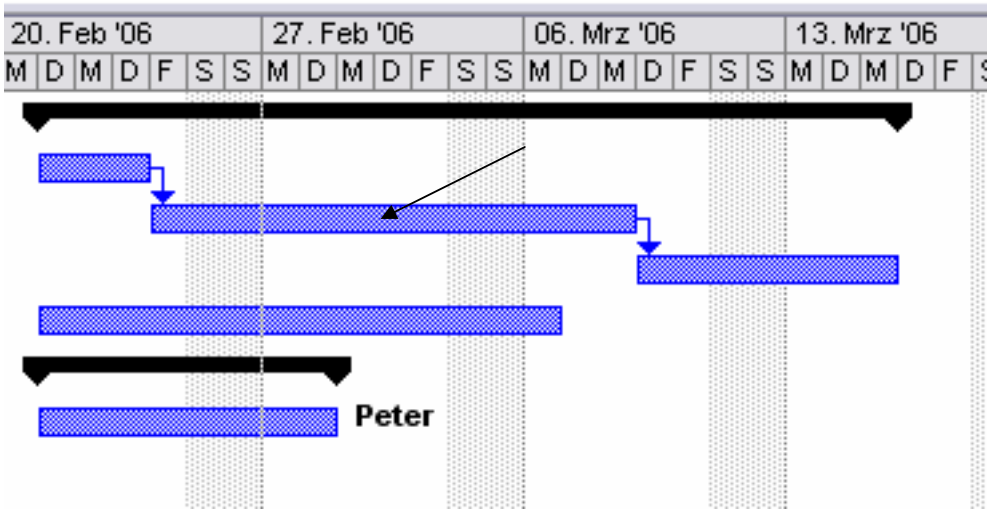


Abbildung 4: Projektplan ohne Konflikte

Dem markierten Vorgang wird nun ebenfalls die Ressource „Peter“ zugeordnet. Es ist ersichtlich, dass zwei Vorgänge nunmehr um dieselbe Ressource konkurrieren.

Darüber hinaus ist es möglich, dass durch die Änderung der Dauer eines Vorgangs die gleiche Situation entstehen kann, indem dadurch ein Vorgang in das Zeitintervall eines Anderen hineinragt. Die folgende Abbildung illustriert diesen Sachverhalt.

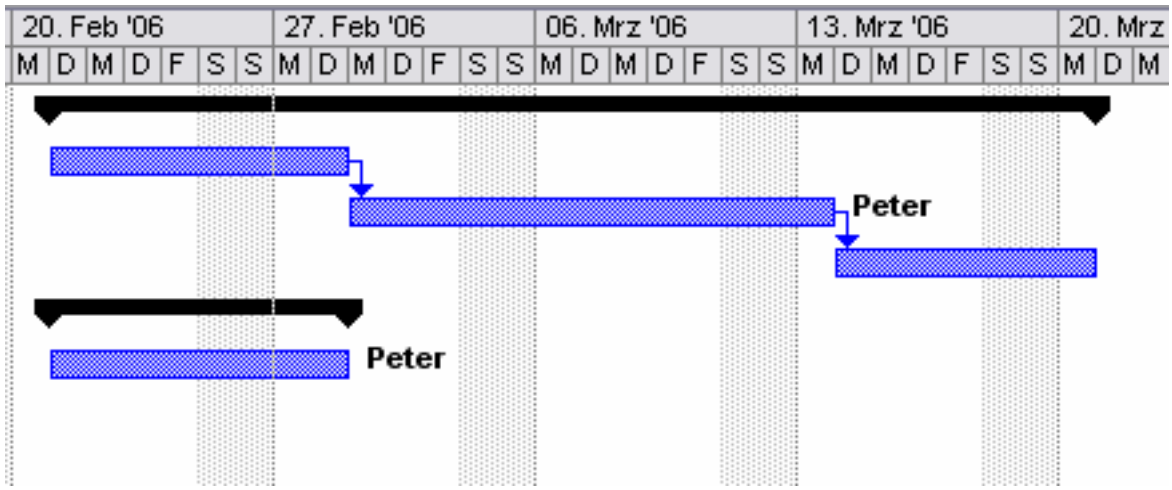


Abbildung 5: Verlängerung der Dauer eines Vorgangs

Aus der Grafik geht hervor, dass der untere Vorgang, welcher der Ressource „Peter“ zugeordnet ist, in den Bereich des Oberen hineinreicht, wenn sich dessen Dauer erhöht. An der gegebenen Grafik kann man weiterhin erkennen, dass ein weiterer Modellierungsschritt zu

einem Ressourcenkonflikt führen kann: die Vorbedingungen zu dem oberen Vorgang werden gelöscht, sodass er an den Anfang des Projektes gesetzt werden kann.

Man könnte an dieser Stelle argumentieren, dass, um einen Konflikt zu vermeiden, der Vorgang einfach nicht an den Projektanfang verschoben werden muss und es entsteht erst gar kein Problem. Jedoch ist darauf anzumerken, dass im Zuge der Optimierung des Projektplans überprüft werden muss, welcher Vorgang verschoben werden muss, um die zeitliche Bearbeitung des Projektes zu optimieren.

Diese einfachen Beispiele sollen illustrieren, welche vielfältigen Möglichkeiten zum Auftreten von Ressourcenkonflikten führen können. Für jeden einzelnen Fall muss geprüft werden, welche Algorithmen geeignet sind den inkonsistenten Zustand aufzulösen.

Ein für den Nutzer nur schwer einsehbaren Zustand entsteht, wenn die bereits beschriebenen Situationen projektübergreifend entstehen, sodass z.B. zwei Vorgänge aus unterschiedlichen Projekten eine zeitliche Überschneidung aufweisen. In diesem Fall ist das nicht ersichtlich für den Nutzer, da dieser lediglich den gerade zu bearbeitenden Projektplan vor Augen hat. Es sind somit dementsprechende Benutzerführungen im Programm zu entwickeln, um gegebenenfalls Unklarheiten zu vermeiden.

Bisher wurden vergleichsweise einfache Modelle vorgestellt. Die folgende Grafik verdeutlicht jedoch, wie sich der Berechnungsaufwand mit der Anzahl der Aufgaben und der beteiligten Personen vergrößert und welche zusätzlichen Nebenbedingungen zu beachten sind.

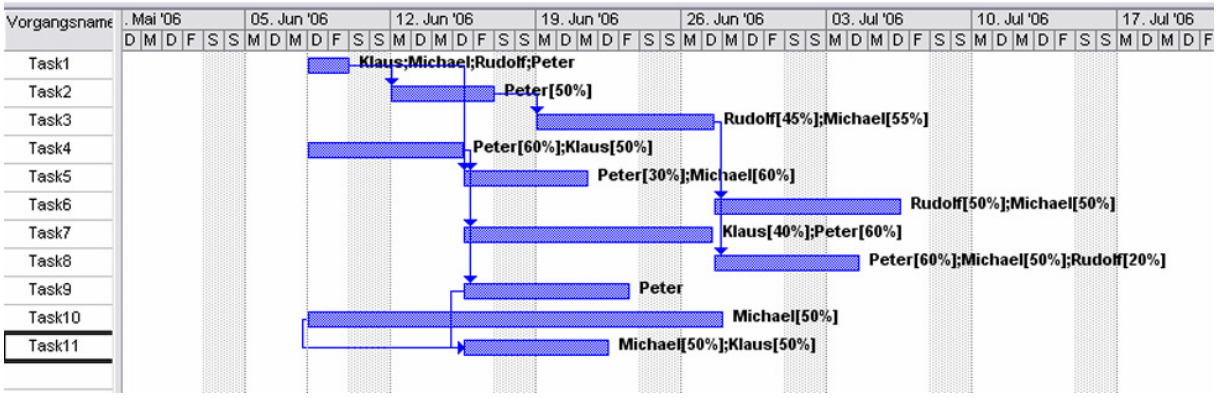


Abbildung 6: Komplexe Ablaufpläne

Auch muss man an dieser Stelle anmerken, dass der Umfang des obigen Projektes noch nicht vergleichbar ist mit den in der Praxis vorkommenden Modellen. Hier sind Netzpläne mit mehreren Hundert Aufgaben sicher keine Seltenheit. Das Modell verdeutlicht jedoch viele Aspekte, die in den zu diskutierenden Algorithmen berücksichtigt werden müssen. Zunächst ist es möglich, eine Ressource gleichzeitig an mehreren Projekten arbeiten zu lassen. Die Beteiligung an einer Aufgabe wird demzufolge vom Modellierer bestimmt. Eine

Überbelastung besteht demnach dann, wenn eine Ressource zu einem bestimmten Zeitpunkt an einem oder mehreren Arbeitspaketen beteiligt ist und seine Gesamtbelastung dabei 100% überschreitet.

Ein weiterer Aspekt sind die Reihenfolgebeziehungen unter den einzelnen Arbeitspaketen. Im unteren Abschnitt der Grafik ist ein Beispiel für eine „Anfang-Anfang“, wie sie in MS Project definiert ist, illustriert. Die „Task11“ steht in Abhängigkeit zu den Aufgaben 9 und 10. Sie kann erst beginnen, wenn die Vorgänger ebenfalls schon gestartet sind. Kommt es nun zum Ressourcenausgleich zwischen den Aufgabenpaketen, so muss diese Aufgabe nur an das Ende der Vorgänger geschoben werden, wenn eine konkrete Ressourcenüberlastung besteht.

So wird klar, dass eine gute Lösung für das Problem immer auch eine hohe Ressourcenbelastung mit sich führt, was für die Unternehmen sicher ein wichtiges Ziel bei der Projektdurchführung darstellt. So können durch gezielte Umverteilung des Personals oder durch den Einsatz unterbelasteter Mitarbeiter weitere Verbesserungen im bestehenden Netzplan erreicht werden. Dies kann nur unter der Berücksichtigung der Qualifikationen der Mitarbeiter erfolgen.

Anhand dieser Ausführungen wird deutlich, welche vielfältigen Aspekte zusammenwirken, wenn ein automatischer, also durch das Programm ausgeführter, Ressourcenausgleich durchgeführt wird.

3.5 The Schedule Generation Schemes (SGS)

Schedule Generation Schemes oder kurz SGS sind der Kern einer jeden heuristischen Berechnung zum RCPSP. Dabei besteht das Grundprinzip dieser Methoden in der schrittweisen Erweiterung bestehender Teillösungen bis hin zur kompletten Berechnung des zugrunde liegenden Projektplans.
Als Teillösung wird eine Menge oder Set von Aktivitäten/Vorgängen verstanden, deren zeitliche Einordnung in den Projektplan auf Basis der verwendeten Methode bereits berechnet wurde. Der Schedule ist somit zu Beginn des Algorithmus leer und wird mit jedem Iterationsschritt um einen Vorgang erweitert, wonach zum Ende der Berechnung alle im Projekt existierenden Vorgänge berücksichtigt wurden. Für einen Iterationsschritt steht eine bestimmte Menge von Aufgaben zur Verfügung, die als Nächstes in den Projektplan eingeordnet werden können. An dieser Stelle kommen nur diejenigen Aktivitäten in Betracht, deren Vorgänger bereits in den Schedule aufgenommen wurden. Ein Auswahlverfahren, hier als Prioritätsregel bezeichnet, steuert nun die Auswahl des nächsten zu berücksichtigenden Vorgangs. Wenn eine Aktivität das Kriterium der Prioritätsregeln unter allen Kandidaten am besten erfüllt, wird sie in die bestehende Teillösung aufgenommen.
Das Resultat des Algorithmus ist ein vollständig berechneter Ablaufplan des Projektes – die Bearbeitungszeit, also tatsächlicher Anfangs- und Endpunkt einer Aufgabe, wurde festgelegt.

In den nachfolgenden Absätzen soll nun das Vorgehen der seriellen und der parallelen Methode des SGS beschrieben werden.

3.5.1 serielle Methode

Um formelle Beschreibung der seriellen Vorgehensweise des SGS zu geben, müssen zunächst folgende ergänzende Notationen definiert werden (Notation nach [KOL]):

C_n das Set der bereits kalkulierten Aktivitäten – complete set,

D_n das Set der zur Berechnung in Frage kommenden Aktivitäten – decision set,

R_n verbleibende Aktivitäten – remaining set,

PS_n zum Teil berechnete Aktivitäten in der Periode t - partial Schedule,

πK_{rt} restliche Kapazität einer erneuerbaren Ressource in der Periode t

Für diese Sets ergeben sich nunmehr logische Mengenvorschriften, wie sie im Folgenden dargestellt sind:

$D_n = \{j \setminus J \notin C_n; P_j \subseteq C_n \}$

$R_n = X \setminus \{ C_n \cup D_n \}$

$PS_n = \{ C_n \}$

$\pi K_{rt} = K_r - \sum k_{jr}$

$j \in A_t(PS_n)$

In dem hier vorliegenden Beispiel wird das grundlegende Prinzip des seriellen SGS beschrieben, welcher die Grundlage für einige Heuristiken bildet. Es wird im folgenden Beispiel angenommen, dass für jede Aktivität ein Prioritätswert existiert, welcher bei einer Entscheidung zwischen mehreren Aktivitäten ausschlaggebend dafür ist, welche Aktivität im nächsten Rechenschritt innerhalb des Netzplans verschoben wird.

WHILE $|PS_n| < J$ DO Stage n

BEGIN

UPDATE D_n, R_n, and πK_{rt}, $t = 1,\ldots,T$, $r \in R$;

$j^* :=$ min {j|v(j) extremum v(i)};

$j \in D_n \qquad i \in D_n$

$PST_{j*} := \max \{FT_i | i \in P_{j*}\};$

$ST_{j*} := \min\{t \mid PST_{j*} \leq t \leq LST_{j*}, k_{j*r} \leq \pi K_{rt}, \tau = t+1,\ldots,t+d_{j*}, r \in R\};$

$FT_{j*} := ST_{j*} + d_{j*};$

$C_{n+1} := PS_{n+1} := C_n \cup \{j*\};$

$n := n+1;$

END

STOP: Eine zulässige Lösung $S=(FT_1,\ldots, FT_j)$ wurde generiert.

Die serielle Methode besteht aus J Schritten. In jedem dieser Schritte wird eine Aktivität ausgewählt und zeitlich festgelegt. Folgende Sets sind für das SGS von Bedeutung:

- Aktivitäten, die bereits zeitlich eingeordnet sind
- vollständige Aktivitäten
- partiell vervollständigte Aktivitäten
- Aktivitäten, die zur Auswahl für den nächsten Schritt stehen
- verbleibende Aktivitäten

In jedem Schritt wird eine Aktivität mit Hilfe einer Prioritätsregel ausgewählt und deren frühest möglicher Startzeitpunkt FAZ in Abhängigkeit mit den Ressourcenrestriktionen ermittelt. Danach wird die Aktivität aus dem Decision-Set entfernt und in das Completed-Set eingefügt.

Nun werden die verbleibenden Aktivitäten überprüft, ob sie in das Decision-Set aufgenommen werden können. Das ist der Fall, wenn alle Vorbedingungen der Aktivitäten erfüllt sind. Der Algorithmus wird beendet, wenn alle Aktivitäten eingeordnet und damit im Completed-Set abgelegt sind.

Im Folgenden soll der Algorithmus anhand eines einfachen Netzplanbeispiels illustriert und die einzelnen Schritte bis zur Erreichung einer zulässigen Lösung aufgezeigt werden.

Das Beispiel in der folgenden Grafik stellt einen einfachen Netzplan mit zwei Hilfsaktivitäten – Quelle als Knoten 1 und Senke als Knoten 7 - dar, welche durch die Dauer 0 und keinen

Ressourcenverbrauch charakterisiert sind. Die Zahl über dem Knoten stellt somit die Dauer und diejenige darunter den Kapazitätsbedarf der Aktivität dar.

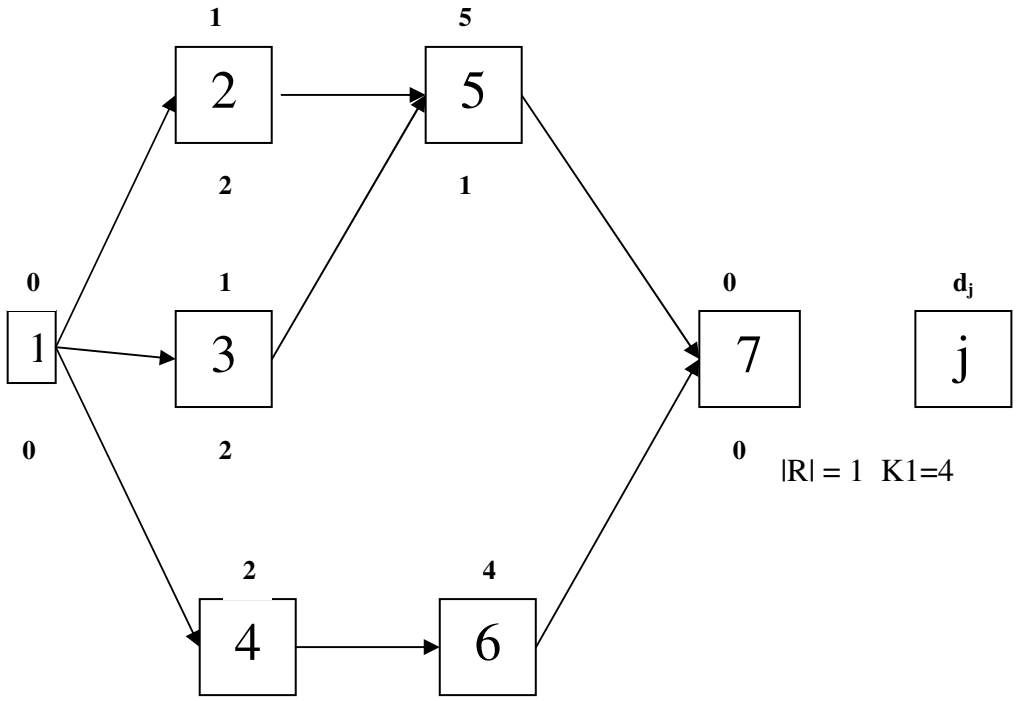


Abbildung 7: Beispielprojekt [KOL S.77]

Um einen Ressourcenausgleich anhand des seriellen SGS zu erläutern, wird in der folgenden Tabelle nun für jeden Knoten eine Prioritätskennzahl eingeführt.

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

Tabelle 1: Prioritätskennzahlen zu dem Beispiel [KOL]

Wenn im Falle einer Konkurrenz zwischen mehreren Aktivitäten um eine Ressource im Verlauf des Algorithmus` eine Aktivität ausgewählt werden soll, ist die Prioritätskennzahl entscheidend. Somit wird anhand des Netzplans bereits ein Beispiel für eine Prioritätsregel gegeben. Weitere Heuristiken zu diesem Thema werden im folgenden Kapitel erläutert.

Die einzelnen Berechnungsschritte sind in der folgenden Tabelle nach [KOL] wiedergegeben.

n	πK_{1t}, $1 \leq t \leq 8$	$C_n=PS_n$	D_n	R_n	v(j)	j*	FT_j*
1	(4,4,4,4,4,4,4,4)	⊘	1	2,…,7	0	1	0
2	(4,4,4,4,4,4,4,4)	1	2,...,4	5,…,7	1,1,2	2	1
3	(2,4,4,4,4,4,4,4)	1,2	3,4	5,…,7	1,2	3	1
4	(0,4,4,4,4,4,4,4)	1,...,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	⊘	6	7	7

Tabelle 2: Berechnungsschritte des seriellen SGS [KOL]

Es ist nun das Ergebnis des Algorithmus` zu überprüfen. Wurde eine zulässige, evtl. optimale Lösung des Problems generiert? Nach Kolisch [KOL] generiert der Algorithmus in jedem Fall eine zulässige Lösung, wenn Anordnungsbeziehungen eingehalten und Ressourcenbedingungen (-kapazitäten) eingehalten werden.

Die Lösung muss nicht immer optimal sein, da zum Einen Nachfolgerbeziehungen nicht beachtet und zum Anderen im Falle von gleichen Prioritätskennzahlen zufällig eine Aktivität für den nächsten Schritt ausgewählt werden kann. Für diesen konkreten Fall ist es jedoch nicht auszuschließen, dass durch Verschieben einer anderen Aktivität mit der gleichen Prioritätskennzahl eine bessere Lösung erzielt werden kann.

3.5.2 parallele Methode

Bei dem parallelen SGS ist es wiederum nötig, Erweiterungen zu definieren, nach [KOL]:

t_n aktuelle Projektzeit – schedule time

A_n Set von Aktivitäten, welche zum Zeitpunkt tn aktiv sind – active set

πK_r verbleibende Kapazität einer erneuerbaren Ressource zur aktuellen Projekt-/ Berechnungszeit

Anhand des Algorithmus` ergeben sich zu jeder Projektzeit der Berechnung folgende Mengenverhältnisse:

$A_n = \{j \mid ST_j \leq t_n < FT_j\}$

$\pi K_r = K_r - \sum_{j \in A_n} k_{jr}$

$D_n = \{j \mid j \notin \{C_n \cup A_n\},\ P_j \subseteq C_n, k_{jr} \leq \pi K_r \ \forall\ r \in R\}$

$R_n = X \setminus \{C_n \cup A_n \cup D_n\}$

$PS_n = \{C_n \cup A_n\}$

Der Algorithmus zum parallelen SGS ist dann wie folgt nach [KOL] definiert:

WHILE $|PS_n| < J$ DO Stage n

BEGIN

(1) $t_n := \min \{FT_j \mid j \in A_{n-1}\}$;

$A_n := A_{n-1} \setminus \{j \mid j \in A_{n-1}, FT_j = t_n\}$;

$C_n := C_{n-1} \cup \{j \mid j \in A_{n-1}, FT_j = t_n\}$;

UPDATE $\pi K_r \ \forall\ r \in R$, D_n, and R_n;

$j^* := \min_{j \in D_n} \{j \mid v(j) = \text{extremum } v(i)\}$; $i \in D_n$

$ST_{j^*} := t_n$;

$FT_{j^*} := ST_{j^*} + d_{j^*}$;

$A_n := A_n \cup \{j^*\}$;

UPDATE $\pi K_r \ \forall\ r \in R$, PS_n, D_n, and R_n;

IF $D_n \neq \varnothing$ THEN GOTO Step (2), ELSE DO

BEGIN

$PS_{n+1} := PS_n$, $n := n+1$;

END;

END;

STOP: Eine zulässige Lösung $S=(FT_1,\ldots, FT_j)$ wurde generiert.

Dieser Algorithmus unterscheidet sich von der seriellen Vorgehensweise nur in einem kleinen Detail. Auch hier werden J Stufen durchlaufen. Wichtig sind hier folgende Sets:

- Aktivitäten, welche bis zum Zeitpunkt des derzeitigen Bearbeitungsschrittes t_n bereits vervollständigt wurden
- Aktivitäten, die bis zum Zeitpunkt t_n bereits berechnet wurden, jedoch noch aktiv sind
- Decision-Set
- Remaining-Set

So werden in jedem Bearbeitungsschritt zunächst diejenigen Aktivitäten entfernt, welche bis zum Zeitpunkt t_n beendet sind. Dies kann die Vorraussetzung für Aktivitäten aus dem Remaining-Set sein, um in das Decision-Set aufgenommen zu werden. An dieser Stelle wird wie in der seriellen Methode vorgegangen. Es wird nach der Prioritätsregel geprüft, ob eine weitere Aktivität in das Active-Set übernommen werden, indem die Ressourcenrestriktionen eingehalten werden, und somit zum Zeitpunkt t_n starten kann.

STEP 2 wird somit solange ausgeführt, bis das Decision-Set leer ist, also alle Aktivitäten im Complete- oder im Active-Set sind.

Der Algorithmus soll nun am bereits verwendeten Netzplanbeispiel *in* Tabelle 3 dargestellt werden.

n	t_n	πK_1	A_n	C_n	PS_n	D_n	R_n	v(j)	j*	FT_{j*}
1	0	4	⊘	⊘	⊘	1	2,…,7	0	1	0
2	0	4	⊘	1	1	2,…,4	5,…,7	1,1,2	2	1
		2	2	1	1,2	3	4,…,7	1	3	1
		0	2,3	1	1,2	⊘	4,…,7			
3	1	4	⊘	1,…,3	1,…,3	4,5	6,7	2,6	4	3
		1	4	1,…,3	1,…,4	5	6,7	6	5	6
		0	4,5	1,…,3	1,…,5	⊘	6,7			

4	3	3	5	1,…,4	1,…,5	6	7	6	6	7
		0	5,6	1,…,4	1,…,6	⊘	7			
5	6	1	6	1,…,5	1,…,6	⊘	7			
6	7	4	⊘	1,…,6	1,…,6	7	⊘	6	7	7

Tabelle 3: Berechnungsschritte des parallelen SGS [KOL]

Anhand der Tabelle wird der Unterschied zur seriellen Vorgehensweise deutlich:

Bei jedem Berechnungsschritt wird ein Zeitintervall vorangegangen. Es wird analysiert, welche Aktivitäten zum jeweiligen Zeitpunkt aktiv sind und wie viel Kapazität diese von der zu betrachtenden Ressource einnehmen. Es wird nun überprüft, welche der in Frage kommenden Aktivitäten, die noch nicht aktiv waren, zum derzeitigen Zeitpunkt aktiv werden können. Das Auswahlkriterium bildet für diesen Fall wiederum die Prioritäten, die in Tabelle 1 dargestellt wurden. Wenn die beanspruchte Kapazität nunmehr unter der Kapazitätsgrenze liegt zum Zeitpunkt t_n , so kann die Aktivität aktiv werden.

Bei der seriellen Vorgehensweise wird nicht in Zeitintervallen vorgegangen, sondern nach der Reihenfolge der Aktivitäten. Auch bei diesem Algorithmus kann gesagt werden, dass, wenn alle Nebenbedingungen eingehalten und Deadlines zunächst ignoriert werden, eine zulässige Lösung generiert wird.

Es wurden nunmehr die grundlegenden Konzepte für den Ressourcenausgleich innerhalb eines Projektnetzplanes vorgestellt. Das serielle und parallele SGS werden in den folgenden Heuristiken immer wieder als Grundlage dienen. Im folgenden Abschnitt sollen nun konkrete Heuristiken erläutert werden, welche die Lösung von Ressourcenkonflikten zur Zielstellung haben.

3.6 Algorithmen zum RCPSP

PSP – Modelle sind, ähnlich wie andere Problemstellungen, so z.B. die Produktionsplanung und Maschinenbelegungsplanung, durch eine hohe Komplexität gekennzeichnet. Dieser Aspekt wurde bereits in dem Kapitel „Komplexitätsbetrachtung“ ausführlich dargelegt. Praxisrelevante Modelle des PS, also mit mehreren hundert Aktivitäten, können nicht durch exakte Lösungsverfahren in vertretbarem Aufwand gelöst werden. Die Komplexitätsbetrachtungen haben zudem verdeutlicht, dass mit effektiven und exakten Algorithmen für das PSP nicht zu rechnen ist.

Jedoch muss an dieser Stelle auf Veröffentlichungen (z.B. Hartmann [Hart], Kuhlmann[Kuhl], …) hingewiesen werden, die exakte Lösungsverfahren vorgestellt haben – i.d.R. Branch & Bound – Verfahren. Für einen praktischen Einsatz sind allenfalls verkürzte Varianten wie das „Truncate B&B"-Verfahren von Interesse.

In der Praxis ist es somit vorrangig das Ziel sehr gute Lösungen zu finden, die in vertretbarer Rechenzeit ermittelt werden können. Dabei kommen unter Anderem verkürzte exakte Verfahren in Frage, als auch geeignete Heuristiken oder Metaheuristiken wie sie in diesem Kapitel zur Sprache kommen sollen.

Eine Heuristik wird im Allgemeinen als Entscheidungsregel verstanden, die, zugeschnitten auf einen bestimmten Problemkontext, nach einem definierten Muster Lösungen generiert. Eine exakte Lösung kann ein solches Verfahren hingegen nicht garantieren. Zum Vergleich ist eine Metaheuristik ein Prozess, bei dem ein übergeordnetes Prinzip der Steuerung eine untergeordnete Heuristik zielgerichtet dirigiert.

3.6.1 Branch & Bound Verfahren

Ist ein Endtermin mit $\overline{T}$ für die Projektlaufzeit vorgegeben, so ist die Lösungsmenge beschränkt und endlich. In einem solchen Fall kann die Lösung durch vollständige Enumeration, so zum Beispiel einem B&B – Verfahren, ermittelt werden.

Beginnend bei der Wurzel, bestehend entweder aus einem „leeren" Schedule, in dem keine Aktivitäten berechnet sind, oder einer vollständigen Lösung, wird ein Suchbaum ermittelt, der so lange weiter in Äste verzweigt (engl. „branching"), bis eine Begrenzungsregel (engl. „bounding") einsetzt. In einem solchen Fall wird normalerweise in den letzten Knoten zurückgesprungen und ein neuer Ast untersucht. Der Algorithmus ist beendet, bis der komplette Baum aufgespannt wurde oder eine optimale Lösung gefunden wurde. Diese kann ggf. durch den Vergleich von „upper and lower bounds" ermittelt werden. Die folgenden Schemata des B&B – Verfahrens wurden bisher auf das PSP angewandt.

- Precedence Tree
- Mode and Delay Alternatives
- Mode and Extension Alternatives
- Scheduling Schemes
- Block Extensions

Im Folgenden soll nur auf ein Verfahren eingegangen werden, um das Vorgehen von Branch and Bound – Verfahren, angewandt auf das RCPSP, zu verdeutlichen. Der Autor bezieht sich auf die Erläuterungen zu dem Precedence-Verfahren nach Brucker und Knust [BruKnu] sowie Sprecher [Sprech94] und Hartmann [Hart].

Das Hauptproblem liegt nach Sprecher im Aufbau des Baumes. Die zugrundeliegende Idee ist die schrittweise Dekomposition des Problems in Teilbereiche, indem gültige Regionen aufgesplittet werden. Dieser Algorithmus bezieht sich auf das List-Scheduling – ein Schedule codiert in einer Liste von Aktivitäten dargestellt.

Der Algorithmus beginnt bei der Dummy-Ressource zum Zeitpunkt 0. In jedem Berechnungsschritt werden die Sets SJ_g mit den derzeit berechneten Aktivitäten und EJ_g mit den noch zu berechnenden Aktivitäten ermittelt. Zu beachten ist, dass nur Aktivitäten zur Berechnung für den nächsten Schritt bereit stehen können, deren Vorgängeraktivitäten bereits berechnet wurden. Es wird nun eine Aktivität bestimmt und deren frühest-mögliche Startzeit berechnet, welche in Abhängigkeit mit den Vorrangsbeziehungen und den Ressourcenrestriktionen möglich ist. Nun kann in den nächsten Berechnungsschritt – Branch – übergegangen werden. Wenn die Senke, also die Dummy- Aktivität, erreicht ist, so wurde ein kompletter Schedule ermittelt. Ist dies erreicht, kommt es zum „backtracking“, d.h. es wird in den vorangegangenen Level zurückgesprungen. So werden nach und nach alle Aktivitäten getestet und jede mögliche Kombination zum Ausgleich des Projektplans durchgerechnet. Am Ende gibt es einen Schedule der alle anderen Lösungen hinsichtlich der Zielfunktion dominiert und somit das Optimum vorgibt.

Wie bereits erwähnt, sind exakte Methoden wie der Branch & Bound Algorithmus schon ab einer kleinen Anzahl von Aktivitäten mit extrem langen Rechenzeiten und einem überforderten Arbeitsspeicher verbunden. Hartmann [Hart] gibt als Alternative einen Truncated Branch & Bound vor, der eine optimale Lösung nicht garantieren kann. Es wird während der Berechnung abgebrochen, bevor eine optimale Lösung gefunden oder eine Prüfung der besten Ergebnisse abgeschlossen wurde.

Hartmann verweist in seinem ersten Beispiel auf Pollack-Johnson [PolJohn]. Das Vorgehen orientiert sich an der parallelen SGS. Anstatt die Aktivität mit dem höchsten Prioritätswert als nächstes einzulasten, wird der Baum nach bestimmten Kriterien weiterverzweigt. So ist es denkbar, dass beispielsweise eine Verzweigung die Aktivität mit dem höchsten und eine weitere die Aktivität mit dem zweitgrößten Prioritätswert berücksichtigt.

Eine andere Vorgehensweise erörtert Sprecher [Sprech96]. Hiernach wird der Branch & Bound Algorithmus nach einer bestimmten Rechenzeit abgebrochen. Sprechers Vorschlag orientiert sich an dem „Precedence Tree“, wie er im vorangegangenen Kapitel zur Sprache kam. Es wird jedoch schon an früher Stelle, während des Aufspannens des Baumes, versucht,

gute Lösungen zu finden, indem viel versprechende Prioritätsregeln angewandt werden, welche die Aktivitäten mit dem höchsten Prioritätswert herausfiltern und diese zuerst berechnen.

3.6.2 Prioritätsheuristiken

Es wird zunächst zwischen Prioritätsregeln im Ein- und Mehr-Projektfall unterschieden. Darüber hinaus werden Methoden vorgestellt, welche für den Mehr-Projekt-Fall in Frage kommen, da sich das Software-Produkt ebenfalls an diesem orientiert.

Allgemein können diese Heuristiken auf formelle Weise zusammengefasst werden. In jedem Fall gilt, dass eine Aktivität aus dem Decision-Set mit dem Maximum- oder Minimum- Wert eines Prioritätskriteriums ausgewählt wird. Dies kann formell wie folgt beschrieben werden:

$$J^* = \min\{j \mid v(j) = \text{extremum } v(i)\}$$

$$j \in D_n \qquad i \in D_n$$

So kann auch die erste Heuristik an dieser Stelle, die Kürzeste-Bearbeitungszeit-Heuristik (Shortest processing time - SPT) wie folgt beschrieben werden:

$$v(j) = d_j \qquad \text{extremum} = \min$$

Es kann nunmehr eine weitere Unterteilung der Heuristiken herangezogen werden. So ist zwischen Netzplan-, Zeit- und Ressourcen- basierten Heuristiken zu unterscheiden. Jede dieser Klassifikationen soll im Folgenden anhand einer Beispielheuristik definiert werden.

„Most immediate successors (MIS)“ – Netzplan- basiert

$$v(j) = |S_j| \qquad \text{extremum} = \max$$

Es wird diejenige Aktivität aus dem Decision-Set ausgewählt, die die meisten direkten Nachfolger aufweist.

„Latest start time (LST)“ – Zeit-basiert

$$v(j) = LST_j \qquad \text{extremum} = \min$$

Diejenige Aktivität geht in die Netzplanberechnung ein, welche die niedrigste späteste Startzeit aufweist.

„Greatest resource demand (GRD)“ – Ressourcen-basiert

$$v(j) = dj \sum k_{jr} \quad \text{extremum} = \max$$

Prioritätsregeln, welche unabhängig vom Berechnungsschritt für eine bestimmte Aktivität stets den gleichen Wert ermitteln, werden als statisch bezeichnet, wohingegen unterschiedliche Werte auf eine dynamische Heuristik verweisen. So ist zu sagen, dass MIS und GRD als statisch und beispielsweise die „minimum slack“-Heuristik als dynamisch klassifiziert werden kann.

Des Weiteren können Heuristiken auch nach dem Input an Informationen unterteilt werden. Algorithmen, die mit einem geringen Anspruch von Informationen auskommen, werden als lokal, wohingegen umfangreichere netzplan- umfassende Inputs als global bezeichnet werden. So kann die SPT-Heuristik auch als lokal eingegliedert werden. Ein Beispiel für eine globale Heuristik ist Greatest rank positional weight (GRPW). So können Prioritätsheuristiken nach Einzelkriterien als auch Verbundprioritäten, z.B. gewichtete Summe mehrerer Prioritätenregeln, unterteilt werden.

Als letztes Kriterium, welches an dieser Stelle erwähnt werden soll, ist die Vorgehensweise nach unteren oder oberen Grenzen. Die LST-Heuristik arbeitet beispielsweise mit einer unteren Grenze, wohingegen MIS diejenige Aktivität auswählt, welche die meisten direkten Nachfolger aufweist.

Bisher wurde auf single-pass-Methoden eingegangen, was bedeutet, dass die entsprechende Heuristik eine einfache Berechnung vornimmt - es wird eine SGS-Methode angewendet und anhand einer spezifischen Prioritätsregel eine mögliche Lösung errechnet. Ein Algorithmus, welcher hingegen mit mehreren Berechnungsabläufen arbeitet, wird als multi-pass-method bezeichnet.

So können die SGS-Methoden und Prioritätsregeln beliebig zu einer neuen Heuristik kombiniert werden. Am gebräuchlichsten nach Hartmann [HART] sind dabei die Multi-Prioritätsregeln, die Vorwärts-Rückwärts-Berechnungsmethoden, Sampling – Methoden und Adaptive Methoden. Die Multi-Prioritätsregeln durchlaufen das SGS mehrere Male. Bei jedem Durchlauf wird eine andere Prioritätsregel genutzt. Oft werden laut Hartmann [HART S.67] die Prioritätsregeln absteigend nach deren Ergebnisqualität, ermittelt aus statistischen Untersuchungen, durchlaufen. Alternativ kann durch Konvexkombination von n Prioritätsregeln eine viel größere Anzahl von Berechnungsdurchläufen generiert werden. Folgende Formel definiert diese Möglichkeit:

$$v(j) = \sum_{i=1}^{n} \Omega_i * v_i(j) \text{ mit } \Omega_i \geq 0 \text{ für alle i und } \sum_{i=1}^{n} \Omega_i = 1$$

Vorwärts- und Rückwärts- Berechnungsmethoden durchlaufen ein SGS, indem sie iterativ vorgehen und während der Berechnung abwechselnd eine Vorwärts- und Rückwärtsberechnung durchführen.

3.6.3 Sampling Algorithmen

Sampling bedeutet übersetzt soviel wie „ausprobieren“ oder „testen“. Anhand dieser Übersetzung ist die Vorgehensweise schon fast erklärt und bedarf weniger Erläuterungen.

Nach Kolisch [KOL] werden Zufalls-Anweisungen (random-devices) genutzt, um eine bestimmte Anzahl von Z Berechnungen zu erstellen. Im Anschluss wird die beste Berechnung ausgewählt und als Lösung präsentiert. Die entscheidenden Fragen sind jedoch: wie groß ist Z zu wählen, welche random-devices sind zu definieren, um aus dem Decision-Set eine Aktivität auszuwählen?

Es ist festzuhalten, dass mit einer wachsenden Anzahl von Beispielberechnungen auch die Wahrscheinlichkeit steigt, eine bessere Berechnung als die bisher Ermittelte zu finden. [KOL] beschreibt weiter, dass die Anzahl abhängig von dem SGS ist, welches verwendet wird:

„The decision on the sample space depends on the scheduling scheme employed. The serial method provides the set of active schedules as population while the parallel method restricts the sample space to the set of non-delay schedules“.

Die Verwendung von Zufalls-Anordnung kann als Mapping interpretiert werden:

$$\Psi : j \in D_n \rightarrow [0,1]$$

Diese Vorschrift meint, dass jeder Aktivität im Decision-Set eine Wahrscheinlichkeit Ψ zugeordnet wird, wonach sie im nächsten Schritt ausgewählt wird. Kolisch [KOL] definiert weiter 3 Beispiel-Varianten zu dieser generellen Vorgehensweise:

random sampling, biased random sampling, regret-based biased random sampling

Die erste Methode – random sampling – weist jeder Aktivität aus dem Decision-Set die gleiche Wahrscheinlichkeit zu. Die folgende Formel gibt diesen Sachverhalt wieder (siehe Kolisch S.98):

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

Biased random sampling, wobei bias in diesem Zusammenhang mit Tendenz oder Ausrichtung übersetzt werden kann, ist eine sensiblere Vorgehensweise. Hiernach wird denjenigen Aktivitäten, basierend auf einer bestimmten Prioritätsregel, welche einen höheren Erfolg versprechen, auch eine höhere Wahrscheinlichkeit zugeordnet. Die nachfolgenden Formeln unterscheiden zwischen max/min- Prioritätswerten v(j):

$$\psi(j) = \frac{v(j)}{\sum_{i \in D_n} v(i)} \qquad \text{wenn extremum = max}$$

$$\psi(j) = \frac{\frac{1}{v(j)}}{\sum_{i \in D_n} \frac{1}{v(i)}} \qquad \text{wenn extremum = min und v(j)} \neq 0$$

Dabei offenbaren sich an dieser Stelle zwei Nachteile: Zum Einen ist es nur möglich, lediglich Prioritätswerte zu verwenden die größer als 0 sind → $v(j) > 0 \ \forall \ j \in D_n$. Zusätzlich wird nach [KOL] bei gleich großem v(j) das Sampling unabhängig von der Prioritätsregel durchgeführt. Man kann nur spekulieren, dass Kolisch an dieser Stelle die Nachfolger-Aktivitäten meint, welche ebenfalls bestimmte Prioritätswerte besitzen und in diesem Fall nicht in die Bewertung und damit in das Sampling mit eingehen.

Um diese Nachteile zu umgehen, verweist Kolisch [KOL S.98f] an dieser Stelle auf Alvarez-Valdez/Tamarit (1998b) die den vorliegenden Algorithmus für Minimalwerte - extremum = min - aufgreifen und wie folgt definierten:

$$\psi(j) = \frac{T - v(j)}{\sum_{i \in D_n} (T - v(i))} \qquad \text{wenn extremum = min T}$$

Kolisch [Kol] stellt noch weitere Möglichkeiten vor, dem Autor ging es in erster Linie nur darum, die allgemeine Vorgehensweise vorzustellen.

Die letzte Methode erweitert die eben erläuterte um so genannte „regret measures“. Kolisch [KOL] referiert hier auf Drexl (1991) und Drexl/Grünewald (1993), die diesen Algorithmus erstmals eingeführt und beschrieben haben. Der Regret-Wert ρ_j vergleicht somit den Prioritätswert einer Aktivität mit dem schlechtesten Wert aus dem Decision-Set:

$$\rho_j = \begin{cases} \max v(i) - v(j), \quad i \in D_n & \text{wenn extremum = min} \\ v(j) - \min v(i), \quad i \in D_n & \text{wenn extremum = max} \end{cases}$$

So kann das parametrisierte Wahrscheinlichkeits- Mapping wie folgt definiert werden:

$$\psi(j) = \frac{(\rho_j + 1)^\alpha}{\sum_{i \in D_n} (\rho_i + 1)^\alpha}$$

Indem man die Konstante „1“ zu ρ_j hinzufügt, wird garantiert, dass die Wahrscheinlichkeit für jede Aktivität aus dem Decision-Set größer als 0 ist. Mit dem Parameter α kann der bias-Wert kontrolliert werden. Mit einem hohen Wert α existiert kein bias-Wert und die Wahrscheinlichkeit der Aktivitäten tendieren gegen 0.

Kolisch [KOL] fand heraus, dass ein Wert $\alpha = 1$ gute Resultate erzielt.

3.7 Genetische Algorithmen

Unter dem Absatz Metaheuristiken wurde bereits der Begriff Genetischer Algorithmus angesprochen. In diesem Absatz soll nunmehr erläutert werden, was darunter zu verstehen ist. Zunächst soll die grundlegende Vorgehensweise und im nächsten Schritt die unterschiedlichen Ausprägungen dieser Algorithmen-Gruppe erläutert werden (siehe Hartmann [Hart S.83ff]).

Angelehnt an die anerkannte Lehre der Evolutionstheorie von Charles R. Darwin und der später weiterentwickelten Vererbungstheorie von Gregor Mendel finden sich viele Begriffe wieder, die in Zusammenhang mit dem GA angesprochen werden. Ein Individuum oder Phänotyp vereinigt Merkmale und Charakteristiken, welche in seinen Genen verankert sind und ihm von den Eltern weitergegeben wurden. Dabei kommt es zur Kombination und gegebenenfalls Mutation der elterlichen Gene, was zur Folge hat, dass das Kind mehr oder weniger an seine Umgebung angepasst ist. Im Laufe der Evolution, so Darwin, ist es demnach nur den Individuen und Spezies gelungen zu überleben, welche ausreichend an Ihre Umgebung angepasst waren. Im Folgenden soll nun erörtert werden, wie sich dieses Prinzip in den Kontext des RCPSP einordnen lässt.

3.7.1 Vorgehensweise

Genetische Algorithmen wurden von H. J. Holland in seinem Werk „Adaption in natural and artifical Systems“ im Jahre 1975 entwickelt. Sie integrieren sich in Felder wie die kontinuierliche und diskrete Optimierung, Maschinelles Lernen und die Spieltheorie.

In einer ausreichend großen Anzahl von Generationen ist die Wahrscheinlichkeit, für einen definierten Problembereich eine optimale Lösung zu finden, groß.

Basierend auf ein bestimmtes Optimierungsproblem repräsentiert ein Individuum immer eine Lösung des Problems verbunden mit einer bestimmten Güte, die diese Lösung repräsentiert.

Eine Population kann demzufolge als ein Set von Lösungen oder Individuen verstanden werden. Die Stärke eines Individuums wird nunmehr objektiv anhand einer definierten Funktion ermittelt, die besagt, wie gut diese ermittelte Lösung in Bezug auf einen bestimmten Zielkontext ist. So kann man sagen, dass eine Lösung nach verschiedenen Kriterien auch

unterschiedliche Gütemerkmale aufweist. Um zurück auf das Projektmodell zu kommen, muss ein vergleichsweise kurzer Projektplan nicht immer kostenoptimal sein.

Mithilfe eines Crossover-Operators wird nun ein Elternpaar, also zwei „fitte“ Individuen, herangezogen, um daraus eine neue Lösung bzw. Kindelement zu ermitteln. Ein Mutations-Operator kann darüber hinaus weitere Lösungsmerkmale generieren, die durch eine einfache Kombination durch die Elterngene nicht entstanden wären. Somit ist auch ein gewisser Zufallsfaktor vorhanden, der die Findung einer globalen Lösung des Problems weiter vorantreibt.

Der genetische Algorithmus generiert somit im ersten Schritt eine Population mit einer vorgegebenen Größe von POP Individuen, wobei POP eine beliebige Integerzahl ist. Diese Individuen können durch ein Eröffnungsverfahren ermittelt werden, wie zum Beispiel eine Prioritätsregel. Infolgedessen werden aus dieser Population Paare zufällig ermittelt, welche wiederum durch den Crossover-Operator zwei neue Individuen hervorbringen. Der Genotyp dieser Kinder wird nun durch Mutation weiter vermischt. Die Größe der Population ist nunmehr auf 2*POP angestiegen und mithilfe der Fitnessfunktion wurde jedem Individuum ein Kennwert zugeordnet, welcher wiedergibt inwieweit sich diese Lösung für das zu lösende Problem eignet. Nun werden die schlechtesten Lösungen aus der Population entfernt, sodass diese wieder ihren ursprünglichen Umfang erhält.

Dieses Vorgehen wird nun solange wiederholt, bis eine vordefinierte Anzahl von Generationen oder ein Zeitlimit erreicht ist. Somit wurde im Laufe der Berechnungen eine Anzahl von POP*GEN Individuen erzeugt.

Diese grundlegende Vorgehensweise kann nunmehr durch folgenden Pseudocode nochmals zusammengefasst werden. *CHI* bezeichnet dabei die Liste von Individuen, welche in der aktuellen Generation hervorgebracht wurden.

Algorithmus:

G:=1

Generiere eine Initial-Population POP;

Ermittle die Fitnesswerte für jedes Individuum;

WHILE G<GEN AND das Zeitlimit ist noch nicht erreicht DO

BEGIN

G:=G+1;

Ermittle Kinder CHI aus POP durch Crossover;

Wende die Mutation auf die ermittelten Kinder I ∈ CHI;

Ermittle die Fitnesswerte für jedes Individuum I ∈ CHI;

POP:= POP ∪ CHI;

Reduziere die Population durch das Eliminieren der schwächsten Insdividuen;

END.

3.7.2 Activity List – GA

Eine Form eines Genetischen Algorithmus`, der an dieser Stelle vorgestellt werden soll, ist der „Activity List"-GA.

Individuen werden durch eine Liste von Aktivitäten repräsentiert:

$$\lambda = (j_1, j_2, \ldots, j_J).$$

Jeder dieser ermittelten Genotypen ist verbunden mit einer Netzplanberechnung (Phänotyp), die nach Muster der seriellen SGS vorgenommen wird. Die erste (Dummy-)Aktivität startet zum Zeitpunkt 0. Im Anschluss wird für jede Aktivität der früheste mögliche Beginn errechnet.

Die Fitness des ermittelten Individuums kann nun beispielsweise die Zeitspanne des gesamten Netzplans oder die daraus resultierenden errechneten Kosten, welche z.B. durch Überschreiten von Meilensteinen entstehen, sein. Dabei ist zu beachten, dass auf der einen Seite ein Individuum immer von einer eindeutigen Berechnung (Schedule) abgeleitet wird, wohingegen ein Schedule mehreren Individuen zugeordnet werden kann. Das bedeutet: Wenn zwei Aktivitäten zur gleichen Zeit beginnen und parallel verlaufen, so ändert ein Austausch dieser nichts an der Berechnung selbst, lediglich die Anordnungsbeziehungen und damit auch der Genotyp sind verschieden. Beide Individuen haben somit den gleichen Fitnesswert. Im nächsten Iterationsschritt (neue Generation) kommt es wieder zu einer neuen Durchmischung des Genotyps und an dieser Stelle kann ein derartiger Unterschied von Bedeutung sein.

Die Kreuzung der Elternpaare kann wiederum auf unterschiedliche Weise erfolgen. Ein „One-Point-Crossover" vollzieht sich wie folgt:

Zunächst werden eine Mutter mit $\lambda^M = (j_1{}^M, j_2{}^M, \ldots, j_J{}^M)$ und ein Vater mit $\lambda^V = (j_1{}^V, j_2{}^V, \ldots, j_J{}^V)$ ermittelt. Ein Zufallswert q vom Typ Integer, wobei gilt: $1 \leq q < J$, ist nun entscheidend für die Ausprägung der Tochter $\lambda^T = (j_1{}^T, j_2{}^T, \ldots, j_J{}^T)$ und des Sohnes $\lambda^S =$

$(j_1^S, j_2^S, \ldots, j_J^S)$. Die Positionen i = 1, …, q in λ^T werden demnach von der Mutter übernommen, der Rest i= q + 1, …, J geht vom Vater in die Berechnung ein. Dabei werden diejenigen Aktivitäten, die bereits aus der Anordnung der Mutter stammen, nicht noch einmal in die neue Liste eingeordnet. Die Anordnung der Aktivitäten in der Sohn-Liste wird, analog dazu, in umgekehrter Reihenfolge vorgenommen.

Ein praktisches Beispiel soll dieses Prinzip nochmals verdeutlichen:

$\lambda^M = (1,4,6,2,5,3)$ daraus ergibt sich bei einen q=3: $\lambda^T = (1,4,6,3,2,5)$

$\lambda^V = (3,2,6,1,5,4)$ $\lambda^S = (3,2,6,1,4,5)$

In den jeweiligen Listen ist somit jede Aktivität einmal enthalten. Ausgangspunkt für die Berechnung war, dass alle Reihenfolgebeziehungen in den Elternelementen erfüllt und somit zwei gültige Lösungen vorhanden waren. Es stellt sich die Frage ob in den neu ermittelten Individuen diese Bedingungen ebenfalls erfüllt sind. Der Autor verzichtet an dieser Stelle auf diesen Beweis und verweist an [HART] S.88. Darin wird gezeigt, dass die Reihenfolge gewahrt bleibt.

Bei dem Two-Point-Crossover gibt es zwei Faktoren q_1 und q_2 mit $1 \leq q_1 < q_2 < J$.

Dabei wird ähnlich zu den vorangegangenen Erläuterungen wie folgt vorgegangen:

I = 1,…, q_1 leitet sich wiederum von den entsprechenden „Mutter-Genen“ ab, i = q_1 +1, …, q_2 wird vom Vater übernommen und i = q_2 +1, …, J geht wiederum zurück auf die Mutter. In dem vorangegangenen Beispiel ergibt sich für q_1 = 1 und q_2 = 3 die folgenden Lösungen:

$$\lambda^T = (1,3,2,4,6,5) \text{ und } \lambda^S = (3,1,4,2,6,5)$$

Im nächsten Schritt werden die ermittelten Lösungen nochmals durch den Mutations-Operator bearbeitet. Das bedeutet, dass alle Positionen j_i und j_{i+1} in einer Liste i = 1,…, J-1 mit einer Wahrscheinlichkeit von $p_{mutation}$ getauscht werden. Dabei wird vor jeder Tauschoperation geprüft, ob beide Positionen nach den Reihenfolgebeziehungen überhaupt vertauscht werden können. Das Resultat ist wiederum eine gültige Lösung. Wie vorangegangen erläutert, muss eine Mutation nicht zwangsläufig eine Veränderung in der Berechnung des Projektes nach sich ziehen.

Zuletzt muss entschieden werden, welche Individuen für die nächste Generation erhalten werden. Auch hier gibt es verschieden Möglichkeiten, damit am Ende eines Iterationsschrittes wieder POP Individuen zur Verfügung stehen. Das Ranking-Verfahren ermittelt auf die einfachste Weise die besten Lösungen aus der Population und entfernt den Rest. Auch ist es möglich, durch eine Tournament-Selektion immer jeweils zwei Individuen gegeneinander antreten zu lassen und das Schwächere wird in jedem Schritt aus der Population gestrichen.

3.8 Handlungsalternativen

Neben den Ressourcenkonflikten, welche an späterer Stelle genauer besprochen werden, sind Terminkonflikte die Herausforderung in der Projektplanung und -durchführung. In fast jedem Projekt in der Praxis kommt es im Laufe der Durchführung irgendwann zu einem Terminengpass. An dieser Stelle müssen effektive Analysemethoden und Handlungsalternativen vorhanden sein, um für den speziellen Fall eine Lösung herbeiführen zu können. Der Vorteil einer Projektplanungssoftware, hier speziell SimProQ, ist zunächst die schnelle Visualisierung der Projekte, sodass der Nutzer schnell erkennen kann, an welcher Stelle Terminverzögerungen eintreten werden, wenn nicht rechtzeitig geeignete Mittel zum Einsatz kommen. Vor allem der Projektmanager und die Verantwortlichen der einzelnen Prozessbausteine haben einen umfangreichen Überblick für ihren Arbeitsbereich.Wenn ein Terminkonflikt anhand der Visualisierung in der Software erkannt wurde, geht es im nächsten Schritt daran, mögliche Lösungen auf ihre Relevanz für den konkreten Fall zu untersuchen und den Anwender in seiner Entscheidungsfindung zu unterstützen.

Dabei müssen folgende Handlungsmöglichkeiten untersucht werden:

- die Vorgangsdauer muss verkürzt werden – was in den meisten Fällen Überstunden bedeuten würde.
- die Zuordnungen und Abhängigkeiten müssen untersucht werden, um zu überprüfen, ob es in dieser Richtung Möglichkeiten zur Ummodellierung gibt.
- Deadlines müssen überdacht und deren Konsequenzen analysiert werden.
- unter Umständen ist es möglich, Teilvorgänge parallel durchzuführen (Splitting und damit Detaillierung der Arbeitspakete), um dadurch Zeit zu sparen
- eine Umstrukturierung kann wieder durch den Modeller durchgeführt und im Anschluss durch die Netzplanberechnung neu kalkuliert werden

Die Analyse der Ressourcenzuordnung kommt an dieser Stelle in der Praxis in den meisten Fällen in Betracht. Es ist ersichtlich, dass Vorgänge nicht ohne Weiteres einfach gekürzt werden können, ohne dass dies direkte Auswirkung auf die Ressourcen hat.

Änderungen sollten jedoch in jedem Fall durch eine Notiz festgehalten werden, sodass im nächsten Schritt bei der Ressourcenzuordnung nicht die nächsten Probleme entstehen. Darüber hinaus ist es wichtig, dass bei Änderungen der Projektstruktur oder dem Verlegen von Deadlines entsprechende Dokumente hinterlegt werden, um in der letzten Phase des Projekts, der Nachbereitung, Fehler nochmals durchgehen und für spätere Projektvorhaben auszuschließen zu können. Auch im späteren Verlauf des Projektes ist es wichtig, zu jeder Strukturänderung eine informelle Übersicht über deren Konsequenzen auf spätere Arbeitspakete und den Verlauf des Projektes zu haben. In SimProQ werden in diesem Fall die betroffenen Arbeitspakete mit Notizen oder Office-Dokumenten verknüpft.

4 Überführung in die Praxis

4.1 Anwendungsfälle aus der Praxis

In diesem Kapitel sollen praxisbezogene und aus realweltlichen Gesichtspunkten relevante Probleme beschrieben werden, die sich auf Modelle des Project Scheduling übertragen lassen. Dazu formuliert Brucker und Knust [BruKnu] die folgenden Beispiele. Dies soll an erster Stelle verdeutlichen, welche komplexen Erweiterungen zu diesem Problem denkbar sind und wie schnell sich diese ergeben können. Zum Anderen soll durch diese Darstellung ein erster Rückschluss auf die Anforderungen aus der Praxis gezogen und somit der Bogen zu Project-Management-Tools und deren Anspruch an theoretische Modelle des Project Scheduling gespannt werden. Dazu stellt Kuhlmann [Kuhl] einen Anforderungskatalog zusammen, welcher im Unterkapitel 4.2 vorgestellt werden soll.

4.1.1 Ausgewählte Beispiele

Als erstes soll das Cutting – Stock – Problem zur Sprache kommen, welches wie nachfolgend dargestellt auf die Modelle des Project – Scheduling übertragen werden kann.

Materialien wie Papier oder Textilien werden in der Praxis in Standardrollen hergestellt. Diese müssen in Abhängigkeit mit der Kundennachfrage in kleinere Rollen geschnitten werden, sodass am Ende die Anzahl der so erhaltenen Rollen möglichst hoch ist und der Verschnitt minimiert wird. Dieses Problem kann mit Hilfe des RCPSP dargestellt und gelöst werden. Dabei wird eine einzige Ressource mit $R_1 = W$ Einheiten definiert. Die Aktivitäten $i = 1,...,n$ entsprechen dabei den Rollen, die zurecht geschnitten werden sollen. Die Aktivität i hat eine Arbeitszeit von $p_i = 1$ und nutzt $r_{i1} = w_i$ Einheiten der Ressource, wobei w_i die Größe der Rolle angibt.

Das nächste Beispiel wird von Brucker und Knust [BruKnu] als das „High school timetabling“ – Problem bezeichnet. Demnach sind in der Basiskonfiguration die folgenden Variablen beschrieben:

Es sind eine Menge von m Klassen $c_1,..., c_m$, eine Auswahl von h Lehrern $A_1,... ,A_h$ und T Lehrperioden mit $t = 1,...,T$ gegeben. Darüber hinaus ist eine Anzahl von Vorlesungen vorgegeben, welche wiederum mit einem bestimmten Lehrer und einer Klasse verknüpft sind. Es ist zulässig, dass ein Dozent beispielsweise nur in bestimmten Zeitperioden zur Verfügung steht, was das Problem um eine weitere Restriktion erweitert. Das Ziel des Problems ist nun, einen gültigen Stundenplan zu erstellen und die Vorlesungen auf bestimmte Zeiträume zu verteilen. Dabei müssen weitere Restriktionen beachtet werden: So können eine Klasse und ein Dozent zu einem bestimmten Zeitpunkt auch nur an einer Vorlesung teilnehmen.

Das RCPSP muss im Zuge der formulierten Beschränkungen somit um zeitabhängige Ressourcenprofile erweitert werden. Jede Vorlesung wird als Aktivität definiert. In der theoretischen Darstellung des Problems wird ein gültiger Schedule ermittelt durch $C_{max} \leq T$, in der Praxis jedoch ist es möglich, dass weitere Anforderungen zur Geltung kommen. So ist es denkbar, dass für jeden Dozenten oder jede Klasse die Anzahl der Unterrichtsstunden nach unten und oben begrenzt sind. Auch ist es aus Sicht der Praxis unumgänglich, dass ebenfalls Raumpläne in die Betrachtung aufgenommen werden müssen. So sind an vielen Universitäten speziell ausgestatte Räume wie Computerlabore permanent ausgebucht. Auch ist es sinnvoll, dass Vorlesungen mit weiterführenden Veranstaltungen wie Übungen oder Seminaren nicht zur gleichen Zeit stattfinden sollten.

Das letzte praktische Problem was an dieser Stelle zur Sprache kommen soll, wird unter Brucker und Knust als das „Sports League scheduling"- Problem beschrieben.

Diese Problemklasse ist wieder im Standardfall durch 2n unterschiedliche Mannschaften i = 1,...,2n ausgezeichnet. Diese Mannschaften spielen jeweils eine Saison mit Hin- und Rückrunde, wobei jedes Team gegen alle anderen zweimal antritt – in einem Heim- und einem Auswärtsspiel. So besteht jede Serie aus 2n-1 Runden, sodass jedes Team ein Spiel pro Runde zu absolvieren hat. Die Rückrunde wird gewöhnlich in Abhängigkeit zur Hinrunde geplant, sodass eine komplementäre Abfolge der Spiele mit umgekehrtem Heimrecht erstellt wird. Es ist daher unumgänglich, dass Anforderungen aus der Hinrunde auf entsprechende Gegebenheiten aus der Rückrunde abgestimmt werden müssen und umgekehrt. Wenn beispielsweise ein Stadium zu einem Zeitpunkt nicht zur Verfügung steht, so muss das Heimspiel in der komplementären Gegenrunde stattfinden. Ein solches Planungsproblem lässt sich mit Hilfe eines Multi-Modus RCPSP mit zeitabhängigen Ressourcen modellieren, solange eine Halbserie mit n(2n-1) Spielen geplant werden soll. Die Begegnungen zwischen den Teams (i,j) lassen sich wiederum auf die Aktivitäten übertragen, welche in zwei verschiedenen Modi geplant werden können (in Modus H, wonach das jeweilige Spiel im Heimstadium von Mannschaft i stattfindet, oder in Modus A im Stadium von Team j). Alle Aktivitäten haben eine Bearbeitungszeit in jedem Modus. Es werden 2n Mannschafts-Ressourcen modelliert mit einer konstanten Kapazität von 1, sodass garantiert werden kann, dass jedes Team nur ein Spiel pro Runde absolvieren wird. Eine Aktivität, die einem Spiel (i,j) zugeordnet wird, beansprucht eine Einheit der zwei Team-Ressourcen (i,j).

Das hier dargestellte Modell kann ebenfalls durch weitere Restriktionen erweitert werden. So muss man in der Praxis berücksichtigen, zu welchen Zeiten Stadien u.U. durch andere Veranstaltungen wie z.B. Konzerte beansprucht werden. Auch sollte berücksichtigt werden, dass Mannschaften, die in enger räumlicher Nähe zueinander gelegen sind, möglichst nicht alle an einem Spieltag im Heimstadion gegeneinander antreten. Des Weiteren ist zu beachten, dass starke Spiele möglichst gleich verteilt auf die Saison stattfinden, sodass die Attraktivität

über die gesamte Saison beibehalten wird. So lassen sich auch noch weitere Anforderungen formulieren, die in der Praxis von Bedeutung sind. Für das zu konfigurierende Modell lassen sich somit weitere Komponenten für die Zielfunktion formulieren. Es sollte vermieden werden, dass Mannschaften in mehreren Spielen hintereinander im Heimstadion oder auswärts spielen müssen, oder der Spielplan unausgeglichen anspruchsvolle Gegner in kurzen Zeitabständen vorsieht. Es ist ersichtlich durch alle diese Anforderungen, dass es sich um ein sehr komplexes System handelt, welches berechnet werden muss und dass in der Praxis mitunter scheinbar nicht zu erfüllende Ansprüche miteinander vereint werden müssen.

4.1.2 Machine Scheduling

Ein weiteres wichtiges Teilgebiet des Scheduling, welches ebenfalls mit Hilfe des RCPSP bearbeitet werden kann, ist das bereits angesprochene Machine Scheduling. Kuhlmann [Kuhl], als auch Brucker und Knust sprechen diese Thematik in ihren Publikationen an. Im Folgenden sollen die wichtigsten Ausprägungen dieses Problembereichs in kompakter Weise zusammengefasst werden.

Das „Single Maschine Scheduling“ stellt die einfachste Ausprägung dieser Modelle dar. Es sind n Aufgaben $j = 1,... ,n$ mit einer Bearbeitungsdauer p_i vorgegeben, welche an einer einzigen Maschine bearbeitet werden. Zudem sind Vorrangsbeziehungen vorgegeben. Diese Problematik kann durch ein einfaches RCPSP mit erneuerbaren disjunktiven Ressourcen $r = 1$ modelliert werden. Die Kapazität der Ressourcen betragen $R_1 = 1$, sodass eine Maschine immer nur jeweils eine Aufgabe bearbeiten kann. Die Beanspruchung pro Aufgabe j betragen analog dazu somit $r_{j1} = 1$ für $j = 1,\ldots, n$.

Das „Parallel machine scheduling“ geht dabei einen Schritt weiter und behandelt anstatt einer Ressource mehrere Maschinen $M_1,..., M_m$ auf welchen die Jobs bearbeitet werden können. Wenn die Maschinen dieselbe Leistung erbringen und die Bearbeitungszeit identisch ist, so kann an dieser Stelle von einem RCPSP mir einer kumulativen Ressource gesprochen werden. Im Gegensatz dazu wird der Berechnungsaufwand größer, sobald die Maschinen unterschiedliche Leistungen erbringen, infolge dessen die Bearbeitungszeiten der Jobs davon abhängen, auf welcher Maschine sie bearbeitet werden können. Dieser Fall kann auf das Multi – Mode RCPSP mit m erneuerbaren Ressourcen übertragen werden. Wenn Job j in einem Modus k bearbeitet wird, dann ist p_{jk} die Bearbeitungsdauer und j verbraucht in diesem Fall eine Einheit der Ressource bzw. Maschine M_k.

Eine weitere Tatsache, die in Verbindung mit der Maschinenbelegungsplanung von Bedeutung ist, ist das Scheduling mit flexiblen Maschinen. In dieser Konstellation wird jede Aktivität mit einem Set von Maschinen $\mu_j \subseteq \{M_{1,\ldots,}M_m\}$ in Verbindung gebracht, auf denen diese ausgeführt werden kann. Wenn Aufgabe j auf der Maschine M_k bearbeitet wird, dann

wird die Bearbeitungszeit mit p_{jk} angegeben. Dieses Problem kann, wie oben beschrieben, mit einem multi – mode RCPSP mit m erneuerbaren Ressourcen beschrieben werden.

Beispiele zum Machine – Scheduling, wie z.B. „Job-shop"-Probleme oder das „multi-processor task scheduling" finden sich unter Brucker und Knust [BruKnu].

In diesem Kapitel wurde untersucht, welche Anforderungen aus Sicht der Praxis mit dem RCPSP verknüpft sind. Es wird deutlich, dass Anwendungen aus realweltlicher Sicht oftmals sehr komplex und mit dem Basismodell des RCPSP nicht zu lösen sind. Für diesen Fall hat Kuhlmann 10 Anforderungen formuliert, die an ein praxisrelevantes Modell des RCPSP im Hinblick auf die Umsetzung in einem Project-Management-Tool gestellt werden.

4.2 Anforderungen an ein praxisrelevantes Modell

Bisher war das Hauptaugenmerk der Bestrebungen rund um das Project Scheduling vor allem auf theoretischer Ebene angesiedelt. Auf der anderen Seite handelt es sich bei diesem Problem um einen Themenkomplex, der nicht zuletzt durch den Einsatz von Projekt-managementsoftware durchaus von praxisrelevanter Bedeutung ist. Das Ziel einer praxisrelevanten Lösung ist vorrangig, eine terminlich sehr gute Durchführung der Aktivitäten unter Einplanung knapper Ressourcen zu erhalten. Ein Projektplaner hat im Laufe der Projektdurchführung und Planung mehrmals diese Probleme vorliegen, z.B. im Zuge der Neuplanung und Anpassung. Die ermittelten Berechnungen und Pläne schlagen sich in der Folgezeit direkt in messbaren Erfolg oder Misserfolg nieder. So ist der Bedarf an praxisrelevanten Modellen und analog dazu erfolgreichen Algorithmen groß. Kuhlmann beschreibt die Diskrepanz zwischen theoretischen Modellen und deren Eignung für einen Praxiseinsatz.

Er stellt heraus, dass es zunächst nicht zweckmäßig ist, in erster Linie nach den Praxisanforderungen eines PM-Tools zu fragen. Es ist im ersten Schritt eher sinnvoll, allgemeine Praxisanforderungen an das Project Scheduling zu formulieren und nach diesen Überlegungen eine Anpassung der Software zu gestalten. Kuhlmann beruft sich in seinen Ausführungen auf die gesammelten Erfahrungen von Beratern und Managern aus einem bestimmten Unternehmen. Aus Interviews ergab sich ein Katalog von Anforderungen, die er in αβγ- Anforderungen unterteilt. Kuhlmann [Kuhl] kehrt die ursprüngliche Reihenfolge der Notation um und beginnt mit den γ- Anforderungen. Nach seiner Ansicht erscheint es sinnvoll „…zunächst Anforderungen an die Zielsetzung eines Modells bzw. die erhofften Ergebnisse zu spezifizieren, bevor detailliert auf einzelne Aktivitäts- bzw. Ressourcencharakteristika eingegangen wird." Dem soll im Folgenden entsprochen werden.

4.2.1 γ – Anforderungen

Unter Anforderung 1 wird zunächst das Ziel des PS diskutiert. Nach Kuhlmann ist dies vor allem in der Minimierung der Projektlaufzeit zu sehen. Darin widerspricht er Tukel und Rom, welche nach seinen Angaben angeben, das oberste Ziel des PS sei die Maximierung der Projektqualität. Die Gründe für diesen Widerspruch sind nach Meinung des Autors klar nachzuvollziehen. So hat die Art und Weise, wie Aktivitäten im Projektverlauf eingeordnet werden, keinen nachvollziehbaren Effekt auf die Projektqualität. Dabei wird vorausgesetzt, dass Einschränkungen und Reihenfolgebeziehungen eingehalten werden. So ist das Ergebnis eines Scheduling immer ein gültiger und durchführbarer Projektplan und die Qualität wird ausschließlich von der Projektstruktur und dem Inhalt der Aktivitäten bestimmt. Wichtige Bestandteile sind dabei die Produkt-, Objekt-, Projekt-, und Ablaufstrukturierung sowie die Aufwandsplanung und Ressourcenauswahl. Diese zählen somit als Input und nicht Inhalt des PS – es wird lediglich die Frage nach der zeitlichen Anordnung der Aufgaben geklärt.

Weitere relevante Kriterien sind mit den Kosten und der Zeit definiert. Nach Kuhlmann tritt die Kostenminimierung jedoch in den Hintergrund. Demnach werden die Projektkosten vor allem in der Phase der Projektstrukturierung festgelegt – Ressourcen werden zugeordnet und der Aufwand abgeschätzt.

Weitere Gründe, die für die Minimierung der Projektlaufzeit als oberstes Ziel des PS sprechen:

- So verringert sich die Wahrscheinlichkeit der Überlappung mehrer Projekte und Aktivitäten und somit das Risiko von Ressourcenkonflikten. Auch wird die Planungssicherheit erhöht, indem das Gesamtprojekt besser überschaut werden kann.
- Darüber hinaus führt eine schnellere Abwicklung des Projektes zu einem schnelleren Kapitalrückfluss und verringerten Kapitalbindungskosten.

Anforderung 2 wird von Kuhlmann folgendermaßen beschrieben: „Der aus dem PS resultierende Plan hat einen Ressourcenfokus und unterstützt die Projektdurchführung“

Diese Anforderung rückt die Ressourcensicht in den Vordergrund. Es soll gewährleistet werden, dass sämtliche Ressourceninformationen, aus dem Projektplan ausgelesen werden können. Eine transparente Sicht auf die Aufgaben und den beteiligten Ressourcen ist demnach primäre Eigenschaft eines Projektplans, welcher nach dem Projekt-Scheduling ermittelt wird.

Es sollte zu jedem Zeitpunkt während der Projektdurchführung ersichtlich sein, wer was in diesem Moment zu tun hat.

4.2.2 β- Anforderungen

Dieser Typ von Anforderungen wendet sich an die Aktivitäten, die den größten Teil der Praxisanforderungen ausmachen.

Unter Anforderung 3 unterstreicht Kuhlmann den Aufwand, welcher in Zusammenhang mit einer Aktivität ermittelt werden muss. So ist nicht die Dauer einer Aufgabe Grundlage der Planung, als vielmehr deren Arbeitsaufwand.

Demnach ergibt sich der Arbeitsaufwand durch die Messung und Abschätzungen nach Planungsinstrumenten wie der „COCOMO"- oder „Function-Point" -Methode, welche den Aufwand in Form von Personenstunden oder -tagen abschätzen. Anders ist dies in den vorgestellten Modellen geregelt. Darin sind Aktivitäten primär durch ihre Dauer gekennzeichnet.

Anforderung 4 wird wie folgt beschrieben: „Eine Aktivität wird von genau einer Ressource durchgeführt".

Das besagt in erster Linie, dass es sinnvoll ist, Aktivitäten so detailliert wie möglich zu planen. Diese Detaillierung führt dazu, dass eine so definierte Aufgabe nicht mehr als einige Personentage benötigt und lediglich eine Ressource beanspruchen muss. Wird eine Aufgabe weniger genau geplant, so entsteht die Gefahr der Fehleranfälligkeit und Ungenauigkeit. Zudem kann es in Folge dessen zu Kompetenz- und Zuständigkeitskontroversen kommen, wonach im Falle eines Fehlers kein Mitarbeiter Verantwortung übernehmen will. Im Gegenzug wird mit steigendem Detaillierungsgrad die Identifikation des Mitarbeiters zu seinen Aufgaben gestärkt und die Motivation wird gefördert.

Unter der Anforderung 5 bis 7 untersucht Kuhlmann die Aufteilung einer Aktivität über den Zeitverlauf. Anforderung 5 beschreibt den Standardfall, wonach es keine Einschränkungen, abgesehen von Vorrangs- und Ressourcenrestriktionen, bei der Zuordnung von Arbeitseinheiten zu Zeitperioden geben darf. Einerseits sind Ressourcen nicht zu jedem Zeitpunkt während des Projektes mit derselben Einsatzkraft vorhanden. Schulungen, Regeldienste und Tarifbestimmungen beispielsweise reglementieren diesen Aspekt. Andererseits müssen auch Unterbrechungen, wie z.B. Urlaubstage, Ausfälle, einkalkuliert werden.

Es gibt viele Möglichkeiten, wonach innerhalb des Projektes Unterbrechungen oder Verschiebungen auftreten können.

Die Anforderung 6 schränkt die vorangegangen Aussagen ein und besagt, dass es im Gegenzug auch möglich sein muss, eine Ressource mit gleich bleibender Kapazität über den Zeitverlauf einzuplanen. Des Weiteren wird unter Anforderung 7 eine weitere Relativierung

zu Anforderung 5 gegeben, indem gesagt wird, dass in der Praxis Aufgaben existieren, die nicht unterbrochen werden können oder dürfen. Bei der Durchführung physikalischer und chemischer Experimente beispielsweise gibt es vielerlei Prozesse, die nicht unterbrochen werden können. Auch ist es aus Sicht der Praxis häufig der Fall, dass die beiden zuletzt genannten Anforderungen auch in Kombination auftreten können.

In der von Kuhlmann verfassten Anforderung 8 werden erstmals die Vorrangbeziehungen unter den Aufgaben beleuchtet. Es sind vielerlei Modellierungsmöglichkeiten zu berücksichtigen:

- Start – Start
- Start – Fertigstellung
- Fertigstellung – Start
- Fertigstellung – Fertigstellung

Diese Beziehungstypen können immer mit minimalen und maximalen Zeitverzögerungen versehen werden. Demnach können immer acht verschiedene Vorrangsbeziehungen modelliert werden. Diese Anforderung ist demzufolge wichtig, um die in der Praxis auftretenden Wechselwirkungen unter den Aufgaben zu modellieren. Der Standardfall ist sicherlich die Fertigstellung – Start – Beziehungen, wobei alle Typen in der Praxis zur Anwendung kommen. Um ein Beispiel zu nennen: es ist eine Start – Start – Beziehung denkbar, wenn etwa mehrere Aufgaben unter den gleichen Bedingungen durchgeführt werden müssen - es müssen spezielle Räume angemietet werden, Spezialwerkzeuge kommen zum Einsatz oder eine Maschine wird umgerüstet.

Die angesprochenen Zeitverzögerungen werden beispielsweise benötigt, wenn es um zeitkritische Anwendungen wie die Bearbeitung von Lebensmitteln geht. So verderben Obst und Gemüse nach relativ kurzer Zeit, wohingegen Wein erst nach einer bestimmten Zeitspanne ausgereift ist.

Die Anforderung 9 - „Aktivitäten können Stichtage besitzen“ - schränkt die absolute zeitliche Position einer Aktivität ein. Start- und Fertigstellung müssen in einigen Projektsituationen terminlich fest vorgeschrieben werden.

So sind folgende Modellierungen umzusetzen:

- „muss anfangen am“ (Wartungsarbeiten)
- „muss enden am“ (Releasetermine)

- „muss anfangen vor“ (Schulungsmaßnahmen für eine neue Software)
- „muss enden vor“ (Euro-Fähigkeit von Systemen vor dem 1.1.2000),

um nur einige Möglichkeiten zu nennen. Es sollte in der Praxis jedoch darauf geachtet werden, dass diese Termine sehr sparsam eingesetzt werden, so kann die Flexibilität des Projektplanes gewährleistet werden.

4.2.3 α – Anforderungen

Nach einer umfangreichen Betrachtung der Aktivitäten kommen in der letzten Anforderung 10 die Praxisanforderungen an die Ressourcen zur Sprache:

„Die Verfügbarkeit einer Ressource wird für jede Zeitperiode einzeln angegeben in Arbeitseinheiten je Zeitperiode“ [Kuhl].

Dieser Punkt knüpft an Anforderung 3 an. So wird die Vergleichbarkeit zwischen Nachfrage nach Arbeitskapazität und Aktivitäten und dem Angebot durch Ressourcen sichergestellt.

Das Zusammenwirken der vorgestellten Praxisanforderungen verdeutlicht, welche Probleme und Sachverhalte bei der Modellierung eines Projektes und der dazugehörigen Planung der Aktivitäten aus Sicht des Projektmanagers berücksichtigt werden müssen. Kuhlmann kommt in seinen Ausführungen zu dem Schluss, dass kein einziges der PSP-Modelle auch nur annähernd alle Anforderungen erfüllt.

Mit bestehenden Modellen werden bestenfalls vier der zehn Praxisanforderungen erfüllt.

Somit muss eine Lücke zwischen Praxisanforderungen und bestehenden Modellen geschlossen werden.

Es soll an dieser Stelle auf die Publikation von Kuhlmann verwiesen werden, der darin ein erweitertes Modell erstellt, welches den von ihm formulierten Anforderungen genügt. Darüber hinaus zeigt er, dass es möglich ist, den genetischen Algorithmus auf dieses Modell zu übertragen. Der Autor verzichtet an diesem Punkt auf die Beschreibung des Modells und Algorithmus`, da dies den Umfang der vorliegenden Diplomarbeit sprengen würde.

Im Verlauf der Arbeiten mit dem beschriebenen Projektmodell aus SimProQ und der Implementierung des genetischen Algorithmus` war es dem Autor möglich, das PS|*prec*| * C_{max} vollständig in die Praxis umzusetzen. Im Rahmen der Diplomarbeit ist es somit die Intention in Form von praktischen Tests nachzuweisen, dass bereits bei diesem vereinfachten Modell des PS|*prec*| * C_{max} erhebliche Qualitätssteigerungen bei etablierten Projektmanagementsystemen möglich sind.

4.3 Algorithmus in MSP

Im Folgenden wird Bezug genommen auf die Aussagen auf der Internetseite [MS] von Microsoft und dem Handbuch zu MSP 2000 [MSP]. Dort wird der verwendete Algorithmus in groben Hinweisen beschrieben, die grundsätzliche Vorgehensweise kann jedoch nachvollzogen werden.

So arbeitet MSP mit einfachen Verzögerungen, sodass Vorgänge, die im Konflikt zueinander stehen, einfach nach hinten geschoben werden, bis diese nicht weiter parallel arbeiten müssen. Diese Beschreibung lässt sich auf das SGS übertragen, welches bereits in Kapitel 4 zur Sprache gekommen ist. Nun ist es interessant, nach welchen Auswahlkriterien entschieden wird, welche Aktivitäten verschoben werden oder nicht.

Hartung, Mellentien und Trautmann [HaMelTrau] kommen bei dieser Frage bei der Analyse von MS Project 2000 zu den folgenden Ergebnissen: Demzufolge werden Vorrangbeziehungen, Puffer, Termine, Prioritäten und Einschränkungen der einzelnen Aktivitäten ausgewertet und daraus entschieden, ob und wie Vorgänge abgeglichen werden sollen. Vorgänge mit einer größeren Pufferzeit und/oder späteren Anfangsterminen werden somit zuerst verzögert.

Laut den offiziellen Angaben von Microsoft [MS] wird nach den folgenden Kriterien erwogen, welche Aufgabe als nächstes berechnet wird oder nicht:

- verfügbarer Puffer einer Aufgabe
- Dauer
- Definierte Aufgaben-Beschränkungen
- ID
- Prioritätskennzahl
- Abhängigkeitsbeziehungen
- Berechnungsdaten

Aber auch Informationen über die benötigten Ressourcen fließen in die Entscheidung mit ein:

- die Ressourcenverfügbarkeit, ersichtlich im Ressourcenkalender
- die maximal verfügbaren Einheiten einer Ressource zu einem bestimmten Zeitpunkt
- die benötigten Einheiten einer Ressource zur Erledigung einer Aufgabe

Nicht alle dieser Daten sind in diesem Kontext interessant, da sie teilweise auf das Modell von MS Project zugeschnitten sind. Es wird jedoch anhand der Erläuterungen deutlich, dass der Kapazitätsausgleich in MS Project auf einer gut konfigurierten Prioritätsregel beruht und

eine wirkliche Optimierung - und damit eine Metaheuristik - nicht zum Einsatz kommt. Die erste Berechnung wird somit als Lösung präsentiert.

Demnach ist bereits an dieser Stelle ersichtlich, dass durch die Auswahl geeigneter Algorithmen bereits bei kleineren Projekten bessere Lösungen erzielt werden können.

Jedoch zeigte sich im Verlauf der Tests, die vergleichend zu der Implementierung der hier vorgestellten Algorithmen durchgeführt wurden, dass dieses Vorgehen durchaus gute bis sehr gute Ergebnisse erzielen kann. Auch wurde schnell klar, dass vor allem die Eröffnungsberechnungen, nach den klassischen Prioritätsregeln, einerseits nicht an die Qualität von Microsoft heranreichten und andererseits auch als Grundlage für den genetischen Algorithmus an mancher Stelle nicht gut genug waren. So musste ein erweitertes Vorgehen implementiert werden, wonach sich der Autor an den Vorgaben von Microsoft orientierte (siehe Kapitel 5.1).

4.4 Testergebnisse aus der Literatur

4.4.1 Tests der Algorithmen

Die beschriebenen Algorithmen wurden in der Vergangenheit unter Berücksichtigung verschiedener Kriterien mehrfach getestet. An dieser Stelle sollen, bevor im letzten Abschnitt dieser Arbeit eigene Testresultate präsentiert werden, diese Ergebnisse herangezogen werden, um erste Rückschlüsse auf die Qualität der Algorithmen zu ziehen.

Zuerst soll auf die Ausführungen von Danis [Dan] eingegangen werden. Danis untersucht den Genetischen Algorithmus auf dessen Eignung in Bezug auf die Optimierung der Kosten in einer Projektberechnung. Er nutzt für seine Tests einen Pentium 4Prozessor mit 3,06 GHz, 1024 MB RAM auf der Plattform Windows XP Professional. Die von ihm implementierten Algorithmen sind in Java unter dem JDK 1.4.2. geschrieben. Die Voraussetzungen sind also ähnlich der hier Vorliegenden. Zu der genauen Testbeschreibung wird auf das Unterkapitel 4.5 verwiesen.

Es wird unter anderem ein Testfall mit 100 generierten Berechnungen für 100 Tasks beschrieben. Es fällt auf, dass die durchschnittliche Berechnungsdauer des GA mit 70,5 s sehr hoch ausfällt. Dieses Performanceproblem ist für die reale Anwendung sicherlich nicht akzeptabel und soll durch eigene Tests im Rahmen dieser Arbeit verbessert werden.

Dennoch muss die erreichte Qualität und die gerechtfertigte positive Bewertung der von Danis beschriebenen Tests hervorgehoben werden. Er kommt zu dem Schluss, dass es mithilfe des GA für ein Unternehmen möglich und realistisch ist, seinen Cash-Flow innerhalb

von 5 Minuten pro Projekt zu verbessern. Eine manuelle Verbesserung innerhalb einer Projektmodellierungssoftware würde einen Experten eine lange Zeit kosten.

Auch Hartmann [HART] hat bereits umfangreiche Tests mit den im vorhergehenden Kapitel beschriebenen Algorithmen durchgeführt. Was bei diesen Ausführungen auffällt, ist die hohe Qualität der GA-Lösungen. Bei jeder von ihm untersuchten Projektgröße, war es der GA-Algorithmus, welcher die besten Ergebnisse lieferte. Auch ist es nach Ansicht des Autors erstaunlich, dass einfache Prioritätsregeln mit verhältnismäßig geringem Berechnungsaufwand eine verhältnismäßig gute Lösung anbieten und als Eingangsalgorithmen für den GA als geeignet erscheinen.

Vor allem diese dargestellten Testergebnisse waren für den Autor bezüglich der Wahl des Algorithmus für den hier beschriebenen Problemfall ausschlaggebend.

4.4.2 PM-Softwaretests

In diesem Kapitel sollen Testergebnisse kommerzieller PM-Software zusammengefasst werden. Kolisch (1999) stellt zu diesem Thema die PM-Tools Artemis Schedule Publisher, CA SuperProject, MS Project, Primavera Project Planner, Project Manager Workbench, Project Scheduler und TimeLine vor (Literaturverweis aus Kuhlmann [Kuhl]) und beschreibt diese wie folgt:

So verwenden alle untersuchten Softwarepakete ausschließlich proprietäre Heuristiken zur zeitlichen Allokation von Aktivitäten und Ressourcen. Diese beruhen auf einfachen Prioritätsregeln, welche unter 3.5 bereits zur Sprache gekommen sind. Des Weiteren verfügt keine Anwendung über eine exakte Methode, Local Search Verfahren oder Metaheuristiken. Im Laufe der Diplomarbeit wurde bereits geklärt, warum eine exakte Methode wie das Branch and Bound – Verfahren aus Komplexitätsgründen für eine praktische Anwendung nicht in Frage kommt. Des Weiteren bieten selbst Add-Ons zu diesen Softwarepaketen keine solchen Verfahren an. Kolisch kommt zu dem Schluss, dass hohe unausgeschöpfte Verbesserungspotenziale bestehen und beruft sich dabei auf eigene beste Lösungen zu vorliegenden Projektmodellen.

Eine wichtige Publikation, die sich ausschließlich mit dem PS|*prec*| C_{max} beschäftigt und kommerzielle Softwaresysteme auf die Lösungsqualität dieses Problems testet, ist die von Hartung, Mellentien und Trautmann [HaMelTrau]. Darin werden die Softwarelösungen Project Scheduler 8.0.1, CA Super Project 5.0a, CS Project Professional 3.0, Acos Plus.1 8.2 und MS Project 2000 getestet. Die Untersuchungen beziehen sich auf 1560 Testinstanzen mit 30 (480 Instanzen), 60 (480 Instanzen) und 120 (600 Instanzen) Aktivitäten und jeweils 4 Ressourcen ist somit sehr umfangreich und daher auch aussagekräftig. Keinem der erwähnten

Programme ist es gelungen, auch nur eine Referenzlösung aus dem Test zu unterbieten. Die folgende Tabelle gibt die mittlere und maximale prozentuale Abweichung der von den Programmen berechneten Projektdauern für das gesamte Testset (Instanzen mit 30,60 und 120 Aktivitäten) wieder.

	Mittlere Abweichung [%]				Maximale Abweichung [%]			
	30	60	120	alle	30	60	120	Alle
MS Project	5.18	6.23	14.02	8.90	31.03	29.89	46.79	46.79
Scitor PS 8	4.85	4.98	11.15	7.31	37.93	36.89	31.11	37.93
CS Project	3.50	5.28	13.70	7.97	25.42	23.16	30.00	30.00
SuperProject	5.39	6.37	13.99	9.00	36.23	32.14	41.98	41.98
Acos Plus.1	3.87	4.05	9.69	6.16	24.62	26.39	28.95	28.95

Tabelle 4: Ergebnisse des Testsets (Quelle: [HaMelTau])

Dabei fällt auf, dass die maximale Abweichung im schlechtesten Fall bei nahezu 50% liegt. Der so entstandene Projektplan würde somit fast die Hälfte länger dauern, als die komplementäre Referenzlösung. Die zusätzlichen Kosten und verspätete Zahlungseingänge wären dementsprechend hoch, von ggf. auftretenden Terminkonflikten noch abgesehen. Von der nächsten Tabelle wird dieses Ergebnis noch zusätzlich untermauert.

	Varianz [%]				Referenzlsg. Erreicht			
	30	60	120	alle	30	60	120	Alle
MS Project	44.66	60.70	83.20	80.94	95	76	23	194
Scitor PS 8	48.23	53.44	54.35	61.40	129	145	98	372
CS Project	22.99	46.06	52.79	62.54	135	125	57	317
SuperProject	53.60	67.13	75.26	81.82	109	88	30	227
Acos Plus.1	30.16	39.83	46.34	47.15	143	174	115	432

Tabelle 5: Varianz und erreichte Referenzlösungen (Quelle: [HaMelTau])

Wieder sind es die Werte von Acos Plus.1, welche die anderen Ergebnisse dominieren. Auffällig ist die vergleichsweise kleine Varianz bei CS Project bei den Instanzen mit 30 Aktivitäten. Schließlich sollen die Anzahl der Instanzen betrachtet werden, bei denen die SW-Pakete einen Projektplan mit gleicher Dauer errechneten. Wieder sind es Acos Plus.1 und Scitor PS 8, welch durch gute Werte hervorstechen.

Abschließend stellen Hartung, Mellentien und Trautmann [HaMelTrau] fest, dass keines der betrachteten Softwarepakete einen Algorithmus bereitstellt, der den besten state-of-the-art – Heuristiken aus der Literatur nahe kommen würde. Nach den Ergebnissen waren die heuristischen Methoden von Acos Plus.1 und Scitor Oroject Scheduler den Konkurrenzlösungen überlegen. Darüber hinaus ist festzuhalten, dass die Dauer der Projektpläne deutlich steigt, sobald realistische Szenarios, d.h. Pläne mit einer großen Anzahl von Aktivitäten und Ressourcen, betrachtet werden. Somit verfügt kein Produkt über eine exakte Methode; jede Software setzt auf eine schnelle Heuristik. Nach Einschätzung von Hartung, Mellentien und Trautmann [HaMelTrau] ist eine zusätzliche Implementierung weiterer Algorithmen gerechtfertigt, auch unter der Prämisse, längere Rechenzeiten und der damit nicht zu erfüllenden Benutzerforderung nach einer interaktiven Software in Kauf zu nehmen.

5 Implementierung und Tests

5.1 Das Simulationstool

Die praktische Arbeit als Basis zur vorliegenden Diplomarbeit lässt sich am Besten an einer Beschreibung des entwickelten Simulationstool verdeutlichen. Die folgende Abbildung 8 zeigt die entwickelte Software.

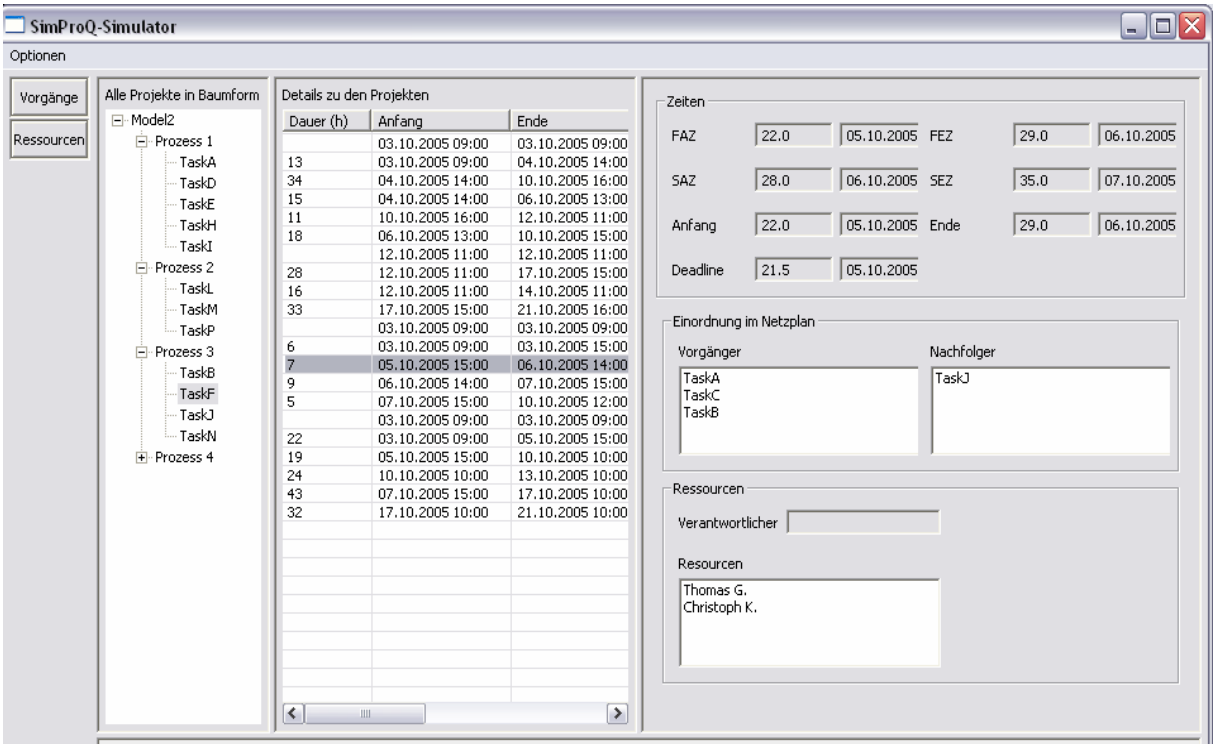


Abbildung 8: Das Simulationstool

In der Abbildung ist zunächst nur die obere Hälfte des Tools dargestellt – die Untere, mit der Visualisierung der Projektinstanzen in einem Gantt-Diagramm, ist im Anhang hinterlegt. Der Linke Teil der obigen Abbildung listet alle Projektinstanzen in Baumstruktur auf und dient der Navigation innerhalb der Projekte, der Auswahl von Vorgängen oder Prozessen. Wenn eine Aufgabe – in der Darstellung als Task bezeichnet – ausgewählt wird, springt der Cursor auf die dazugehörige Zeile in der Tabelle, welche die Attribute der betreffenden Aufgabe beinhaltet. Eine Ausführliche Darstellung der Attribute wird auf der rechten Seite dargestellt. Hier findet der Nutzer Informationen zur zeitlichen Einordnung (FAZ, FEZ, SAZ, SEZ, Anfang und Ende), der beteiligten Ressourcen sowie Vorgänger und Nachfolger.

Auf der linken Seite ist zudem eine Navigationsleiste zu sehen, welche, ähnlich wie in MS-Project, zwischen der vorgangs- und der ressourcenbasierten Sicht wechselt. Die zuletzt genannte Ansicht beinhaltet die Auflistung aller bisher definierten Ressourcen in Form einer Tabelle. Es sei an dieser Stelle auf den Anhang verwiesen, in welchem ein Beispiel für die grafische Darstellung der Ressourcenauslastung gegeben ist.

An dieser Stelle wird klar, dass eine Implementierung der Algorithmen ohne diese grafischen Elemente kaum zu realisieren gewesen wäre. Allein zur Überprüfung der Richtigkeit der Berechnungen und Veranschaulichung der Qualität der Ergebnisse war hiermit eine gute Unterstützung gegeben.

Der Kapazitätsausgleich in dem entwickelten Simulationstool erfolgt nach der seriellen Methode. Des Weiteren konnten der „Genetische Algorithmus“ basierend auf der Activity List (siehe 3.8.2) und das „Sampling“ bereits implementiert werden. Zur Implementierung und Erprobung weiterer Metaheuristiken hat im Rahmen dieser Diplomarbeit schlichtweg die Zeit gefehlt. Wie bereits erwähnt, ging aus der Literatur, z.B. Hartmann [Hart], hervor, dass basierend auf diesen Heuristiken, durchaus sehr gute Ergebnisse erzielt werden konnten. Das war der entscheidende Grund, welcher zur Auswahl der genannten Algorithmen für eine Implementierung geführt hatte. Die Algorithmen können darüber hinaus, zur Erzielung spezieller Testergebnisse, vom Nutzer kalibriert werden – z.B. das Verändern der Populationsgröße und der Anzahl der Generationen für den genetischen Algorithmus. Als Eingangslösungen für den GA wurden die Prioritätsregeln MinimalSlack, MostTotalSuccessor, LatestFinishTime und ein an die Angaben von Microsoft orientierter Algorithmus implementiert. Es wurde eine Heuristik entwickelt, die, ähnlich wie bereits erläutert, auf die Attribute einer Aufgabe zurückgreift und in Abhängigkeit davon entscheidet, welcher Vorgang im nächsten Berechnungsschritt verschoben wird oder nicht. Darin werden jedem Vorgang, der für den nächsten Berechnungsschritt in Frage kommt, nach Berücksichtigung seiner Lage im Netzplan, Punkte zugeordnet. Punkte gibt es beispielsweise, wenn ein Vorgang besonders viele Nachfolger aufzuweisen hat, bzw. die Nachfolge-Aufgaben eine hohe Restbearbeitungszeit aufweisen.

Das Simulationstool erfasst alle auf Datenbank-Ebene existierenden Projektinstanzen. Wie bereits erwähnt ist es aus diesem Grunde notwendig, die Überlastbereiche einer Ressource auch über alle Projekte hinweg zu registrieren. Die Implementierung eines Ressourcenkalenders, der alle ressourcenspezifischen Vorgänge abspeichert, bewerkstelligt dies durch geeignete Erkennungsalgorithmen. Für den genetischen Algorithmus kann infolgedessen anhand der Ressourcenkalender eine Activity-List zusammengestellt werden.

5.2 Beschreiben der eigenen Testumgebung

Diese Phase soll die erstellten Algorithmen auf ihre Eignung für den definierten Anwendungsbereich prüfen. Nach Dumke [Dum S.93] wird die Phase der Erprobung wie folgt definiert:

„Die Erprobung eines Software-Systems (software acceptance testing)“ ist der Nachweiß seiner Validität auf der Grundlage von Akzeptanzkriterien in einem ausgewählten Anwendungsfeld. “

Dumke beschreibt weiter, welche notwendigen Schritte durchgeführt werden müssen, um eine korrekte Erprobung durchzuführen und repräsentative, korrekte Ergebnisse zu erzielen. So müssen zunächst Akzeptanzkriterien definiert werden, welche die Art, den Umfang und den Bereich der Testumgebung festlegen, wonach es möglich ist, eine Bewertung abzugeben.

Eine Verlaufsbeschreibung gibt genau vor, in welchen Schritten die Tests erfolgen. Während des Testverlaufs ist somit konkret vorgeschrieben, was im jeweiligen Schritt gemacht wird und welche Ergebnisse dieser hervorbringt. Darüber hinaus muss beschrieben werden, auf welchen Soft- und Hardwareressourcen die Erprobung stattfinden soll. Die Schnelligkeit des Prozessors stellt dabei einen nicht zu unterschätzenden Input bei der Bewertung der Performance der Heuristiken dar. Auch ist akquirierte Standard- und Anwendungssoftware zu benennen, die während der Erprobung zum Einsatz gekommen ist.

Um auf einen repräsentativen Datenbestand für das hier zu betrachtende System zurückgreifen zu können, muss geprüft werden, inwieweit es möglich ist, Projektbäume generieren zu können. Allein manuell erstellte Projektmodelle wären im Umfang nicht ausreichend und würden zu viel Zeit in Anspruch nehmen.

5.2.1 Richtigkeit des Netzplans

Um einen Test überhaupt erst durchführen zu können, muss sichergestellt werden, ob die Richtigkeit der Berechnungen gewährleistet wird.

Zu den nun folgenden Punkten muss gesagt werden, dass der Autor und gleichzeitig auch Programmierer des Simulationstools die erläuterten Berechnungsvorschriften mit einem hohen Aufwand geprüft hat und von einer Korrektheit des Netzplanes überzeugt ist. Tatsächlich kann erst nach langzeitigen Tests am Endanwender mit letztendlicher Sicherheit gesagt werden, dass das Projektmodell im Gesamten, im Zusammenwirken mit den implementierten Algorithmen, wirklich reibungslos funktioniert.

Danis [Dan] nennt dazu folgende Kriterien, welche mit dem Begriff Korrektheit zur Sprache kommen sollen: Zum Einen muss gewährleistet sein, dass der Netzplan - zunächst ohne Berücksichtigung des Ressourcenausgleichs – korrekt initialisiert wird. So müssen die Reihenfolgebeziehungen, also Vorgänger/Nachfolge-Relationen richtig abgebildet werden und daraus resultierend, die Zeiten (FAZ, FEZ, SAZ, SEZ usw.) für jede Task berechnet werden.

Sind zudem alle Deadlines und Zeiträume (Dauer) richtig gesetzt und die Zuordnung der Ressourcen richtig erfolgt? Darüber hinaus ist es vor allem im Hinblick auf die Performance von entscheidender Bedeutung, dass die Berechnungen innerhalb der Simulation in den richtigen Schritten und definierten Abläufen stattfinden – was aber auf die eigentliche

Korrektheit des Netzplanes keinen Einfluss hat, lediglich die Qualität des Ergebnisses wird dadurch beeinflusst.

Im nächsten Schritt muss geprüft werden, ob diese vorangegangenen Aspekte auch nach einem Ressourcenabgleich noch erfüllt sind und die Aufgabenpakete während der Berechnung auch korrekt verschoben wurden.

5.2.2 Die Generierung von Testdaten

Danis [Dan] gibt in seinen Ausführungen eine weit reichende Beschreibung der Eigenschaften und Bedienung der ProGen/max-Software. Auch in weiteren Literaturquellen wird dieses Programm als Grundlage und Standard zur Testung von Optimierungsalgorithmen im Zusammenhang mit dem RCPSP beschrieben, wie z.B. Hartmann [Hart].

ProGen/Max ist eine Software zur Generierung von Projektmodellen zum RCPSP. Die damit zu erstellenden Projektinstanzen können zur Bewertung der unterschiedlichen Optimierungsalgorithmen herangezogen werden. ProGen/max generiert Graphen, die auf den AON Netzplänen beruhen und nutzt gewichtete Verbindungen, um Zeitspannen zwischen den Aufgaben nachzubilden. Der Generator erstellt Netzpläne zu drei Problemtypen: minimal make-span RCPSP, Ressourcen-leveling und net present value. Die folgende Grafik zeigt den Umfang der Bedienmöglichkeiten durch den Nutzer.

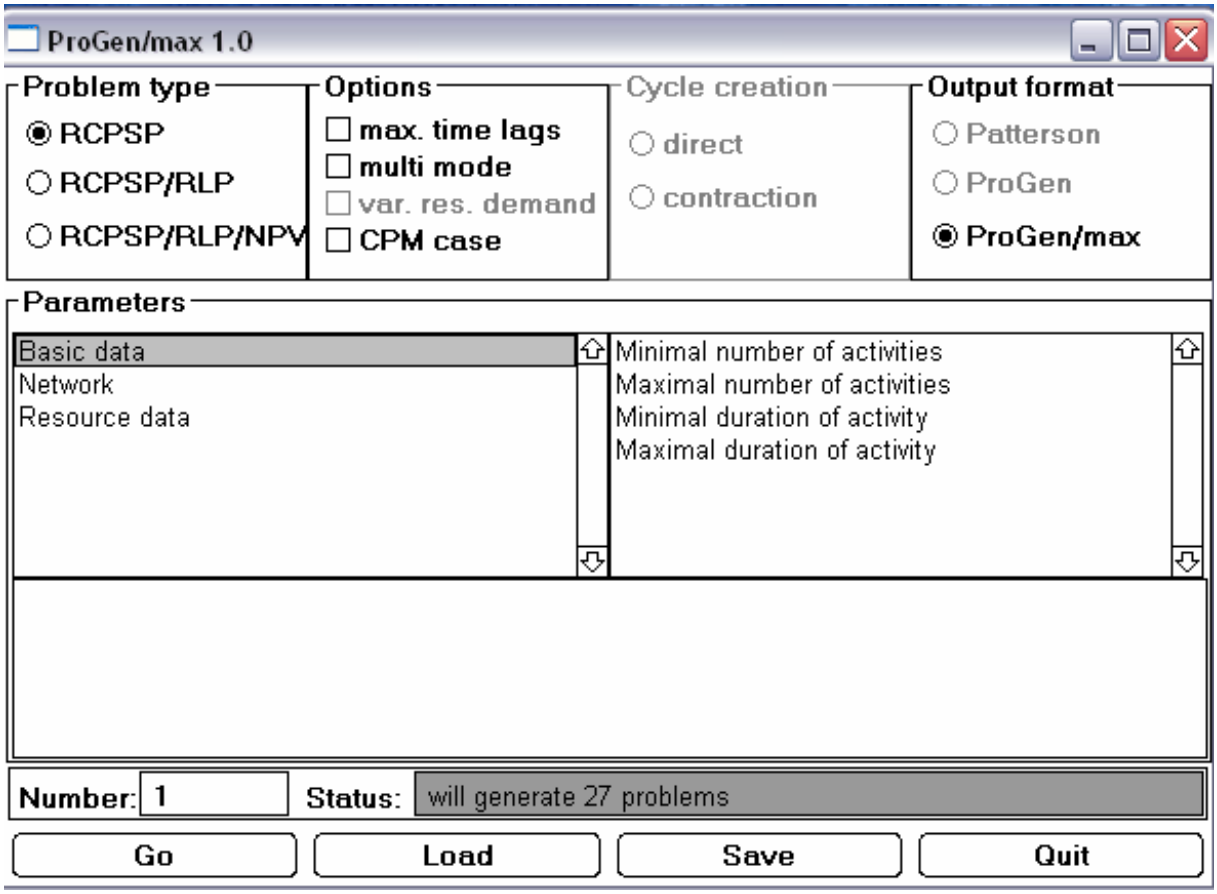


Abbildung 9: Progen/max

Die Nutzer können, wie aus der Grafik hervorgeht, Parameter, wie z.B. die minimale und maximale Dauer und Anzahl der zu generierenden Arbeitspakete, angeben. Auch die minimale und maximale Anzahl der Vorgänger- und Nachfolgeaufgaben, sowie die Anzahl der Tasks, die am Anfang und Ende eines Projektmodells stehen, lassen sich so festlegen.

Ein weiterer wichtiger Punkt ist die Festlegung der Ressourceneigenschaften innerhalb des generierten Projektes. Das Programm nutzt sowohl erneuerbare als auch nicht erneuerbare Ressourcen. Der Nutzer bestimmt die Gesamtanzahl der Ressourcen und wie viele davon maximal an einer Task beteiligt sind.

Nach Betätigung des „GO“ Button, erstellt ProGen/max die Projektinstanzen – in der obigen Grafik wären das demzufolge 27 Modelle, welche einzeln als .sch- Dateien auf der Festplatte gespeichert werden. Zusätzlich werden Textdateien angelegt, welche Statistiken, Daten und allgemeine Parameter der erstellten Projektmodelle wiedergeben.

Die auf diese Weise erstellten Projektinstanzen müssen im nächsten Schritt in das Simulationstool importiert werden, um sie auf die implementierten Algorithmen anzuwenden.

5.2.3 Überführen der Projekte in MS Project

Zum Export der Projekte muss das hier verwendete Modell in ein von MS Project akzeptierten Format exportiert werden. Dazu wird eine bereits existierende Schnittstelle aus SimProQ genutzt, welche das Modell in eine XML-Datei überführt. Darin werden alle Informationen in Baumform angelegt: Attribute der Aufgaben, Reihenfolgebeziehungen und Ressourceninformationen.

Ist das Projekt exportiert, kann im nächsten Schritt der Ressourcenausgleich anhand der in MS Project verwendeten Algorithmen erfolgen. Das Ergebnis dieser Berechnung kann nun wiederum herangezogen werden, um einen qualitativen Vergleich zu den im Rahmen der Diplomarbeit implementierten Algorithmen anzustellen. Dazu muss ein gewisser Umfang von Projektinstanzen getestet werden, um eine repräsentativ gültige Aussage zu den Qualitäten der beiden Systeme treffen zu können. Als Bewertungskriterium wird hier zunächst die Gesamtdauer des jeweiligen Projektes herangezogen.

Um einen gültigen Vergleich anstellen zu können, müssen beide Systeme aufeinander abgestimmt werden. Das beinhaltet vor allem Angaben zu Arbeitszeiten, welche im Standardkalender in MS Project festgelegt werden. So kann ein Arbeitspaket beispielsweise an einem Wochenende nicht weiterbearbeitet werden. Auch müssen Ressourcenkalender, also wöchentliche Arbeitszeiten eines Mitarbeiters, einander entsprechen.

5.3 Testergebnisse

Im Folgenden sollen nach der ausführlichen Zusammenstellung von Ergebnissen aus der Literatur in Kapitel 4.4 eigene Testergebnisse präsentiert werden. Dabei wurden unter ProGen/max Testsets generiert, welche sich jeweils durch spezifische Attribute voneinander unterscheiden. Im Anschluss werden die Projekte berechnet und gegen MS Project getestet.

5.3.1 Testset 1

Im Testset 1 soll zunächst überprüft werden, ob es mit zunehmender Anzahl von Iterationsschritten möglich ist, signifikante Qualitätsunterschiede in den Ergebnissen zu erzielen. Es werden der Genetische- und der Sampling- Algorithmus auf die Probe gestellt, die jeweils nach den Kriterien Projektdauer und Einhaltung der Meilensteine berechnet werden. Die Einstellungen des Testsets unter ProGen/max sind nachfolgend dargestellt:

- Anzahl der Aktivitäten: 50;
- Dauer der Aktivitäten: 5 - 20;
- Anzahl der Ressourcen: 2 - 5;

- Anzahl der Ressourcen, die von einer Aktivität genutzt werden: 1 - 5;

Die Kennzahlen zeigen, dass eine große Anzahl möglicher Ergebnisse zu einem Projekt existieren kann, da die Projekte mit fünfzig Aktivitäten, mit jeweils maximal fünf Ressourcen, durchgeführt werden können. Die generierten Projektinstanzen schwanken zwischen unterschiedlichen Ausprägungen - also Anzahl der genutzten Ressourcen -, wonach es interessant ist, unter welchen Rahmenbedingungen die hier dargestellten Algorithmen gute oder schlechte Ergebnisse erzielen. Es wurden dreißig Projekte getestet. Die durchschnittliche Dauer gibt die Anzahl der Projektstunden an, die für die Durchführung eines Projektes benötigt werden.

Die erzielten Testergebnisse sind in der folgenden Tabelle dargestellt:

Algorithmus	Iterationen	⊘ Dauer	⊘ Rechenzeit in ms
GA	50	379,8	6346
GA	100	373,7	11654,7
Sampling	50	388,9	2275,6
Sampling	100	386,2	4454,7

Tabelle 6: Ergebnisse Testset1

Zum Vergleich betrug die durch MS Project errechnete durchschnittliche Projektdauer 392,8.

So waren alle hier getesteten Algorithmen im Durchschnitt erfolgreicher als die von MS Project errechneten Lösungen. Das beste Ergebnis, mit einer Dauer von 373,7, erzielte der GA, wonach MS Project pro Projekt somit durchschnittlich 19,1 Projektstunden und damit 5% schlechter war. Anhand dieser Daten wird, im Vergleich mit den Ergebnissen von Hartung, Mellentien und Trautmann [HaMelTrau], deutlich, dass die Qualität der Lösungen, für den hier betrachteten Projektumfang von 50 Aktivitäten, die Gegenüberstellung mit vergleichbaren Referenzlösungen nicht scheuen muss. Bei gleichem Umfang (siehe Tabelle 5) war MS Project auch in dieser Untersuchung ungefähr 5-6 % schlechter als die besten Referenzlösungen.

Es soll an dieser Stelle angemerkt werden, dass während der Tests, jeweils die ersten guten Ergebnisse übernommen wurden und somit das Optimum noch nicht erreicht wurde. Auch bei

den hier getesteten Algorithmen stellte sich heraus, dass durch mehrfache Wiederholung des Ressourcenausgleichs durchaus bessere Lösungen erzielt werden können.

Anhand der Tabelle 7 können noch weitere Aussagen getroffen werden. Zunächst wird deutlich, dass durch zunehmende Iterationsschritte eine Qualitätssteigerung erreicht wurde. Ferner konnte das Sampling gute Ergebnisse aufweisen und wird auch im Hinblick auf die benötigte Rechenzeit interessant für die Integration in das Projekttool zur Anwendung auf kleinere Projekte. Mit steigenden Iterationsschritten steigt die Rechenzeit linear mit. Überträgt man nun die Rechenzeiten der GA aus diesem Testset auf 100 Aktivitäten, so kann man erkennen, dass diese Kennzahl besser ausfällt als die Ergebnisse aus der Literatur – siehe Danis [Dan]. Es ist ersichtlich, dass in Verbindung mit einer geeigneten Datenstruktur und leistungsstarken Implementierung durchaus Performancesteigerungen möglich sind.

Überdies fiel während der Erprobung auf, dass die Lösungen von MS Project dann sehr gut gerieten, wenn an einem Projekt nur wenige Mitarbeiter arbeiteten und pro Aufgabe höchstens zwei Ressourcen beteiligt waren. Um diesen Sachverhalt genauer zu untersuchen, soll nun ein weiteres Testset erörtert werden.

5.3.2 Testset 2

Die Daten für das Testset 2 sind wiederum den folgenden Punkten zu entnehmen

- Anzahl der Aktivitäten: 50;
- Dauer der Aktivitäten: 5 - 20;
- Anzahl der Ressourcen: 3 - 5;
- Anzahl der Ressourcen, die von einer Aktivität genutzt werden: 1 oder 2;

Es soll, wie bereits erläutert wurde, geprüft werden, welche Ergebnisse erzielt werden, wenn pro Aufgabe nur höchstens zwei Mitarbeiter eingesetzt werden. Dazu wird lediglich der genetische Algorithmus angewandt, da dieser im Beispiel 1 die besten Ergebnisse erzielte. Aus den Tests mit den generierten Projektinstanzen ergab sich eine durchschnittliche Projektdauer von 297,5 Stunden. Es wurden 100 Iterationen durchgeführt bi einer durchschnittlichen Rechenzeit von 11270,4 ms.

Der Projektdauer in MS Project ist mit 303,9 Projektstunden wiederum schlechter ausgefallen, wonach sich auch hier bestätigt, dass der in MS Project angewandte Algorithmus in Bezug auf dessen Qualität mit einem verhältnismäßig geringen Aufwand verbessern lässt.

Die Leistungsfähigkeit fällt mit der ermittelten Rechenzeit immer noch zu langsam aus. Im Kapitel 6 soll demnach analysiert werden, welche Maßnahmen ergriffen werden können, um dieses Problem einzudämmen.

5.3.3 Probleme während der Durchführung

Beim Testen mit dem Simulationstool zeigten sich unterschiedliche Probleme, welche in der Folgezeit, also der Weiterentwicklung der Algorithmen, bearbeitet und gelöst werden müssen.

Es muss die lange Berechnungszeit genannt werden, die bei einer praktischen Nutzung nach der Integration in die Software unbedingt verbessert werden muss. Vor allem die Performance ist ein herausragender Punkt während der Nutzung durch den User, welche auch sofort auffällt.

So ist zu überprüfen, welche Abbruchkriterien beispielsweise bei dem genetischen Algorithmus definiert werden können, um gegebenenfalls Iterationsschritte zu sparen. Auch müssen weitere Erweiterungen dieser Heuristik geprüft werden, welche im Einzelfall eine bessere Performance bieten und eventuell auch qualitative Verbesserungen hervorbringen.

Auch ist auf globale Sicht eine Verschlechterung der Qualität der einzelnen Netzplanberechnungen mit ansteigender Zahl von Projekten/Aktivitäten zu verzeichnen. Die bisherige Implementierung ist lediglich auf eine Verbesserung der globalen Kennzahlen ausgelegt, sodass eine lokale Optimierung eines Projektes nicht immer stattfinden kann. Der Autor ist der Meinung, dass im Zuge der Integration der Algorithmen in die Modellierungssoftware in zwei Schritten vorgegangen werden muss:

- Ein neues Projekt wird modelliert und lokal berechnet. Es ist zu beachten, dass hier noch keine Abstimmung mit dem gesamten Projektportfolio erfolgt, sodass es auf globaler Ebene zu Ressourcenengpässen kommen kann.

- Das Projekt wird in den Gesamtnetzplan eingefügt und durch noch zu bestimmende, performancestarke Algorithmen ausgeglichen. So wird nach Ansicht des Autors Rechenzeit gespart und vor allem ein besseres Gesamtergebnis erzielt. Es muss bedacht werden, dass es in einem größeren Unternehmen nicht selten vorkommen kann, dass zehn, zwanzig oder dreißig Projekte zur gleichen Zeit ablaufen. Das zeigt auch wiederum, welche hohe Bedeutung eine Modellierungssoftware einnehmen kann und wie wichtig ein abgestimmter und ausgeglichener Ressourcenplan ist.

5.4 Auswertung der Testergebnisse

Es darf an dieser Stelle festgehalten werden, dass das Hauptziel der Arbeit mit den Tests erreicht werden konnte: Es zeigt sich, dass konventionelle Algorithmen, wie die erweiterte

Prioritätsregel in MS-Project, mit den umfangreichen Lösungsangeboten aus der Literatur ausgehebelt werden kann. Der genetische Algorithmus hat das Potenzial, bestehende Lösungen aufzugreifen und zu verbessern. Dem Nutzer kann somit erstmals ein wirkliches Optimierungsverfahren angeboten werden. Bereits durch die Entwicklung einer eigenen Prioritätsregel konnte in unabhängigen Tests in vielen Fällen eine bessere Lösung gegenüber den klassischen Verfahren erzielt werden. Dieses Vorgehen kann eine sensiblere Auswahl zur Berechnung von Vorgängen treffen, da sie die gesamte Lage der zu berechnenden Vorgänge berücksichtigt - anders als klassische Prioritätsregeln, welche nach einem einzigen Kriterium vorgehen. Wenn die Entwicklung der Algorithmen, vor allem die genaue Kalibrierung des GA und der Punkte-Regel weiter vorangetrieben werden, bestehen sehr gute Möglichkeiten, dem Optimum weiter näher zu kommen. Zu diesem Zweck müssen jedoch weitere umfangreichere Tests stattfinden, um die Feinjustierung der Verfahren zu verbessern.
An diesem Punkt soll auch nicht verschwiegen werden, dass der GA mit zunehmender Projektgröße in der derzeitigen Implementierung nicht nur Performanceschwächen zeigt, sondern auch die Qualität aufgrund der Konzeption des Algorithmus, welcher im Grunde auf Zufallsergebnissen beruht, nachlässt. Die derzeitige Entwicklung muss somit auch an diesem Punkt weiter vorangetrieben werden. Ein Konzept zur Behebung dieser Schwäche soll im Kapitel 6.1.1 „Dekompositionsverfahren“ zur Sprache kommen.

6 Ausblick und Zusammenfassung

6.1 Verbesserungspotenziale

Es ist schnell zu erfassen, dass Metaheuristiken wie der Genetische Algorithmus in größeren Projekten mit 500 oder 1000 Aktivitäten Akzeptanzgrenzen für Performance und Qualität der Lösungen nicht mehr einhalten können. In einem solchen Umfang ist eine Lösung, gestützt auf die „Activity List"-Variante des GA, immer mit Zufall verbunden und eine gezielte Verbesserung kann kaum gezielt hervorgebracht werden. Während auf der einen Seite für eine Sequenz in der Liste eine verbesserte Reihenfolge ermittelt wird, ist für eine andere das genaue Gegenteil der Fall. Auch der Rechenaufwand erreicht für den Nutzer unakzeptable Werte. In diesem Kapitel soll nun ermittelt werden, welche Möglichkeiten es gibt, die angesprochenen Lösungen für den Praxiseinsatz zu verbessern. Dabei soll erstens das Dekompositionsverfahren vorgestellt werden, welches speziell für realweltliche Formate von 500, 1000 oder mehr Aktivitäten von Interesse sein wird. Zweitens ist es notwendig, Implementierungskonzepte auch im Hinblick auf moderne technische Rahmenbedingungen zu analysieren.

6.1.1 Dekompositionsverfahren

Gentner [Gent] stellt zwei Arten von Dekompositionsverfahren vor: die Aktivitäten- und die Ressourcenbasierte Verfahrensweise. Im Folgenden möchte der Autor auf die Aktivitäten-basierte Dekomposition eingehen, um das generelle Prinzip dieser Herangehensweise zu beleuchten und gleichzeitig den Bezug zum „Activity List"- GA zu nutzen.

Die Grundlage dieser Methode ist das Vorhandensein einer Zerlegung der Aktivitätenmenge, aus welcher eine bestimmte Anzahl von Teilproblemen modelliert und eine Lösungsreihenfolge abgeleitet wird. Im Anschluss werden diese Teilprobleme gelöst und durch „Verkettung" zu einem vollständigen Schedule zusammengefasst.

Im ersten Dekompositionsschritt wird eine Zerlegung $Z(V)$ der Aktivitätenmenge V bestimmt, wobei jede Teilmenge $Z \in Z(V)$ genau einer zu lösenden $PS|temp|C_{max}$ – Instanz entspricht. Gentner [Gent] behandelt in seiner vorrangig dieses Modell des RCPSP. Da es jedoch, wie bereits angesprochen, einer erweiterten Form des $PS|prec|C_{max}$ entspricht, kann es ohne weiteres in diesen Kontext übernommen werden. Anschließend werden im nächsten Schritt die Teilprobleme modelliert, wobei V_p der Aktivitätenmenge des Teilproblems entspricht und P das Indize für das zu jeweilige Teilproblem darstellt. Zu dieser Teilmenge Z_p ($p \in P$)wird ein künstlicher Projektstart α_p und ein Projektende ω_p hinzugefügt. Es wurde demzufolge ein vollständig neuer Netzplan ermittelt, der unabhängig gelöst werden soll. Aufgrund der Zerlegung der Aktivitäten kommt es zwischen den einzelnen Teilproblemen zu gegenseitigen

Abhängigkeiten (Interdependenzen), die bei der Modellierung zusätzlich beachtet werden müssen. Im Zuge der Ermittlung der Teilproblemlösungen wurde ein zulässiger Startzeitpunktvektor $S^P := (S^P_i)_{i \in Vp}$ durch Anwendung eines geeigneten Algorithmus ermittelt.

An dieser Stelle ist es, nach Ansicht des Autors, angemessen, bis zu einer bestimmten Teilproblemlösung einen exakten Algorithmus, wie das Branch & Bound Verfahren, zum Einsatz kommen zu lassen. Wie bereits beschrieben ist dies bis zu einer Menge von 20 bis 30 Aktivitäten aus Performancesicht sinnvoll. Die Definitionen bezüglich der Nebenbedingungen und Optimalität des Ausgangsproblems gelten analog für die Teilproblemschedules. Sind alle Lösungen erfasst, entsteht durch eine Komposition eine zulässige Lösung für das Ausgangsproblem. Dazu werden die Teilproblemlösungen gemäß der definierten Reihenfolge zu so genannten Synchronisationszeitpunkten miteinander verkettet (diese werden während des Verfahrensablaufes bestimmt).

Die Bestimmung von Zerlegung und Teilproblemreihenfolgen kann auf zwei verschiedene Arten erfolgen. Einerseits kann die Aktivitätenmenge in mehreren Iterationsschritten aufgeteilt werden. Nach der Bestimmung einer Teilmenge wird anschließend das dazugehörige Problem modelliert und gelöst. Da es sich hierbei um eine schrittweise Generierung der einzelnen Teilprobleme handelt, wird diese Vorgehensweise auch Sukzessiv-Zerlegung genannt. Demgegenüber kann eine Bestimmung der Teilmengen auch innerhalb eines Iterationsschrittes erfolgen. Es wird zuerst eine vollständige Zerlegung der Aktivitätenmenge gebildet und für diese eine Reihenfolge ermittelt. Erst danach kommt es zu einer Modellierung und Lösung der Teilprobleme. Dieses Vorgehen wird auch als Simultan-Zerlegung bezeichnet. Nach Gentner [Gent] eignet sich die Sukzessiv-Zerlegung bei Problemen der ressourcenbeschränkten Projektplanung aus der Praxis.

6.1.2 Nutzung von Multi-Threading in modernen Prozessorarchitekturen

Seit den letzten Jahren zeichnet sich bei der Entwicklung neuer Prozessor-Typen eine entscheidende Trendwende ab. So wurde bereits im Jahre 2004 auf einer Entwicklerkonferenz des Chip-Hersteller Intel angekündigt, Mehrprozessorarchitekturen auch für den PC- und Laptop-Markt anbieten zu wollen (Quelle Ziesche [Ziesche]). Mittlerweile sind sich die Marktführer Intel und AMD einig, dass die Zukunft der Prozessor-Technologie den Doppel- bzw. Mehrkern-Prozessoren gehört (Quelle: [Win] S.118). So sind die Zeiten vorbei, in der Multi-Prozessor-Architekturen ausschließlich für Profi-Anwender in Frage und als Workstation oder Server zur Anwendung kamen. Mittlerweile sind Doppelkern-Prozessoren nicht nur in Desktop- PCs, sondern auch in Notebooks wiederzufinden.

Da die Software-Entwicklung der Hardware-Evolution jedoch weit hinterherläuft, werden die Vorteile der neuen Technologie bisher kaum genutzt. So sind beispielsweise Hyper-

Threading-Prozessoren mit einem zweiten virtuellen Prozessor seit vier Jahren für Workstations und seit mehr als drei Jahren auch für Desktop-PCs erhältlich. Dem gegenüber verfügen nur wenige aktuelle Anwendungen über eine Unterstützung mehrerer Threads. So ist mit einer wirklichen Entwicklung in diesem Bereich möglicherweise erst mit der Veröffentlichung von Windows Vista zu rechnen. Der Nachfolger von Windows XP soll die Unterstützung von Mehrkern – Prozessoren verstärkt unterstützen.

Um modernen Prozessorarchitekturen richtig nutzen zu können, ist es notwendig, moderne Programmierkonzepte zu entwerfen und Programmiersprachen in diese Richtung zu erweitern. Ziesche [Ziesche] gibt in diesem Rahmen Hinweise, wie auf Basis derzeitiger Programmiersprachen ein Ansatz zur Umsetzung von nebenläufigen Programmen durchzuführen ist. Im Folgenden sollen zentrale Begriffe zu diesem Thema zur Sprache kommen.

In der nebenläufigen Programmierung werden Software-Systeme mit mehreren Kontrollflüssen entwickelt. Bisherige sequentielle Vorgehensweisen haben demgegenüber nur einen Kontrollfluss. So sollen mehrere Prozessoren, oder Prozessor-Kerne, gleichzeitig an der Erfüllung einer gemeinsamen Aufgabe arbeiten. Das Betriebssystem verwaltet und steuert die Abarbeitung unterschiedlichen Kontrollflüsse bzw. Anweisungen innerhalb eines Programms als so genannte „threads“. So sind bei nebenläufigen Programmen mehrere threads innerhalb eines Prozesses aktiv und können auf die gleichen Daten zugreifen und damit arbeiten. Der Austausch von Informationen muss fehlerfrei und unter Vermeidung von Deadlocks organisiert werden. Ein Prozessor kann immer nur eine Anweisung oder thread zur gleichen Zeit bearbeiten.

Um bei einem Ein-Prozessor-System Nebenläufigkeit zu simulieren, schaltet das Betriebssystem sehr schnell zwischen mehreren threads um. Eine echte parallele Bearbeitung kann daher nur auf mehreren Prozessoren stattfinden. Als Programmierer, so bemerkt Ziesche [Ziesche] an dieser Stelle, „ ... sollte man immer so programmieren, als stünde für jeden thread ein Prozessor zur Verfügung“. Auch in einem Mehr-Prozessor-System wird es vorkommen, dass mehr threads als Prozessoren zur Bearbeitung bereitstehen. So ist ein Umschalten, welches als scheduling bezeichnet wird, etwas komplexer als in einer Ein-Prozessor-Architektur und wird durch Algorithmen gesteuert.

Nach diesen Überlegungen muss nun überprüft werden, ob sich diese Technologieentwicklung für die Berechnung des RCPSP sinnvoll nutzen lässt. Am naheliegendsten ist die simultane Berechnung mehrerer „Elternpaare“ in einem Generationsschritt im Genetischen Algorithmus. Die Ermittlung eines Kind-Individuums kann demnach auf einem thread abgelegt werden, wobei zur gleichen Zeit andere Lösungsmöglichkeiten berechnet werden.

Auf der anderen Seite ist es möglich, bei einer Dekomposition die Berechnung einzelner Teilprobleme auf mehrere threads und damit Prozessorkerne zu verteilen. Diese Vorgehensweise wird die Bearbeitungsdauer des Ausgangsproblems merklich verkürzen.

6.2 Zusammenfassung

Die vorliegende Arbeit hat gezeigt, dass etablierte Programme wie MS Project dem heutigen Wissen auf dem Gebiet des Project Scheduling nicht gewachsen sind. Der Autor hat nachgewiesen, dass es möglich ist, dass PS|*prec*| C_{max} unter Verwendung von derzeit aktuellen Algorithmen zu lösen und damit einen für die Projektpraxis entscheidenden Wettbewerbsvorteil zu nutzen. Es lässt sich festhalten, dass es auch unter Inkaufnahme von Leistungseinbußen bei der Berechnung der Projektpläne in Hinblick auf den entscheidenden Kostenvorteil lohnend ist, Metaheuristiken in PM-Systemen anzuwenden. Weiterreichende praktische Anforderungen an ein umfassendes Modell zur Darstellung des RCPSP wurden dargestellt (siehe Kuhlmann [Kuhl]). SimProQ ist demzufolge eines der ersten PM-Systeme, welche seinen Nutzern Metaheuristiken für diesen Problemkontext anbieten kann. Dies bedeutet unter Nichtbeachtung anderer entscheidender Programmfunktionen und Features einen klaren Wettbewerbsvorteil gegenüber anderen Programmen. Für eine zukünftige praktische Umsetzung des Problems muss an dieser Stelle überprüft werden, ob das von Kuhlmann [Kuhl] erstellte Modell und der daraus entwickelte Genetische Algorithmus vollständig implementiert werden kann.

Ein weiterer wichtiger Punkt waren die Performancekennzahlen der vorgestellten Algorithmen. Im Unterkapitel 5.1 konnte unter Verweis auf die neuesten Entwicklungen im Bereich der Prozessorherstellung veranschaulicht werden, dass dieses Problem hinreichend eingedämmt werden kann. So ist es in Zukunft möglich, Technologien wie Hyper-Threading gezielt auf dem Gebiet des Projekt-Scheduling zu nutzen.

Es wird überdies deutlich, dass bereits bei einem verhältnismäßig kleinen Projektumfang etablierte PM-Systeme deutlich scheitern. Im Hinblick auf realweltliche Probleme mit 500, 1000 oder mehr Aktivitäten ist mit herkömmlichen Heurisiken, wie sie in derzeitigen Softwarepaketen zum Einsatz kommen, keine vertretbare Lösung zu ermitteln. Auch für dieses Problem wurde unter 6.1.1 ein Ansatz gefunden. Das Dekompositionsverfahren ermöglicht es, das Ausgangsproblem in mehrere Teilprobleme aufzuspalten. Daraus ergeben sich neue Möglichkeiten zur Verbesserung der Qualität der Lösungen und einer Verringerung der Rechenzeit bei deren Berechnung.

Anhang - Berechnung eines Beispielprojektes

In diesem Abschnitt soll nach der theoretischen Betrachtung aus dem Hauptteil der Arbeit ein ergänzender praktischer Vergleich zwischen den hier dargestellten Algorithmen und einer Lösung in MS Projekt mithilfe eines Beispielprojektes veranschaulicht werden. Neben den Tests aus Kapitel 4 soll an dieser Stelle nochmals nachgewiesen werden, dass durchaus Verbesserungspotenzial besteht.

Zunächst soll das Projekt formal beschrieben werden:

Vorgang	Dauer	Vorgänger	Nachfolger	Ressourcen
TaskA	13	-	TaskE, TaskF, TaskD	1,2,3
TaskB	6	-	TaskF	1,2
TaskC	22	-	TaskG, TaskF	1,3
TaskD	34	TaskA	TaskH	2
TaskE	15	TaskA	TaskI,TaskH	1,3,4
TaskF	7	TaskC, TaskA, TaskB	TaskJ	2,3
TaskG	19	TaskC	TaskK	1,2
TaskH	11	TaskE, TaskD	TaskM, TaskL	1
TaskI	18	TaskE	TaskM	3,4
TaskJ	9	TaskF	TaskM, TaskO,TaskN	3,4
TaskK	24	TaskG	TaskQ	2,4
TaskL	28	TaskH	TaskP	3
TaskM	16	TaskJ, TaskI, TaskH	TaskP	4
TaskN	5	TaskJ	-	2,3,4
TaskO	43	TaskJ	TaskQ	1,3,4
TaskP	33	TaskM, TaskL	-	1,2

TaskQ	32	TaskK, TaskO	-	2,4

Tabelle 7: Projektbeschreibung des Beispiels

Aus der Tabelle geht hervor, dass „TaskA“, „TaskB“ und „TaskC“ keinen Vorgänger besitzen und somit zum FAZ 0 starten können. Des Weiteren ist jedoch zu sehen, dass sich alle drei Vorgänge die Ressource zwei und drei teilen, was zur Folge hat, dass beide Ressourcen zu Beginn des Projektes bereits mit 133 % Leistung überbeansprucht sind. So ergibt anhand der Tabelle eine Reihe von Ressourcenkonflikten, die nach Berechnung von FAZ und FEZ auftauchen.

Eine Berechnung des Problems brachte mithilfe von MS Project folgende Lösung:

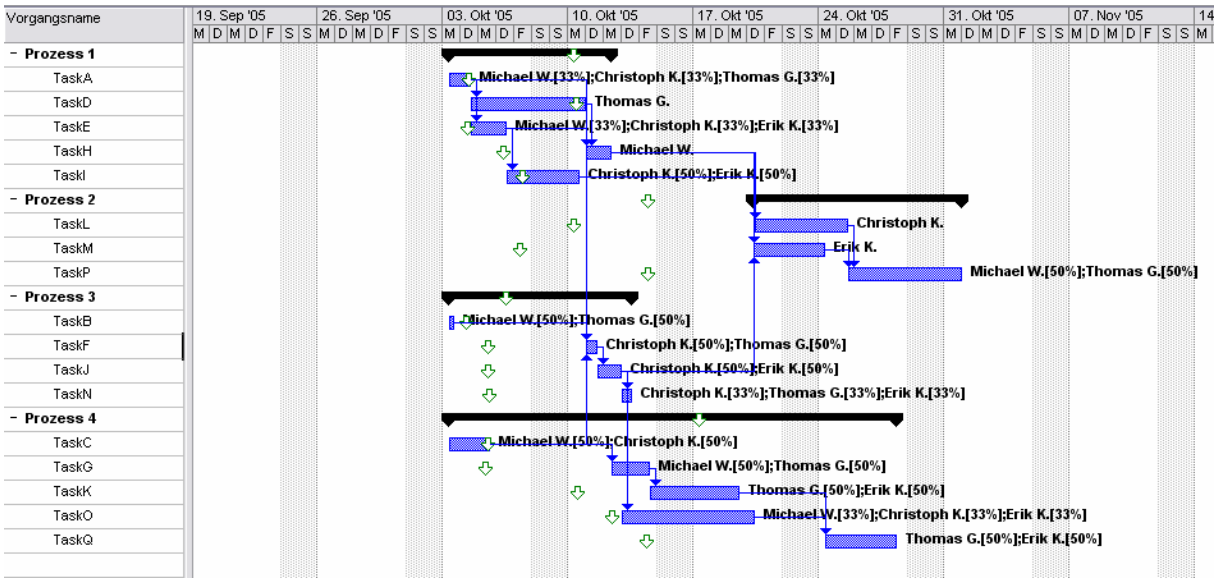


Abbildung 10: Gantt-Diagramm - Lösung in MSP

Es ist ersichtlich, dass der letzte Vorgang – „TaskP“ – erst mit dem 31.10.2005 abgeschlossen und damit das Gesamtprojekt mit diesem Datum beendet wird. Ein Blick in die XML-Datei ergibt einen genauen Zeitpunkt von 17.00 Uhr für das Ende des Projektes. Hingegen ergibt eine Berechnung mit dem vorgestellten Genetischen Algorithmus die folgende Lösung:

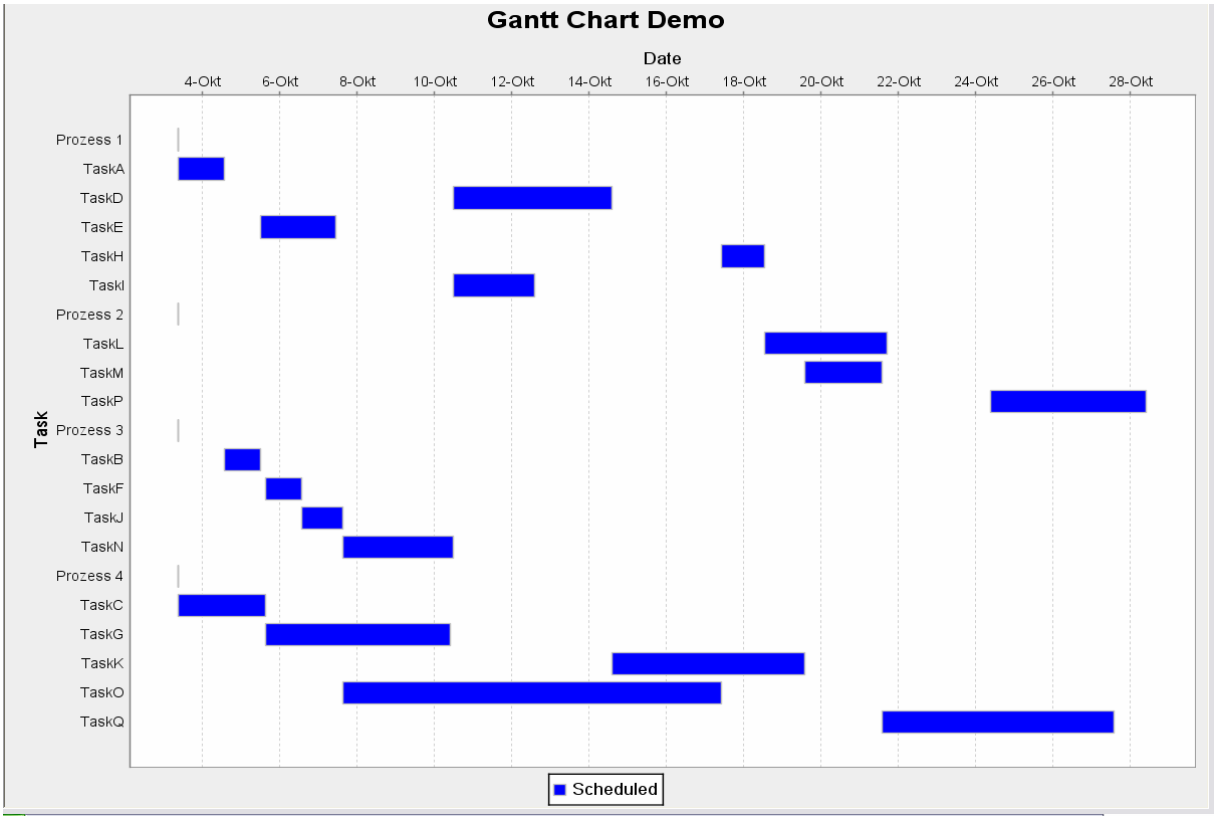


Abbildung 11: Gantt-Diagramm - Lösung mit dem GA

Hier ist „TaskP“ bereits am 28. 10. 2005 10.00 Uhr beendet. Die Arbeitszeiten betragen in beiden Systemen acht Stunden, Beginn um 9.00 Uhr und Ende 17.00 Uhr. Wenn man berücksichtigt, dass der 29. und 30.10. 2005 ein Wochenende war, an dem nicht gearbeitet wurde, ergibt sich eine Gesamtdifferenz von fünfzehn Stunden, die bei einem Projekt von dieser Größe bereits einen beträchtlichen Unterschied ausmacht.

Zur Ergänzung zeigt die anschließende Grafik den Verlauf der Belastung der Ressource 2 über die Projektzeit:

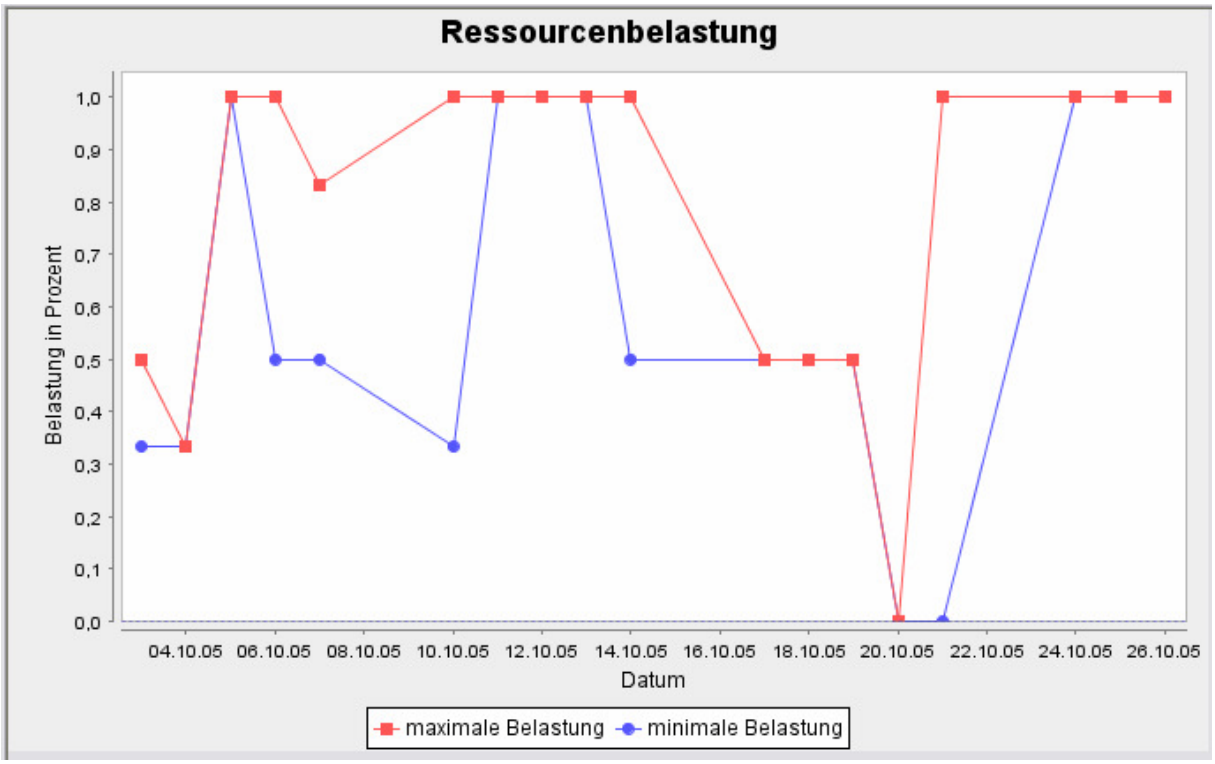


Abbildung 12: Diagramm der Ressourcenbelastung

Zunächst wird deutlich, dass zu keinem Zeitpunkt eine Überbelastung besteht; der Ressourcenausgleich wurde demnach korrekt ausgeführt. Des Weiteren wird der Mitarbeiter über lange Zeitperioden gut ausgelastet. Ein solches Diagramm kann dem Modellierer somit eine gute Übersicht über die Belastungskurve eines Mitarbeiters geben. Es ist offensichtlich, dass in der Zeit vom 14.10.2005 bis 21.10.2005 durchaus noch Reserven mobilisiert werden können.

Literaturverzeichnis

[AhlBa] Ahlemann, F./Backhaus, K. (2006) Project Management Software Systems – Requierements, Selection Process and Products. Nürnberg

[BarKlee] Bartsch-Beuerlein, S./ Klee,O. (2001) Projektmanagement mit dem Internet – Konzepte und Lösungen für virtuelle Teams. Wien

[Breiten] Breitenfeld, S. (2005) Projektmanagement– Vorlesungsscript. Magdeburg

[BruKnu] Brucker, P./ Knust, S.(2006) Complex Scheduling. Berlin, Heidelberg, New York

[Burgh] Burghardt, M. (2002) Einführung in Projektmanagement – Definition, Planung, Kontrolle, Abschluss. Berlin/München

[Carl] Carl, D. (2006) Praxiswissen Ajax. Berlin

[Win] Windeck, C. (2006) Mit doppelter Kraft – Die Dual-Core-Technik kommt in Fahrt. c`t 4/2006, S118

[Dan] Danis, M. (2005) Learning Algorithms in Optimization of Project Scheduling in Microsoft Project 2003. Stockholm

[DeVan] Debels, D./ Vanhoucke, M. (2005) A decomposition - based heuristic for the Resource–Constraint-Scheduling-Problem. Ghent

[DomDre] Domschke, W./ Drexl, A.(2001) Einführung in Operations Research. Berlin

[DrKolSpr] Drexl, A./ Kolisch, R./Sprecher, A (1997) Koordination und Integration im Projektmanagement – Aufgaben und Instrumente. Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel. Kiel

[Dum] Dumke, R. (2001) Software Engineering – Eine Einführung für Informatiker und Ingenieure: Systeme, Erfahrungen, Methoden, Tools. Magdeburg

[Gent] Gentner, K. (2005) Dekompositionsverfahren für die ressourcenbeschränkte Projektplanung. Aachen

[Hart] Hartmann, S. (2000) Project Scheduling under Limited Resources – Models, Methods, and Applications. Heidelberg

[HaMelTau] Hartung, T./ Mellentien, C./Trautmann, N. (2001) Software zur ressourcenbeschränkten Projektplanung im Vergleich. Karlsruhe

[HartKol] Hartmann, S./Kolisch, R(1998) Experimental Evaluation of State – of – the – Art Heuristics for the Resource-Constrained Project Scheduling Problem.

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel. Kiel

[Kol] Kolisch, R. (1995) Project Scheduling under Resource Constraints – Efficient Heuristics for Several Problem Classes. Heidelberg

[Kuhl] Kuhlmann, A.(2003) Entwicklung eines praxisnahen Project Scheduling Ansatzes auf der Basis von Genetischen Algorithmen. Hagen

[MS] http://office.microsoft.com/ (Zugriff 30.06.2006)

[MSP] Kuppinger M./Reinke H./ Jäger M. (2000) Microsoft Project 2000 Das Handbuch – Das ganze Softwarewissen. Unterschleißheim

[Neus] Neus, W. (1998) Einführung in die Betriebswirtschaftslehre

[NeuSchwZi] Neumann, K. / Schwindt, C./ Zimmermann, J. (2003) Project Scheduling with Time Windows and Scare Resources, Second Edition.

[PatzRat] Patzack, G. /Rattay, G.. (2004) Projektmanagement– Leitfaden zum Management von Projekten, Projektportfolios und projektorientierten Unternehmen. Wien

[Peters] Peters, Malte L./ Zelewski S. (2002) Analytical Hierarchy Process (AHP) – dargestellt am Beispiel der Auswahl von Projektmanagement-Software zum Multiprojektmanagement. Essen

[PolJohn] Pollack-Johnson, B. (1995) Hybrid structures and improving forecasting and scheduling in project management. Journal of Operations Management.

[SchnHieb] Schneider, Welf G./ Hieber, D. (1997) Software zur ressourcenbeschränkten Projektplanung. Karlsruhe

[Schwindt] Schwindt, C. (2005) Resource Allocating in Project Management. Berlin/Heidelberg/New York

[SchwTool] Schwab, J. (2004) Projektmanagement mit Tool-Unterstützung – Methodisches Projektmanagement mit MS Project. Berlin

[SchwMSP] Schwab, J (2005) Projektplanung realisieren mit MS Project 2003 und Project Server 2003. Berlin

[Sprech94] Sprecher, A. (1994) Resource-constrained project scheduling - exact methods for the multi-mode case. Berlin

[Sprech96] Sprecher, A (1996) Solving the RCPSP efficiently at modest memory requierements. Manuskripte aus den Instituten der Betriebswirtschaftslehre Kiel

[Wiki] http://de.wikipedia.org/ (2006)

[Ziesche] Ziesche, P. (2005) Nebenläufige & verteilte Programmierung – Konzepte, UML 2-Modellierung, Realisierung mit Java 1.4 & Java5

Abschließende Erklärung

Ich versichere hiermit, dass ich die vorliegende Diplomarbeit selbständig, ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Leipzig, den 12.10.2006

.....

Maik Gruhne