technische universität
dortmund

Dezentrales Grid-Scheduling mittels Computational Intelligence

Joachim Lepping

Genehmigte Dissertation zur
Erlangung des akademischen Grades eines Doktor-Ingenieurs
der Fakultät für Elektrotechnik und Informationstechnik an der
Technischen Universität Dortmund

Dortmund
29. März 2011

Danksagung

Faktisch alle wissenschaftlichen und industriellen Projekte lassen sich nur durch eine eng vernetzte und kollegiale Zusammenarbeit zum Erfolg führen. In diesem Sinne ist auch eine Dissertation, wenngleich diese primär von einem einzelnen Forscher vorangetrieben wird, ein Projekt, das sich nur durch die Unterstützung einer Vielzahl von Menschen, durch Diskussionen, Anmerkungen und Kommentare, aber auch durch kollegiale und freundschaftliche Hilfen realisieren lässt.
Mein Dank gilt deshalb an erster Stelle Prof. Dr.-Ing. Uwe Schwiegelshohn, der mir die Bearbeitung des Teilprojektes „Schedulingverfahren für Computational Grids“ im Rahmen des Sonderforschungsbereiches 531 „Design und Management komplexer technischer Prozesse und Systeme mit Methoden der Computational Intelligence“ übertragen und mir damit die thematische Grundlage für die vorliegende Arbeit bereitgestellt hat. Als Zweitgutachter hat sich freundlicherweise Prof. Dr. rer. nat. Eyke Hüllermeier bereiterklärt.
Weiterhin gilt mein Dank allen Mitarbeiterinnen und Mitarbeitern des Instituts für Roboterforschung an der TU Dortmund, die mich durch ihre Diskussionsbereitschaft, ihre Kommentare und Anregungen motiviert und unterstützt haben. Besonders zu erwähnen sind hier meine engen Kollegen Alexander Fölling, Christian Grimme und Alexander Papaspyrou. Durch die Zusammenarbeit mit ihnen ist ein Großteil der hier vorliegenden Ergebnisse entstanden. Weiterhin danke ich Stefan Freitag, Peter Resch und Lars Schley für die technische Unterstützung bei der Durchführung umfangreicher Simulationen. Für die sorgfältige Durchsicht und Korrektur des Manuskriptes danke ich besonders Sandra Finnenkötter.

1. Gutachter:	Prof. Dr.-Ing. Uwe Schwiegelshohn
2. Gutachter:	Prof. Dr. rer. nat. Eyke Hüllermeier
Tag der Einreichung:	24. November 2010
Tag der Prüfung:	28. März 2011

Abstract

The ever-growing need for universally available computing and storage capacity is more and more satisfied by new architectures for networked interaction between users and providers of computing resources. Today, research and industry have realized an infrastructure for the coordinated use of globally distributed computing resources. The so-called grid computing infrastructure is assumed to be an integral part of the future global resource landscapes. In such an environment, submitted computing jobs are no longer bound locally, but can be flexibly migrated between different resource providers. In the context of community grids, different users are organized into virtual organizations, which typically share—in addition to a joint research—various computing resources. However, cooperation among different virtual organizations at the moment occurs only very rarely as—besides technical issues—this requires more advanced decentralized grid scheduling concepts. In order to fully utilize the capabilities of a federated grid, it is essential to promote a cross-community cooperation in the sense of a global grid infrastructure. At the same time, however, it is most important that the local autonomy is maintained, as no virtual organization would voluntarily cede the control of their local resources.

Thus, the interaction among different communities in the grid brings both opportunities and challenges: The newly formed flexibility makes users even more demanding with respect to computing result delivery and wait times. The efficient operation of future computational grids highly depends on the development of viable architectures and strategies for scheduling, i.e. the allocation of jobs to resources and powerful methods for the negotiation of job migrations.

In this work, we therefore develop distributed scheduling strategies for computational grids using various methods from computational intelligence. Different virtual organizations are seen as autonomous entities that decide on the acceptance or decline of jobs. Here, jobs can be offered by other schedulers or by the local user communities. The authority and security of virtual organizations will be maintained by following a restrictive information policy that strongly limits the exchange of system state information. The fully decentralized grid structure guarantees that the scheduling concepts are also applicable in large-scale environments. Initially, several decentralized strategies are developed and tested by extensive simulations with real-world workload traces. The results give first insights into the dynamics and characteristics of the assumed grid federations. Based on these findings, the mechanisms for decision-making is refined and implemented in a newly designed modular scheduling architecture. Using an evolutionary fuzzy system, the decision-making and interaction between virtual organizations is realized and further optimized. In a last step, also a coevolutionary algorithm is applied to improve the scheduling decisions. The evaluation based on real workload recordings reveals that it is possible to achieve significantly shorter response times for all respective user communities. At the same time, we demonstrate a strong robustness of the procedures, both to changing grid environments and changed user behavior. The results can be seen as a motivation for the increased cross-community cooperation in terms of a global computational grid, as this results in a win-win situation for all participants.

Zusammenfassung

Das ständig wachsende Bedürfnis nach universell verfügbarer Rechen- und Speicherkapazität wird durch die in den letzten Jahren vorangetriebene Entwicklung neuer Architekturen für die vernetzte Interaktion zwischen Nutzern und Anbietern von Rechenressourcen mehr und mehr erfüllt. Dabei ist die Umsetzung einer Infrastruktur zur koordinierten Nutzung global verteilter Rechenressourcen längst in Forschung und Wirtschaft realisiert worden. Diese als Grid-Computing bezeichnete Infrastruktur wird künftig integraler Bestandteil der globalen Ressourcenlandschaft sein, sodass die Ausführung von lokal eingereichten Berechnungsaufgaben nicht mehr ortsgebunden ist, sondern flexibel zwischen unterschiedlichen Ressourcenanbietern migriert werden kann. Bereits heute sind unterschiedliche Nutzergemeinschaften im Rahmen von Community-Grids in virtuellen Organisationen zusammengefasst, die neben einem gemeinsamen Forschungs- oder Anwendungsinteresse auch häufig eine Menge von IT-Ressourcen gemeinsam nutzen. Ziel ist es aber, auf lange Sicht eine Community-übergreifende Kooperation im Sinne einer globalen Grid-Infrastruktur unter Wahrung lokaler Autonomie weiter zu fördern.

Dabei bringt die Interaktion mit anderen Communities im Grid sowohl Chancen als auch Herausforderungen mit sich, da durch die Nutzbarkeit global verteilter Ressourcen auch höhere Anforderungen in Bezug auf Berechnungsgeschwindigkeit und Wartezeiten von Seiten der Nutzer gestellt werden. Der Schlüssel für den effizienten Betrieb künftiger Computational Grids liegt daher in der Entwicklung tragfähiger Architekturen und Strategien für das Scheduling, also in der Zuteilung der Jobs zu den Ressourcen. Bisher sind die Methoden für die Verhandlung von Jobübernahmen zwischen Ressourcenanbietern jedoch nur sehr rudimentär entwickelt.

In dieser Arbeit werden deshalb dezentrale Schedulingstrategien für Computational Grids entwickelt und unter Einsatz von Methoden der Computational Intelligence realisiert und optimiert. Dabei werden einzelne virtuelle Organisationen als autonome Einheiten betrachtet, die über eine Annahme oder Abgabe sowohl von eigenen als auch von extern eingereichten Jobs entscheiden. Durch die Beachtung einer restriktiven Informationspolitik werden die Autorität und Sicherheit virtueller Organisationen gewahrt und zugleich wird die Skalierbarkeit in größeren Umgebungen durch den dezentralen Aufbau sichergestellt.

Zunächst werden verschiedene dezentrale Strategien entwickelt und simulatorisch untersucht. Die Ergebnisse geben dann Aufschlüsse über die Dynamik und Eigenschaften eines derartigen Verbunds. Auf Basis der so gewonnenen Erkenntnisse werden die Mechanismen zur Entscheidungsfindung verfeinert und in einer neu entworfenen modularen Schedulingarchitektur umgesetzt. Mittels evolutionär optimierter Fuzzy-Systeme wird anschließend die Entscheidungsfindung optimiert. Die Interaktion zwischen virtuellen Organisationen wird dann alternativ mittels co-evolutionärer Algorithmen angepasst. Die auf Basis realer Arbeitslastaufzeichnungen durchgeführten Evaluationen zeigen, dass die so erstellten Grid-Schedulingstrategien für alle am Grid teilnehmenden Communities deutlich verkürzte Antwortzeiten für die jeweiligen Nutzergemeinschaften erreichen. Gleichzeitig wird eine große Robustheit der Verfahren sowohl gegenüber veränderlichen Grid-Umgebungen als auch gegenüber verändertem Nutzerverhalten bewiesen. Die Ergebnisse sind als Motivation für die stärkere Community-übergreifende Kooperation im Sinne eines Computational Grid zu sehen, da dies bei Nutzung entsprechend optimierter Verfahren in einer Win-win Situation für alle Teilnehmer resultiert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Die bereits vor mehreren Jahrzehnten formulierte Vision einer ständigen globalen Verfügbarkeit und transparenten Nutzbarkeit allgemeiner Rechen- und Speicherressourcen ist in den letzten Jahren durch die verstärkte Umsetzung von nationalen und internationalen Projekten immer mehr zur Realität geworden. Die konsequente Förderungspolitik zielt auf den Aufbau einer Infrastruktur, die den ständig steigenden Anforderungen moderner Wissenschaften und innovativer Unternehmen nach aufwendigen Simulationen und hochgradig rechenintensiven Entwicklungen jetzt und auch in Zukunft gerecht wird. Die gleichzeitige Einführung von Multikernprozessor-Architekturen führt zur notwendigen Anpassung bestehender Anwendungen an hochgradig parallele Hardware, weshalb gerade die schnelle und effiziente Abarbeitung großer Berechnungen immer mehr in den Fokus der Unternehmen rückt. Dies fördert ein stärkeres Bewusstsein für die Nutzung von Höchstleistungsrechnern zur fristgerechten Erfüllung von Kundenanforderungen und führt dazu, dass die Verfügbarkeit von Rechenleistung ein immer entscheidender werdender Wettbewerbsfaktor für Unternehmen wird.

Aufgrund einer stark nutzerorientierten Ausrichtung hat sich das sogenannte Grid-Computing als eine Schlüsseltechnologie für die universelle Umsetzung einer Infrastruktur zur koordinierten Nutzung global verteilter Rechenressourcen sowohl in der Forschung als auch in der Wirtschaft etabliert. Es kann mit Sicherheit davon ausgegangen werden, dass Grid-Computing-Infrastrukturen künftig integraler Bestandteil einer globalen Ressourcenlandschaft sind, sodass die Ausführung einer von lokalen Nutzergemeinschaften eingereichten Berechnungsaufgaben nicht mehr ortsgebunden ist, sondern flexibel zwischen unterschiedlichen Ressourcenanbietern migriert werden kann. Die geschickte Koordination und Verhandlung derartiger Migrationen und Delegationen sowie die sinnvolle Zuordnung von Ressourcen zu Berechnungsaufgaben (Scheduling) sind eine der wichtigsten Herausforderungen, da sie grundlegenden Einfluss auf den effizienten Betrieb einer solchen Infrastruktur haben.

In einem Computational Grid sind dabei zunächst zwei Rollen voneinander zu unterscheiden. Auf der einen Seite befinden sich die Ressourcenanbieter, die üblicherweise im akademischen Umfeld an großen Rechenzentren, aber im vermehrten Maße auch in Wirtschaftsunternehmen zu finden sind. Sie betreiben häufig große Cluster- und Parallelrechnerinstallationen, die nicht selten einige tausend Prozessoren umfassen. Die Zuordnung von Berechnungsaufgaben (Jobs) zu den unter einer Verwaltungshoheit organisierten Ressourcen wird durch lokal angepasste Schedulingsysteme realisiert, wobei dort üblicherweise spezielle Präferenzen und Prioritäten umgesetzt werden.

Auf der anderen Seite befinden sich die Nutzer oder Nutzergruppen, die ihre teilweise sehr aufwendigen Berechnungen nun auf externen Ressourcen zur Ausführung bringen wollen.

Aus der Sicht einer Nutzergruppe ist dabei neben der ohnehin immer vorhandenen Forderung nach einer sicheren und verschlüsselten Ausführung vor allem die schnelle und zuverlässige Abarbeitung der eigenen Jobs von Interesse.

Historisch gesehen hat sich dieses Szenario bereits seit der Einführung großer Parallelrechner in ähnlicher Form gebildet. Dort stellten die Betreiber einer Ressource Schnittstellen zur Verfügung, die von einer Nutzergruppe zur Einreichung ihrer Jobs genutzt wurden. Traditionell wurden derartige Einreichungen besonders von Nutzern aus dem akademischen Umfeld wie Hochenergiephysik, Astronomie oder auch der Klimaforschung zur Berechnung und Simulation ihrer Modelle genutzt. Der Begriff einer Nutzergruppe bezeichnet hier also eine Gemeinschaft, deren Mitglieder ähnliche Interessen verfolgen oder auch gleiche Anwendungen nutzen und üblicherweise in der Vergangenheit an die Nutzung einer lokalen Rechenressource gebunden waren. Dieses lang erprobte Szenario kann als Ausgangspunkt für die Bildung einer Grid-Infrastruktur gesehen werden.

Im Grid organisieren sich wiederum die Nutzer mit ähnlichen Anwendungen oder gleichen Forschungsinteressen (Communities) in sogenannten *virtuellen Organisationen*, die dann regional übergreifend unter Nutzung von Höchstleistungsrechnern an der koordinierten Lösung von Problemen arbeiten. Dabei können virtuelle Organisationen selbst Ressourcen zur eigenen Nutzung bereitstellen oder diese von externen Ressourcenanbietern nutzen. In jedem Fall bestehen für die von einer virtuellen Organisation genutzten Ressourcen die Möglichkeit beliebige Priorisierungen und Ressourcenzuteilungen umsetzen.

Die Strukturen eines Computational Grid unterstützen nun genau diese Szenarien, indem dort die dynamische Zuordnung von Rechenressourcen zu unterschiedlichen virtuellen Organisationen realisiert werden kann. Das intelligente Scheduling derartiger Zuweisungen erlaubt einen besseren Lastenausgleich zwischen den unterschiedlichen Ressourcen bei Bedarfsspitzen und eine gleichzeitige Umverteilung der Jobs in möglichen Unterlastsituationen. Dabei ist aber besonders darauf zu achten, dass die möglichen Präferenzen und Befindlichkeiten der einzelnen Ressourcenverwalter berücksichtigt werden und sich für die Nutzer möglichst kurze Antwortzeiten für die Bearbeitung ihrer Jobs ergeben.

Im Hinblick auf das Scheduling in existierenden Grids werden aktuell vor allem hierarchische und zentralistische Ansätze auf Basis einer allgemein verwendeten Koordinationsschicht, der sogenannten *Middleware,* umgesetzt. Dabei besteht nicht nur für die Ressourcenanbieter, sondern auch für die Nutzer die Notwendigkeit, sich diesen vorgegebenen Strukturen anzupassen, wobei fraglich ist, inwieweit ein derart zentraler Ansatz für die Erweiterung des Grid skalierbar ist. Weiterhin bedarf es natürlich vieler Abstimmungen, um sich unter derart vielen Partnern und Nutzern mit teilweise gegensätzlichen Interessen auf eine einheitliche Middleware und Schedulingarchitektur zu verständigen.

Ein alternativer Ansatz wurde von vornherein im Rahmen von nationalen Grid-Projekten wie zum Beispiel der deutschen D-GRID Initiative verfolgt, bei denen der sogenannte Service-Grid-Gedanke im Vordergrund der Entwicklung steht. Dort werden zunächst kleinere Grid-Lösungen für einzelne Communities erstellt, die genau auf die Bedürfnisse von Nutzergruppen zugeschnitten sind. Sie sind zumeist dann ebenfalls in einer virtuellen Organisation zusammengefasst und beinhalten häufig einen integrierten oder zumindest assoziierten Ressourcenanbieter. Das Ressourcen-Management ist für ein Community-Grid nun entsprechend angepasst, sodass für die zugreifbaren Ressourcen Priorisierungen und Präferenzen durch lokale Ressourcen-Management-Systeme realisiert werden. Aus abstrakter Sicht betrachtet, bestehen also starke Gemeinsamkeiten zwischen derartigen Community-Grids und

den schon beschriebenen Parallelrechnern mit ihrer dedizierten Nutzergemeinschaft und einem angepassten Ressourcen-Management-System.
Die Zusammenarbeit zwischen den einzelnen Community-Grids wird allerdings zunehmend notwendig, da auch innerhalb einer derart angepassten Gemeinschaft die Nachfrage nach Rechenressourcen ständig steigt und diese oft lokal nicht mehr befriedigt werden kann. Außerdem kann es nötig sein, spezielle Dienste oder Ressourcen temporär innerhalb einer virtuellen Organisation zu nutzen, die lokal zwar nicht verfügbar, aber eventuell in der virtuellen Organisation einer anderen Community vorhanden sind. Deshalb gibt es seit einiger Zeit Bestrebungen, die Grenzen zwischen bereits installierten Community-Grids zu überwinden, und zu einer stärkeren Zusammenarbeit, Synergien und allgemein einem stärkeren Austausch zwischen den Communities zu gelangen. Dies stellt eine zentrale Herausforderung für die Weiterentwicklung der zahlreich existierenden Community-Grids dar. Zum einen stellen sich technische Schwierigkeiten in den Weg, da natürlich die unterschiedlichen Schnittstellen für die Einreichung von Jobs, aber auch die Kommunikationswege nicht standardisiert und deshalb zumeist inkompatibel sind. Weiterhin ergeben sich Schwierigkeiten aus organisatorischer Sicht: Jedes Community-Grid will natürlich auch bei der Zusammenarbeit mit anderen hauptsächlich die bestmöglichen Serviceleistungen für die eigene virtuelle Organisation erzielen und ist an der Verbesserung der gesamten Grid-Effizienz nur insofern interessiert, als dass dabei die eigenen Vorteile primär erreicht werden.
Gerade dieser letzte Aspekt stellt nun aber seit Jahren ein offenes Forschungsfeld dar und ist für rein dezentrale Architekturen bisher nur sehr vereinzelt behandelt worden. Die Entwicklung von dezentralen Schedulingstrategien ist vor allem deshalb von besonderer Wichtigkeit, weil nur so den einzelnen Communities eine hinreichend große Motivation für den Zusammenschluss in größeren Grid-Verbänden gegeben werden kann. In dieser Arbeit werden deshalb Strategien entwickelt, mit denen die Arbeitslast zwischen unterschiedlichen virtuellen Organisationen oder Communities derart koordiniert wird, dass diese–im Hinblick auf übliche Messgrößen–deutliche Vorteile durch die Teilnahme am Grid gegenüber der rein lokalen Ausführung ihrer Jobs erreichen. Sind derartige Vorteile nicht erzielbar, so wird die Bereitschaft zwischen Communities, im Sinne eines globalen Grids zu kooperieren, schnell sinken, da dies nur mit zusätzlichen Koordinations- und Sicherheitsproblemen verbunden wäre, die durch keine Dienstgüteverbesserung aufgewogen würde.
Deshalb muss das dezentrale Ressourcen-Management besonders an die typischen Anforderungen von virtuellen Organisationen und Communities angepasst sein. Dazu gehört aber nicht nur eine leistungsfähige algorithmische Komponente, sondern auch die Beachtung grundsätzlicher Restriktionen und Randbedingungen. So ist es zum Beispiel essenziell notwendig, eine stark restriktive Informationspolitik seitens der virtuellen Organisationen zu unterstützen. Obwohl fast jedes Community-Grid einen eigenen Informationsdienst unterhält, werden doch jegliche Informationen bezüglich der eigenen Ressourcenauslastung oder der vergangenen Nutzeranfragen streng vertraulich behandelt und höchstens innerhalb der virtuellen Organisation zur Verfügung gestellt. Dieses Verhalten kann auch bei Betreibern von Parallelrechnern beobachtet werden, da ungern Daten über wettbewerbskritische Faktoren an potenzielle Konkurrenten weitergegeben werden. Weiterhin muss das Grid-bezogene Ressourcen-Management streng von den lokalen Ressourcen-Management-Komponenten getrennt sein. Sowohl die Betreiber von Parallelrechnern als auch virtuelle Organisationen haben üblicherweise ihre eigenen Betriebspolitiken, die durch dahingehend angepasste lokale Schedulingsysteme umgesetzt werden. Ein in einer Middleware installiertes Grid-Schedulingsystem darf zwar die entsprechend angebotenen Schnittstellen zur Ein-

reichung von Jobs nutzen, aber niemals die direkte Kontrolle über die lokalen Ressourcen übernehmen.

Hinsichtlich der algorithmischen Komponenten sind allgemein zwei zentrale Anforderungen zu berücksichtigen. Zum einen wird es notwendig sein, Schedulingstrategien mit einer Art Situationsbewusstsein auszustatten, sodass je nach aktuellem Systemzustand die Koordination der Arbeitslast und die entsprechenden Entscheidungen über die Annahme oder Weitergabe von Jobs dynamisch verändert werden können. Es handelt sich bei einem Computational Grid um einen hochdynamischen Verbund, bei dem erwartet werden kann, dass statische Strategien nur unzureichende Ergebnisse liefern können. Zum anderen werden an die Strategien hohe Robustheits- und Stabilitätsanforderungen gestellt. Die Strategien sollen möglichst allgemein verwendbar sein und sowohl in sich verändernden Grid-Szenarien als auch in veränderten Lastsituationen gute Resultate liefern.

Verbunden mit den restriktiven Randbedingungen, die nur ein Minimum an Informationen über Systemzustände und das Verhalten anderer virtuellen Organisationen zulassen, stellt sich die Entwicklung von dezentralen Schedulingstrategien als ein hoch komplexes Optimierungsproblem dar. Weiterhin handelt es sich um ein sehr rechenintensives Problem, da bei der Nutzung realer Arbeitslastaufzeichnungen Entscheidungen von bis zu 100.000 Jobs koordiniert werden müssen. Trotz dieser herausfordernden Optimierungsaufgabe soll die eigentliche Bestimmung der Schedulingentscheidung zur Laufzeit möglichst wenig aufwendig sein, sodass diese leicht in existierende Middleware-Systeme integrierbar ist.

Aufgrund dieser Anforderungen kommen hier verschiedene Methoden der Computational Intelligence sowohl einzeln als auch in hybriden Konzepten zum Einsatz. Sie haben sich seit langer Zeit auch zunehmend bei praktischen Problemen zur Lösung komplexer Optimierungsaufgaben als sehr effizient erwiesen. Dabei werden sowohl evolutionäre Algorithmen zur Optimierung als auch Fuzzy-Systeme zur Entscheidungsfindung herangezogen. Letztere bieten sich besonders an, da sie auch in nur vage beschreibbaren Situationen, die sich aufgrund der dargestellten Restriktionen ergeben werden, anwendbar sind. Die auf Basis realer Arbeitslastaufzeichnungen durchgeführten Evaluationen werden zeigen, dass die so erstellten Grid-Schedulingstrategien für alle am Grid teilnehmenden virtuellen Organisationen und Communities deutlich verkürzte Antwortzeiten für die jeweiligen Nutzergemeinschaften erreichen. Gleichzeitig wird eine große Robustheit der Verfahren sowohl gegenüber veränderlichen Grid-Umgebungen als auch gegenüber einem veränderten Nutzerverhalten nachgewiesen. Die Ergebnisse sind als Motivation für den Ausbau von Computational Grids zu sehen, da dies bei Nutzung entsprechend optimierter Verfahren in einer Win-win Situation für alle Teilnehmer resultiert.

Aufbau der Arbeit

Diese Arbeit gliedert sich insgesamt in vier Hauptteile, wobei sich die ersten zwei Teile den allgemeinen Grundlagen, dem Problemkontext und den Modellannahmen widmen und die letzten beiden Teile konkrete Entwicklungen und die Optimierung von Strategien beinhalten. In Teil I wird zunächst der Begriff der Computational Intelligence genauer gefasst und die Arbeit insgesamt in den aktuellen Stand der Forschung eingeordnet. Die hier verwendeten Methoden der Computational Intelligence wie evolutionäre Algorithmen oder Fuzzy-Systeme werden dann in den Kapiteln 3 und 4 beschrieben.

Teil II befasst sich dann mit der Problematik des Job-Schedulings sowohl auf Parallelrechnern als auch im größeren Kontext von Computational Grids. Dabei werden grundlegende Modellannahmen verdeutlicht und begründet, die für die weitere Entwicklung der Schedulingverfahren essenziell sind. Weiterhin werden dort Softwareentwicklungen und die Datengrundlage für die simulatorische Evaluation von dezentralen Grid-Systemen dargestellt. Der dann folgende Teil III befasst sich mit der detaillierten Evaluation von einfachen statischen, dezentralen Schedulingverfahren und dient als erste simulatorische Studie zum Verhalten von dezentralen Grids im Hinblick auf Stabilität, Störanfälligkeit und möglichen Performancevorteilen. Dazu wird in Kapitel 8 zunächst der Job-Austausch mittels einer zentralen Austauschplattform realisiert und in Kapitel 9 ein erster vollständig dezentraler Ansatz untersucht. Allerdings dienen diese Ergebnisse nur dazu, die Dynamik eines derartigen Grid-Verbunds besser zu verstehen, da die verwendeten Strategien nur unter vereinfachten Einsatzbedingungen nutzbar sind. In Kapitel 10 wird dann abschließend das mögliche Verbesserungspotenzial mittels mehrkriterieller evolutionärer Optimierung identifiziert. Dabei werden einige grundlegende dynamische Eigenschaften eines Grids verdeutlicht.
In Teil IV dieser Arbeit werden dann motiviert durch vorherige Untersuchungen neue Strategien für das dezentrale Scheduling in Computational Grids sowohl architektonisch angepasst als auch mittels Methoden der Computational Intelligence optimiert. Dazu wird zunächst die Architektur eines zweistufigen dezentralen Grid-Schedulers in Kapitel 11 entwickelt. Die Entscheidungsstrategien werden dann als Fuzzy-System realisiert. Die Entscheidungslogik wird mittels eines evolutionären Algorithmus auf die Bedürfnisse unterschiedlicher Grid-Umgebungen hin optimiert. Die spezielle Dynamik und die Eigenschaften eines dezentralen Grid werden dann weiterhin in Kapitel 13 dazu genutzt, mittels coevolutionärer Algorithmen die Grid-Schedulingstrategien weiter anzupassen. Die identifizierten Eigenschaften in Bezug auf Wettstreit und Kooperation werden hier vorteilhaft für die Erzeugung hocheffizienter Schedulingstrategien genutzt. Die Arbeit wird mit einer Zusammenfassung und einem Ausblick auf weitere Verbesserungs- und Entwicklungsmöglichkeiten abgeschlossen.

Teil I

Methoden der Computational Intelligence

2
Computational Intelligence

WENN IN dieser Arbeit Methoden der *Computational Intelligence* (CI) für die Optimierung neuer Architekturen zur Bereitstellung von Rechenressourcen zum Einsatz kommen, dann birgt dies schon eine gewisse Ironie in sich. Denn diese Verfahren sind wiederum selbst erst seit der Verfügbarkeit der ersten Digitalrechner und durch die ständige Verbesserung ihrer Leistungsfähigkeit überhaupt denkbar. Nach wie vor motivieren unter anderem CI-Optimierungsmethoden die Anschaffung immer größerer Rechnerinstallationen und die Vernetzung dieser Systeme bis hin zur Bildung von Computational Grids. Dabei ist das Management und Scheduling dieser Installationen ein derart komplexes Optimierungsproblem, dass wiederum viel Rechenleistung benötigt wird, um eben dieses Problem zu lösen.

Historisch gesehen ist erst durch die Verfügbarkeit der ersten Großrechensysteme die Grundlage für die Realisierung eines lang gehegten Traumes, der Konstruktion von künstlichen und intelligenten Systemen, geschaffen worden. Anfangs galt das Interesse vornehmlich dem Entwurf universeller, lernfähiger und adaptiver Systeme auf der Basis einfacher (numerischer) Repräsentationen. Zahlreiche Ansätze wurden verfolgt, darunter das in den 1960er Jahren von Rosenblatt [143] entwickelte Perzeptron-Modell, das als Vorläufer der neuronalen Netze gilt.

Auch die Grundlagen der evolutionären Algorithmen wurden in jener Zeit gelegt. So entwickelten in den USA unabhängig voneinander Holland [89] die genetischen Algorithmen (GA) als Modell für Adaptations- und Klassifikationsprozesse und Fogel [71] die evolutionäre Programmierung (EP) zur Zeitreihenvorhersage mittels endlicher Automaten. Nahezu zeitgleich dazu entwarfen Rechenberg [140] und Schwefel [151] in Deutschland die Evolutionsstrategien (ES) als Suchheuristiken für die experimentelle Optimierung. Ebenfalls in den 1960er Jahren propagierte der amerikanische Systemtheoretiker Lotfi A. Zadeh den Übergang zu unscharf abgegrenzten Mengen in einer qualitativ neuartigen Modellbildungsstrategie und schuf damit die Grundlagen der Fuzzy-Logik [178].

Der entscheidende Erfolg dieser Ansätze blieb aber zunächst aus, was einerseits auf die zu geringe Leistungsfähigkeit der damals verfügbaren Hardware, andererseits aber auch auf die unzulängliche theoretische Durchdringung der neuen Konzepte zurückzuführen war. So wiesen Minsky und Papert [126] Ende der 1960er Jahre die begrenzte Berechnungsfähigkeit des Perzeptron-Modells nach, worauf das Interesse an konnektionistischen Modellen zunächst stark nachließ.

In der Folge konzentrierte sich die Forschung im Bereich künstlicher Intelligenz auf die symbolische Wissensrepräsentation, genauer gesagt, die Wissensverarbeitung auf der Basis scharfen Wissens im Gegensatz zum unscharfen beziehungsweise unvollständigen Wissen. Zadeh hat hierfür das vielleicht treffendere Begriffspaar „Crisp Computing versus Soft Com-

puting“ geprägt. Die Konstruktion von Expertensystemen und Entwicklungsumgebungen führte in den 1970er und 1980er Jahren zu einigen großen Erfolgen, wodurch die evolutionären Algorithmen fast in Vergessenheit gerieten. Erst Mitte der 1980er Jahre zeichneten sich aber auch die Grenzen der rein symbolischen Wissensverarbeitung ab. Beispielsweise lassen sich für viele (insbesondere technische) Probleme inhärent numerische Repräsentationen angeben, für die sich rein symbolische Lösungsverfahren als ungeeignet erwiesen haben.

Aus diesem Bedarf heraus und mit der Verfügbarkeit immer leistungsfähigerer Rechnersysteme erlebten Teilbereiche der Computational Intelligence eine Renaissance. So führten die Arbeiten von Hopfield [91], Rumelhart und McClelland [146] und der Parallel Distributed Processing (PDP) Research Group zu einer Wiederbelebung der Forschung im Bereich neuronaler Netze. Hier entstanden auch die ersten überzeugenden Einzelerfolge von neuronalen, evolutionären und Fuzzy-basierten Systemen, siehe zum Beipiel Sejnowski und Rosenberg [157] oder Holmblad und Østergaard [90], die als Folge die Forschungsaktivitäten in diesen Bereichen weltweit stark ansteigen ließen.

2.1 Paradigmen der Computational Intelligence

Obwohl sich Computational Intelligence heutzutage zu einem sehr umfangreichen Forschungsfeld mit zahlreichen Anwendungen, starkem theoretischem Hintergrund und einer Vielzahl internationaler Zeitschriften und Konferenzen entwickelt hat, so können doch die wesentlichen Bereiche auf folgende Paradigmen eingegrenzt werden: die künstlichen neuronalen Netze, evolutionäre Algorithmen, Schwarmintelligenz, künstliche Immunsysteme und Fuzzy-Systeme. Eine Zusammenstellung ist dazu in Abbildung 2.1 nach Engelbrecht [53] gegeben. Die blauen Pfeile in dieser Abbildung verdeutlichen dabei, dass CI-Paradigmen häufig durch Zufallsmethoden (zum Beispiel Markov-Modelle oder Kalmanfilter) unterstützt und ergänzt werden.

Alle diese Paradigmen haben dabei ihren Ursprung in der Nachahmung biologischer Systeme: Die neuronalen Netze modellieren biologische Nervensysteme, die evolutionären Algorithmen ahmen den Evolutionsprozess mit seiner genetischen Vererbung nach, die Swarmintelligenz-Modelle basieren auf sozialem Verhalten und der Dynamik von lebendigen Kolonien und Schwärmen, die Immunsysteme sind an den körpereigenen Abwehrsystemen des Menschen orientiert und die Fuzzy-Systeme stützen sich auf Studien über die Interaktion von Organismen mit ihrer Umwelt. Die großen Möglichkeiten der CI-Methoden und des gesamten sogenannten „Soft Computing“ liegen nun aber in der Kombinationsvielfalt der Einzelparadigmen, die es ermöglicht, für spezielle Problemstellungen einzelne Verfahren anzupassen und ergänzend zu nutzen. Diese daraus entstehenden hybriden Optimierungssyssteme sind für eine Vielzahl praktischer Probleme geeignete Lösungsverfahren. In Abbildung 2.1 werden mögliche Methodenkombinationen durch schwarze beziehungsweise grüne Verknüpfungspfeile dargestellt. So können zum Beispiel auch in dieser Arbeit die fehlenden Lernfähigkeiten eines Fuzzy-Systems durch die Nutzung eines evolutionären Algorithmus kompensiert werden. Ein anderes Beispiel ist die Kombination von Suchverfahren, wie Schwarmverhalten, für die grobe Orientierung und Zufallsmethoden in Form einer lokalen Suche für die detaillierte Lösungsfindung, siehe zum Beispiel Grimme u. a. [19]. Die vorliegende Arbeit beschränkt sich auf die evolutionären Algorithmen und die Fuzzy-Systeme aus den CI-Methoden, da diese sich für die behandelte Problemstellung ideal eignen.

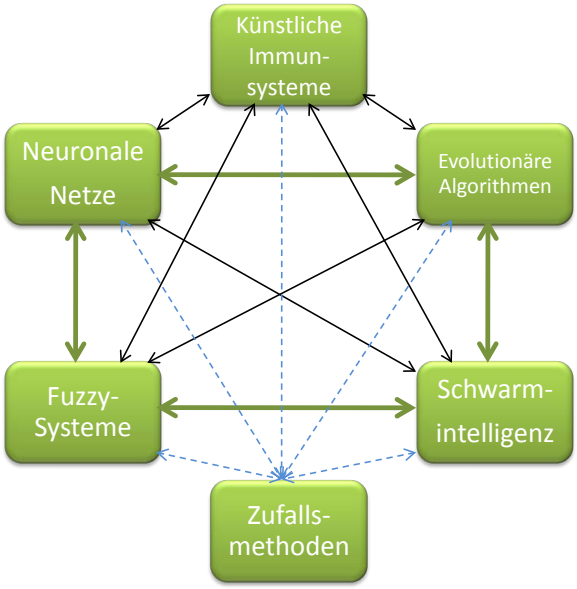


Abbildung 2.1: Paradigmen der Computational Intelligence nach Engelbrecht [53].

2.2 Evolutionäre Algorithmen

Die Entwicklung der evolutionären Algorithmen geht auf die ersten Untersuchungen auf dem Feld der genetischen Algorithmen von Fraser [78] und Bremermann [36] in den 1950er bis 1960er Jahren zurück. Aber erst durch die Arbeiten von Holland [89] wurde das Konzept der Darwin'schen Evolutionstheorie algorithmisch auf die Lösung von Optimierungsproblemen übertragen. In den 1960er Jahren inspirierte diese grundlegende Arbeit Darwins sowohl Rechenberg [140] als auch zeitgleich Schwefel [151] zur Entwicklung der sogenannten Evolutionsstrategien. Im Zuge dessen wurde auch das Konzept für eine eigenständige Anpassung der Mutationsschrittweite theoretisch untermauert.
Ein weiteres sehr wichtiges Forschungsfeld wurde ebenfalls Ende der 1960er Jahre in der Anwendung von evolutionären Algorithmen auf mehrkriterielle Problemstellungen erstmals betrachtet. Der erste Hinweis zur Möglichkeit der Verwendung von evolutionären Algorithmen für die Lösung eines multikriteriellen Problems erscheint in der Doktorarbeit von Osyczka [133] aus dem Jahre 1967, in der jedoch kein wirklicher mehrkriterieller evolutionärer Algorithmus (MOEA) entwickelt wird, sondern vielmehr das mehrkriterielle Problem auf ein einkriterielles aggregiert und dann mit einem genetischen Algorithmus gelöst wird. Die ursprüngliche Idee, einen genetischen Algorithmus zur Lösung von mehrkriteriellen Problemen einzusetzen, reicht bis zu den nur selten erwähnten Versuchen von Ito u. a. [95] aus dem Jahre 1983 zurück. Heute gilt hingegen allgemein Schaffer [148] als der erste, der in der Mitte der 1980er Jahre einen echten MOEA entwickelte. Schaffer's Ansatz, genannt „Vector Evaluated Genetic Algorithm (VEGA)", besteht aus einem einfachen genetischen Algorithmus mit einem modifizierten Auswahlmechanismus, der sich aber bei genauerer Prüfung mit einer Reihe von Nachteilen bei der Selektion konfrontiert sah. Dies motivierte die direkte Einbeziehung des Konzepts der Pareto-Dominanz in einem evolutionären Algorithmus und wurde von Goldberg [81] in seinem Buch über genetische Algorithmen erstmals formuliert. Obwohl dies in Abschnitt 3.3 noch genauer betrachtet wird, sei hier schon bemerkt,

dass eine Vielzahl der heute eingesetzten MOEAs immer noch auf diesen frühen Ideen basiert. Dabei zeigt die Menge der aktuellen Veröffentlichungen, dass gerade auf dem Gebiet der mehrkriteriellen evolutionären Optimierung sehr aktiv Forschung betrieben wird.
Neben dem bis heute andauernden Ausbau der Methoden evolutionärer Optimierung im Mehrkriteriellen, hat sich im Einkriteriellen noch eine Weiterentwicklung als besonders praktisch für viele Anwendungen erwiesen: Es handelt sich um die sogenannten co-evolutionären Algorithmen, bei denen mehrere Spezies in einer gemeinsamen Umwelt miteinander im Anpassungswettstreit stehen. Diese Idee wurde bereits im Jahre 1967 entwickelt, motiviert durch spieltheoretische Untersuchungen von Reed u. a. [142]. Maßgeblich wurde dieses Gebiet von Potter und De Jong [138] und Paredis [134] weiter untersucht. Dabei führten sie erfolgreich Optimierungen sowohl mit kooperativen als auch mit kompetitiv co-evolutionären Algorithmen durch. Seitdem Axelrod [29] Mitte der 1980er Jahre zeigte, wie Kooperation unter egoistischen Individuen selbst dann zustande kommen kann, wenn diese nicht durch eine äußere Instanz, durch Moral oder durch Gesetze erzwungen wird, kann Co-Evolution auch immer im Kontext von sozialem Verhalten, Spieltheorie und dem Phänomen der Emergenz gesehen werden.

2.3 Fuzzy-Systeme

In Bezug auf Fuzzy-Logik und die damit verbundenen Fuzzy-Systeme können philosophische Grundgedanken bis in die Zeit des Buddha (563 v. Chr.) und des Buddhismus zurückverfolgt werden, da auch dort besonders im moralischen Zusammenhang schon Handlungen in „Schattierungen von Grau“ bewertet werden. Allerdings wird in der westlichen Welt heute eher die im Jahre 1920 von Jan Łukasiewicz (1878–1956) veröffentlichte Arbeit als Ausgangspunkt für die Entwicklung der Fuzzy-Logik angenommen, da dort zum ersten Mal die klassische aristotelische zweiwertige Logik auf zunächst eine dreiwertige Logik ($Ł_3$) und dann auf ein logisches Kalkül mit einer beliebigen Anzahl von Werten erweitert wurde. Der Quanten-Philosoph Max Black (1909–1988) führte quasi Fuzzy-Mengen ein, bei denen er Elementen einen „Grad der Zugehörigkeit“ zu Mengen zuordnete.
Schließlich war es aber Lotfi A. Zadeh [178], der 1965 auf dem Gebiet der Fuzzy-Logik und der Fuzzy-Mengen die entscheidenden Beiträge leistete. Von diesem Zeitpunkt bis zu Beginn der 1980er Jahre waren Fuzzy-Systeme ein dominierendes Forschungsgebiet, besonders der Regelungstechnik, in dem Forscher wie Mamdani, Sugeno, Takagi und Bezdek entscheidende Beiträge leisteten. Dann begann in den 1980er Jahren das dunkle Zeitalter für die Fuzzy-Systeme, in dem sie fast vollständig aus dem Fokus der Forschung verschwanden, um dann erst in den späten 1980er Jahren von japanischen Forschern wieder forschungstechnisch weiterentwickelt zu werden. Heute sind Fuzzy-Systeme wieder ein sehr aktives Forschungsgebiet mit vielen erfolgreichen Anwendungen, vor allem bei der Steuer- und Regelungstechnik, aber auch in Bereichen der Robotik, Bilderkennung und der Datenanalyse.
Nach dieser kurzen historischen Einordnung werden nun die einzelnen in dieser Arbeit genutzten CI-Methoden genauer beschrieben. Dabei wird zunächst in Kapitel 3 auf die evolutionären Algorithmen und dort insbesondere auf die Evolutionsstrategien eingegangen. Anschließend werden in Kapitel 4 Fuzzy-Systeme und deren Kopplung mit evolutionären Algorithmen als Lernverfahren erläutert.

3
Evolutionäre Algorithmen

EVOLUTIONÄRE ALGORITHMEN sind randomisierte Suchheuristiken, inspiriert durch die natürliche Evolution der Arten. Die Grundidee liegt in der Simulation der Entwicklung von möglichen Kandidatenlösungen, in Analogie zur Natur auch *Individuen* genannt, für ein gegebenes Optimierungsproblem. Ein typischer evolutionärer Algorithmus enthält dazu eine vorgegebene Anzahl von Kandidatenlösungen, die als *Population* bezeichnet wird. Die Qualität der Kandidatenlösung in der Population ist dabei durch den Funktionswert der zu optimierenden Funktion bestimmt und wird auch als *Fitness* der Individuen bezeichnet.

Der Algorithmus versucht nun iterativ, die gegenwärtige Population in ihrer Fitness zu verbessern. Dazu werden in jeder Generation Eltern aus der aktuellen Population selektiert (üblicherweise unter Berücksichtigung der jeweiligen Fitnesswerte) und aus diesen werden durch die Anwendung von unterschiedlichen Variationsoperatoren wie Mutation oder Rekombination neue Kandidatenlösungen, die sogenannten Nachkommen, erzeugt. Da nur gute Individuen durch den Selektionsprozess in die nächste Generation übernommen werden, ist davon auszugehen, dass durch die wiederholte Anwendung dieser Prozedur immer besser an ihre Umwelt angepasste Individuen in der Population entstehen und somit die Lösung des Optimierungsproblems immer besser approximiert wird. Dies entspricht der Imitation des von Darwin formulierten Prinzips des „Überleben des besser Angepassten“.

Evolutionäre Algorithmen kommen vorzugsweise dort zum Einsatz, wo aufgrund fehlenden Wissens über Details der Problemstellung klassische Optimierungsverfahren nicht anwendbar sind. Dies kann zum Beispiel der Fall sein, wenn keine mathematische Beschreibung der Fitnessfunktion angegeben werden kann und diese nur experimentell oder simulatorisch auswertbar ist. Weiterhin sind diese Verfahren so allgemein verwendbar, dass sie gern genutzt werden, wenn aufgrund von Ressourcenmangel keine problemspezifische Lösung erarbeitet werden kann. Zusammengefasst kann ein evolutionärer Algorithmus als sogenanntes „Black Box“-Optimierungsverfahren verstanden werden, bei dem im Extremfall kein Wissen über die Struktur und Eigenschaften des zu optimierenden Problems vorhanden sein muss. Für derartige Probleme sind evolutionäre Algorithmen oft die einzig einsetzbaren Lösungsverfahren.

In diesem Kapitel werden nun zunächst die klassischen Evolutionsstrategien besprochen. Am Anschluss daran wird deren Erweiterung auf das co-evolutionäre Konzept unter Berücksichtigung mehrerer Spezies verdeutlicht. Abschließend werden kurz die hier verwendeten Algorithmen zur mehrkriteriellen Berechnung und die damit verbundenen Problemstellungen betrachtet. Es sei aber hier schon darauf hingewiesen, dass in der in dieser Arbeit vorgenommenen Darstellung nur die für das weitere Verständnis unverzichtbaren Grundprinzipien und ausschließlich die in der Arbeit verwendeten Konzepte dieser umfangrei-

chen Gebiete berücksichtigt werden können. In der aktuellen Literatur seit 1970 sind umfassende Arbeiten zu allen Teilbereichen der evolutionären Algorithmen verfügbar, weshalb für weitergehende Informationen zu einzelnen Aspekten im Text auf entsprechende Monografien verwiesen wird.

3.1 Evolutionsstrategien

Ausgangspunkt ist eine Population aus μ Individuen aus dem Individuenraum $\mathbb{I}$. Im Folgenden wird eine Multimenge[1] von μ Individuen aus dem Individuenraum $\mathbb{I}$ als $\mathcal{M}_\mu(\mathbb{I})$ bezeichnet. Evolutionsstrategien arbeiten nun auf Basis einer Population $P_{t,\mu} \in \mathcal{M}_\mu(\mathbb{I})$ von μ Elternindividuen in jeder Generation t. Die schon oben erwähnten Variationsoperatoren werden dann dazu genutzt, in jeder Generation λ Kinder aus den μ Eltern zu erzeugen. Die Parameter μ und λ sowie die später eingeführte Lebenszeit κ oder die Anzahl ρ der für die Rekombination zu berücksichtigenden Eltern sind sogenannte *exogene* Parameter, die einmal gewählt werden und dann während des gesamten evolutionären Prozesses unverändert bleiben.
Individuen sollen die Parametereinstellung des zu optimierenden Systems repräsentieren. Dazu ist es notwendig, die Problemstellung in geeigneter Form zu kodieren, damit sie in ein Individuum abgebildet werden kann. Formal wird ein Individuum $\vec{a}_k \in \mathbb{I}$, siehe Gleichung 3.1, durch einen Vektor von sogenannten *Objektparametern* $\vec{o}_k = (o_1, o_2, \ldots, o_u)$ und eine Menge von *Strategieparametern* $\vec{s}_k = (s_1, s_2, \ldots, s_v)$ definiert.

$$\vec{a}_k := (\vec{o}_k, \vec{s}_k) \tag{3.1}$$

Die Strategieparameter $\vec{s}_k$ sind sogenannte *endogene* Parameter, da sie Einfluss auf die eigentlichen Objektparameter ausüben. Diese Parameter sind nicht statisch, sondern können sich während des evolutionären Prozesses verändern, sodass durch sie zum Beispiel eine Selbstadaption der Mutationsschrittweite realisiert werden kann, siehe Abschnitt 3.1.3.
Der Algorithmus startet mit der Initialisierung der ersten Population $P_{0,\mu}$ mit μ Individuen $\vec{a}_k \in \mathbb{I}$. Diese Initialisierung wird typischerweise zufällig durchgeführt, solange kein genaueres Problemwissen verfügbar ist. Dann wird die evolutionäre Schleife so lange ausgeführt, bis ein vorher definiertes Abbruchkriterium erfüllt ist. Dieses kann zum Beispiel eine maximale Zahl von Funktionsauswertungen oder Generationen sowie das Unterschreiten eines bestimmten Schwellwertes der erreichten Lösungsqualität sein. Die aktuelle Generation $P_{t,\mu}$ wird dazu genutzt, die neue Nachkommengeneration $Q_{t,\lambda}$ zu bilden. Dazu wird zunächst eine Vereinigungsmenge $\mathcal{P}_\rho(\mathbb{I})$ von ρ Eltern gebildet, die dann als Grundlage für die Rekombination dient. Wenn $\rho = 1$ gilt, wird überhaupt keine Rekombination ausgeführt. Die Objektparameter einer Population werden dabei erst nach der Rekombination der Strategieparameter ebenfalls rekombiniert.
Im Anschluss werden dann die Strategieparameter mutiert und erst nach diesen wird die eigentliche Mutation der Objektparameter durchgeführt. Dies geschieht dann auf Basis der zuvor schon mutierten Strategieparameter. Nun wird jedem Individuum ein Fitnesswert auf Basis einer Auswertung der Zielfunktion für die jeweils repräsentierten Parametereinstellungen zugeordnet.

[1] Multimengen unterscheiden sich von gewöhnlichen Mengen dadurch, dass Elemente einer Multimenge mehrfach vorkommen können. In dieser Arbeit werden Multimengen durch kalligrafische Buchstaben wie $\mathcal{M}$ oder $\mathcal{P}$ gekennzeichnet.

Der nächste Schritt innerhalb der Evolutionsschleife ist die Selektion der nächsten Generation aus der Nachkommenpopulation. Die Evolutionsstrategien können daher in zwei Klassen unterteilt werden, die sich durch ihr Selektionsverhalten unterscheiden:

- **Komma-Strategie** (μ, λ)
  Bei der *Komma*-Strategie wir die nächste Elterngeneration $P_{(t+1),\mu}$ ausschließlich aus der Nachkommengeneration $Q_{t,\lambda}$ selektiert.

- **Plus-Strategie** $(\mu + \lambda)$
  Im Gegensatz dazu werden bei der *Plus*-Strategie die besten Individuen aus der Vereinigungsmenge von Eltern und Nachkommen $(P_{t,\mu} \cup Q_{t,\lambda})$ selektiert.

Bei der Plus-Strategie wird immer die bisher beste gefundene Lösung in die nächste Generation übernommen, weshalb es bei diesem Selektionsverfahren zu keiner Verschlechterung während des Evolutionsprozesses kommen kann. Deshalb wird ein derartiger Algorithmus auch *elitär* genannt. Es besteht dabei allerdings das bekannte Problem der vorzeitigen Konvergenz, da die permanente Gefahr besteht, in einem lokalen Optimum stecken zu bleiben. Dies kann vor allem dann geschehen, wenn einige wenige Individuen mit guten Fitnesswerten die gesamte Population dominieren. Sie würden dann in folgenden Generationen immer wieder selektiert und somit allein verantwortlich für die Erzeugung der Nachkommenschaft sein. Dies führt oftmals dazu, dass keine Verbesserungen mehr erzielt werden. Um diesem Phänomen zu begegnen, kann eine maximale Lebenszeit κ eingeführt werden, nach deren Überschreitung Individuen aus dem evolutionären Prozess ausgeschlossen werden. Durch die Wahl von $\kappa = 1$ ist diese allgemeine Strategie identisch mit der (μ, λ)-Evolutionsstrategie, während es sich bei $\kappa = \infty$ um eine reine $(\mu + \lambda)$-Evolutionsstrategie handelt. Wenn nun alle diese eingeführten exogenen Parameter kombiniert werden, so entsteht die sogenannte $(\mu, \kappa, \lambda, \rho)$-Evolutionsstrategie nach Schwefel und Rudolph [153]. Die prinzipielle Struktur ist nochmal in Algorithmus 1 formal dargestellt.
Für die Selektions- und Variationsoperatoren ist seit der Erfindung der evolutionären Algorithmen eine Fülle von Realisierungen entworfen und untersucht worden. Ist detaillierteres Wissen über die zu optimierende Problemstellung vorhanden, kann durch die Anpassung speziell auf das Problem zugeschnittener Operatoren die Effizienz von Algorithmen deutlich gesteigert werden. In dieser Arbeit ist derartiges Problemwissen aber nur für die in Kapitel 10 behandelte mehrkriterielle Problemstellung vorhanden, weshalb dort auch entsprechende Operatoren angepasst werden können. Für die in dieser Arbeit betrachteten einkriteriellen Probleme ist dies leider nicht der Fall, weshalb dann an entsprechenden Stellen auf die im Folgenden kurz beschriebenen Standardoperatoren zurückgegriffen wird.

3.1.1 Selektionsoperator

Die Selektion reduziert die Anzahl der Individuen, die eine neue Population in der nächsten Generation bilden, indem eine Auswahl von μ Individuen aus der gesamten Nachkommenschaft $Q_{t,\lambda}$ der Größe λ getroffen wird. Dies geschieht bei Evolutionsstrategien üblicherweise deterministisch und unter Berücksichtigung der jeweiligen Fitness. Durch die andauernde Selektion bewegt sich die gesamte Population allmählich auf Regionen in der Nähe von optimalen Lösungen zu. Das Verhältnis von Eltern zu Nachkommen $\mu : \lambda$ wird dabei als *Selektionsdruck* bezeichnet und soll nach Schwefel [152] möglichst im Verhältnis $1 : 7$ stehen.

Algorithmus 1: $(\mu, \kappa, \lambda, \rho)$-Evolutionsstrategie

```
1  P_{0,μ} ← Initialisierung mit μ Individuen, P_{0,μ} ∈ M_μ(I);
2  P_{0,μ} ← Auswertung;
3  t ← 0;
4  while Abbruchkriterium nicht erfüllt do
5      for k = 0 to λ do
6          P_{ρ,k}(I) ← vereinige(P_{t,μ}, ρ);
7          s⃗_k ← Rekombination der Strategieparameter (P_{ρ,k}(I));
8          o⃗_k ← Rekombination der Objektparameter (P_{ρ,k}(I));
9          s̃⃗_k ← Mutation der Strategieparameter (s⃗_k);
10         õ⃗_k ← Mutation der Objektparameter (s̃⃗_k, o⃗_k);
11         F̃_k ← F(õ⃗_k);
12     end
13     Q_{t,λ} ← {(õ⃗_k, s̃⃗_k), k = 1, ..., λ};
       // Q_{t,λ} ∈ M_λ(I)
14     Q_{t,λ} ← Auswertung;
15     if κ = 1 then
16         P_{(t+1),μ} ← Selektion (Q_{t,λ}, μ);
17     if κ = ∞ then
18         P_{(t+1),μ} ← Selektion (Q_{t,λ} ∪ P_{t,μ}, μ);
19     else
20         P_{(t+1),μ} ← Selektion (Q_{t,λ} ∪ P_{t,μ}, μ, κ);
           // P_{(t+1),μ} Enthält nur noch Individuen mit einer
           // Lebenszeit von ≤ κ Generationen
21     end
22     t ← t + 1;
23 end
```

Übliche Verfahren sind zum Beispiel die *binäre Turnierselektion* oder die *Fitness-proportionale Selektion*. Bei der binären Turnierselektion werden zwei Individuen aus der aktuellen Population ausgewählt und das Individuum mit der besseren Fitness wird in die nächste Generation übernommen. Dann wird dieser Vorgang wiederholt, bis eine ausreichende Zahl von Individuen ausgewählt wurde. Die verschiedenen Turnierverfahren unterscheiden sich nur in der Art, wie sie die nächsten zwei Individuen für ein Turnier auswählen, siehe Deb [48].

Die Fitness-proportionale Selektion basiert auf dem Gedanken, dass Individuen mit einer hohen Fitness auch mehr Nachkommen in der nächsten Generation haben sollen als diejenigen mit niedriger Fitness. Dazu steigt die Wahrscheinlichkeit für ein Individuum, übernommen zu werden, proportional mit seiner Fitness im Vergleich zur durchschnittlichen Fitness der gesamten Population. Als Umsetzung wird hier häufig die sogenannte *Rouletterad-Selektion* genutzt, bei der auf einem imaginären Rad Partitionen für jedes Individuum eingeteilt sind. Diese Partitionen entsprechen ihrer Größe nach den unterschiedlichen Fitnesswerten der Individuen. Die μ Individuen werden dann durch ein μ-faches „Roulettespiel“ selektiert. Nachteil einer derartig stark Fitnesswert-betonten Selektion ist die Gefahr eines

Diversitätsverlusts in der Population, weshalb in dieser Arbeit nur die Turnierselektion verwendet wird.

3.1.2 Rekombinationsoperatoren

Bei der Rekombination werden die Gene zweier Individuen vermischt. Aus den vermischten Genen wird ein neues Individuum erzeugt, was dem natürlichen Fortpflanzungprozess von Individuen entspricht. Bei Evolutionsstrategien ist die Anzahl der Individuen für die Rekombination jedoch nicht auf ein Elternpaar beschränkt, sondern kann auch unter ρ Eltern, siehe Algorithmus 1, durchgeführt werden. Für die Rekombination werden im Allgemeinen die zwei folgenden Verfahren verwendet:

Diskrete Rekombination: Bei der diskreten Rekombination werden zwei Eltern $\vec{a}_1$ und $\vec{a}_2$ mit der gleichen Wahrscheinlichkeit aus der Elternpopulation μ ausgewählt. Im Anschluss daran wird für jedes Gen $a'(i)$ des entstehenden Nachkommens $\vec{a}'$ wieder mit gleicher Wahrscheinlichkeit das Gen eines zuvor ausgewählten Elternteils übernommen. Es gilt demnach entweder $a'(i) = a_1(i)$ oder $a'(i) = a_2(i)$. Dieses Verfahren ist entsprechend einfach auch auf ρ Eltern übertragbar, wobei dann jedes Gen gleich verteilt aus einer größeren Elternschaft gewählt werden muss.

Intermediäre Rekombination: Hier ist die Auswahl der Eltern identisch mit dem Prozess der diskreten Rekombination, aber für die Bestimmung eines Gens der Nachkommenschaft a'_i werden die Gene der Eltern miteinander verrechnet. Es gilt also

$$a'_i = \frac{a_1 + a_2}{2}$$

oder allgemein

$$a'_i = \frac{1}{\rho} \sum_{k=1}^{\rho} a_k(i).$$

Hierbei handelt es sich also um eine Schwerpunktbildung der beteiligten Vektoren. Wenn $\rho = \mu$ gesetzt wird, so ist die gesamte Population an der Rekombination beteiligt und die Nachkommen werden um den Schwerpunkt der Population herum erzeugt.

Üblicherweise wird nach Schwefel [152] für die Objektparameter eine diskrete und für die Strategieparameter eine intermediäre Rekombination verwendet.

3.1.3 Mutationsoperator

Mutation ist in der Natur auf Fehler bei der Vererbung der genetischen Informationen, aber auch auf äußere Einwirkungen zurückzuführen. Durch Zufall können so Individuen entstehen, die noch besser an ihre Umgebung angepasst sind, da sie neue durch Mutation entstandene vorteilhafte Merkmale besitzen. Zur Nachahmung der Mutation in Evolutionsstrategien dient ein Mutationsvektor:

$$\vec{r}_m = \begin{bmatrix} \mathcal{N}_1(0, \sigma_m) \\ \vdots \\ \mathcal{N}_u(0, \sigma_m) \end{bmatrix}$$

Dieser wird einmal pro Generation neu bestimmt und enthält durch $\mathcal{N}_u(0, \sigma_m)$ für jedes Gen $a(i)$ eine normal verteilte reellwertige Zufallszahl. Die Standardabweichung σ_m der Normalverteilung wird in diesem Zusammenhang auch Mutationsschrittweite genannt. Sie kann entweder für alle Merkmale gleich gewählt werden (isotrope Mutation) oder unterschiedlich für jeden Parameter (anisotrope Mutation). Für ein Individuum $\vec{a}'$ wird also die Mutation durch Addition, siehe Gleichung 3.2, realisiert.

$$\tilde{\vec{a}}' = \vec{a} + \vec{r}_m \tag{3.2}$$

Dabei ist wichtig zu beachten, dass für die Veränderung der Strategieparameter ein anderes Verfahren eingesetzt werden muss, das auf rein multiplikativer Mutation beruht, da sonst dort auch negative Werte entstehen könnten. Diese Problematik wird im Folgenden verdeutlicht.

Selbstanpassung der Mutation

Die Einführung und Nutzung zusätzlicher Strategieparmeter ist einer der großen Unterschiede von Evolutionsstrategien im Vergleich zu genetischen Algorithmen. Es ist dadurch nämlich möglich, die oben beschriebene Mutationsschrittweite σ_m als Strategieparmeter mit in die Individuen zu kodieren und sie sich damit selbst im evolutionären Prozess anpassen zu lassen, siehe auch Beyer [32]. Dies basiert auf der Annahme, dass bei der Selektion von gut angepassten Elternindividuen auch gute Schrittweiten mit selektiert werden und diese dadurch ebenfalls evolutionär optimiert werden. Damit dies funktioniert, müssen die Schrittweiten als Strategieparameter $(\vec{s}_k)$ *vor* den Objektparametern mutiert werden, damit für die dann folgende Mutation der Objektparameter $(\vec{o}_k)$ auch die bereits mutierten Strategieparameter $(\tilde{\vec{s}}_k)$ verwendet werden, siehe auch Algorithmus 1. Diese einzelnen Schrittweiten werden, ähnlich wie die Objektparameter jedoch multiplikativ wie in Gleichung 3.3 beschrieben, mutiert.

$$\vec{\sigma}'_m = \vec{\sigma}_m \cdot \xi \tag{3.3}$$

Dabei stellt ξ einen Mutationsoperator dar, für dessen Bildung zahlreiche Konzepte seit Einführung der Evolutionsstrategien existieren. In dieser Arbeit wird die ebenso einfache wie lang erprobte *Zwei-Punkt-Regel* nach Rechenberg [141] verwendet, siehe Gleichung 3.4.

$$\xi = \begin{cases} 1+\beta, & \text{wenn } z \leq 0,5 \\ \dfrac{1}{1+\beta}, & \text{wenn } z > 0,5 \end{cases} \tag{3.4}$$

Hierbei ist z eine gleich verteilte Zufallsvariable auf dem Intervall $z \in]0;1]$ und β ist ein exogener Parameter, der vorzugsweise aus dem Intervall $\beta \in [0,1;0,6]$ zu wählen ist. Nach Rechenberg [141] ist für den Schrittweitenänderungsfaktor ein Wert von $\beta = 1,3$ empfehlenswert. Jedoch haben genauere Untersuchungen von Scheel [149] gezeigt, dass β abhängig von der Anzahl der Objektparameter und der Anzahl der Nachkommen λ ist. Nach Kost [106] empfiehlt es sich für eine kleine Anzahl von Merkmalen $(u < 100)$, den Parameter β an der oberen Definitionsgrenze zu wählen.

3.2 Co-evolutionäre Algorithmen

Co-evolutionäre Algorithmen (CAs) sind aus der Beobachtung motiviert, dass in der Natur aus dem dynamischen Wechselspiel von unterschiedlichen Spezies aus ein gegenseitiger Entwicklungsdruck resultiert. Dabei interagieren Spezies strategisch miteinander und unterziehen sich in ihrer Entwicklung einer gegenseitigen Anpassung. In der Natur wird beobachtet, dass sich nur diejenigen Spezies weiter verbreiten können, die es schaffen, sich gegenüber anderen Arten zu etablieren. Dies muss nicht zwangsläufig zu einer gegenseitigen Auslöschung führen, sondern kann ganz unterschiedliche Ausprägungen, wie Symbiosen, Kooperationen oder auch Nischenbildung und Spezialisierung, zur Folge haben. Technisch gesehen ist der Ansatz der Modularisierung in einem CA aus der Feststellung motiviert, dass viele reale Probleme zu schwer zu optimieren sind, wenn sie als monolithische Instanz betrachtet werden. Vielmehr ist es effektiver, sie in eine Menge von Unterproblemen mit reduzierter Komplexität zu zerlegen, die dann leichter einzeln optimiert werden können, siehe auch Jansen und Wiegand [98].
In CAs wird also nicht mehr nur eine Spezies in einer Population betrachtet, sondern mehrere Spezies gleichzeitig in sich getrennt voneinander entwickelnden Populationen.[2] Innerhalb einer Population findet dann zum Beispiel ein konventionelles evolutionäres Entwicklungsschema, wie es bereits im vorigen Abschnitt beschrieben wurde, statt. Daher unterscheiden sich CAs von konventionellen evolutionären Algorithmen hauptsächlich in ihrem Bewertungsprozess: Ein Individuum kann nur bewertet werden, wenn es mit Individuen anderer Spezies in einer gemeinsamen Umwelt interagiert. Das zu lösende Optimierungsproblem ist demnach also so in unterschiedliche Spezies aufgeteilt, dass nur deren Zusammenwirken ein funktionstüchtiges System ergibt. Übliche Interaktionen sind zum Beispiel die Gegenüberstellung eines Individuums mit allen Individuen der übrigen Spezies in Form eines „Jeder gegen jeden"-Prinzips oder auch die Zusammenstellung rein zufälliger Paarungen. Allgemein lassen sich aber nach Paredis [135] zwei Typen von Interaktionsmodellen unterscheiden: Die sogenannte *kooperative Co-Evolution*, beschrieben von Potter und De Jong [139], löst ein komplexes Problem durch die gleichzeitige Evolution von Teilproblemen. Jedes Teilproblem wird durch eine eigene Spezies repräsentiert. Jede Spezies versucht *gemeinsam* mit anderen Spezies, sich in ihrem jeweiligen Teilgebiet zu verbessern und eine größere Fitness dadurch zu erhalten, dass das Gesamtproblem besser gelöst wird. Jedem Individuum wird dann ein *einzelner* Fitnesswert, der bei der Auswertung des *Gesamtsystems* erreicht wurde, zugeordnet. Derartige Interaktionen (manchmal auch Symbiosen genannt) beschreiben eine Situation, in der mehrere Spezies nebeneinander existieren und jede von dem jeweiligen Evolutionsfortschritt der anderen profitiert.
Im Gegensatz zu diesem symbiotischen Konzept wird bei *kompetitiven co-evolutionären Algorithmen* die Fitness eines Individuums durch dessen Wettstreit mit Individuen anderer Spezies bestimmt. Der evolutionäre Fortschritt einer Spezies erhöht dann den Anpassungsdruck bei Individuen anderer Spezies. Dazu muss das Problem in solch einer Form aufteilbar sein, dass jedem Individuum ein einzelner Fitnesswert zugeordnet werden kann. Ein global bezogener Fitnesswert ist in diesem Szenario zwar berechenbar, aber für den evolutionärem Prozess von keiner Bedeutung. Vielmehr spiegelt dieses Konzept einen Interaktionsprozess wider, bei dem jede Spezies nur nach ihrem eigenen Vorteil strebt und deshalb nur nach

[2] Auch in konventionellen evolutionären Algorithmen können mehrere Populationen berücksichtigt werden. Diese sind dann aber nicht Teil des gemeinsamen Bewertungsprozesses, sondern zielen primär auf Nischenbildung und sogenannte Inselmodelle zur Erhaltung von Teillösungen ab.

individueller Fitnessverbesserung strebt. Eine Verbesserung des Gesamtsystems ergibt sich dann allerdings als Konsequenz dieser Entwicklung, ist aber ihr deren explizit berücksichtigtes Ziel. Durch den gegenseitigen Konkurrenzkampf resultiert häufig ein verbesserter Fitnesswert bei einer Spezies in einer Verschlechterung der Fitness bei einer anderen Spezies. Idealerweise ersetzen sich während der Evolution gegenseitig konkurrierende Lösungen jeweils derartig, dass sich daraus ständig bessere Lösungen ergeben. Dieses Konzept wurde auch häufig, wie zum Beispiel von Rosin und Belew [144], als eine Form des „Wettrüstens“ beschrieben, in dem sich Populationen gegenseitig zu verbesserten Lösungen steigern.

Weiterhin wird in diesem Zusammenhang immer wieder der starke Bezug von Co-Evolution zu einem Räuber-Beute-Verhältnis propagiert. So werden zum Beispiel in der Arbeit von Grimme u. a. [13] komplexe (und sogar mehrkriterielle) Optimierungsprobleme durch ein wechselseitiges Zusammenspiel von Räuber- und Beuteindividuen erreicht. Dabei verfolgen unterschiedliche Räuber rein egoistische Einzelziele und die Lösung des mehrkriteriellen Problems ergibt sich als Resultat der wechselseitigen Einflüsse der Räuber auf die Beute.

Obwohl sich sowohl in der Optimierung als auch im Bereich des Managements die Nutzung emergenten Verhaltens durch gegenseitiges Kooperations- und/oder Konkurrenzverhalten als Alternative zu klassischen deliberaten Problemlösungsstrategien immer größeren Interesses erfreut, so bleibt Co-Evolution doch in der Anwendungsforschung bisher eine Randerscheinung. Hillis [88] präsentiert eine frühe Anwendung aus dem Bereich der Sortiernetzwerke, indem dort die Topologie durch Wirte und Parasiten innerhalb eines genetischen Algorithmus modelliert wird. Dabei repräsentieren die Wirte Sortierstrategien, während die Parasiten Testfälle in Form von zu sortierenden Zahlenreihen darstellen. Weiterhin untersuchte Olsson [132] ein auf kompetitiven Spezies beruhendes Modell, das sich als sehr effektiv für die Erzeugung und Erhaltung von genetischer Diversität herausstellte. Diese Konzepte führen dort zu besseren Endlösungen als übliche nicht co-evolutionäre Verfahren.

3.3 Mehrkriterielle evolutionäre Optimierung

Bei der Formulierung von Optimierungsproblemen unterschiedlichster Art ist es häufig unmöglich, nur einem einzelnen Bewertungsmaßstab für die Güte der erreichten Lösung den Vorzug zu geben. Dabei verhalten sich die verschiedenen Gütekriterien bei mehrkriteriellen Problemen widersprüchlich zueinander. Beispiele aus dem in dieser Arbeit betrachteten Bereich des Job-Schedulings für konfliktäre Ziele sind die Auslastung der Maschinen und die Durchlaufzeit der einzelnen Jobs. Für die Entscheidungsfindung ergibt sich daraus das Problem, einem der Kriterien den Vorzug vor den anderen geben zu müssen, oder *a priori* einen Kompromiss zwischen all den zu betrachtenden Kriterien finden zu müssen, obwohl die relative Wichtigkeit der Einzelkriterien vielleicht noch gar nicht eingeschätzt werden kann.

Durch mehrkriterielle Optimierung ist es nun möglich, alle diese Ziele gleichzeitig zu erfassen und einen Überblick über die wechselseitigen Abhängigkeiten und strukturellen Eigenschaften der Lösungsmöglichkeiten zu gewinnen. Dabei werden durch entsprechende Algorithmen zunächst möglichst sowohl alle extremen Entscheidungen als auch alle Kompromisse gefunden. Anschließend kann auf Basis der erzeugten Lösungsmenge ein Entscheider unter weiterer Berücksichtigung seiner Erfahrung und eventuell vorhandener zusätzlicher Bedingungen einzelne Lösungen auswählen. In dieser Arbeit wird mit der Erzeugung aller möglichen Kompromisslösungen primär versucht, die Dynamik und die gegenseitigen

Abhängigkeiten in einem Grid zu identifizieren. Die Abhängigkeiten dienen dann als Erfahrungsgrundlage für den weiteren Entwurf von Schedulingalgorithmen.

Ein mehrkriterielles Optimierungsproblem liegt also immer dann vor, wenn mehr als eine Zielfunktion gleichzeitig betrachtet wird und keine einzelne Lösung, sondern eine Menge von möglichen Kompromisslösungen berechnet werden soll. Um den Begriff der Menge der Kompromisslösungen formell genauer zu fassen, wird der auf Vilfredo Pareto um 1896 zurückgehende Begriff des *Pareto-Optimums* verwendet. Dieser besagt, dass es für eine Pareto-optimale Lösung nicht mehr möglich ist, sie in einem Zielkriterium zu verbessern, ohne sie dadurch zugleich in einem anderen Zielkriterium zu verschlechtern. Die gesamte Menge dieser Pareto-optimalen Lösungen wird als *Pareto-Front* bezeichnet. Alle diejenigen Lösungen, die in allen Zielkriterien schlechter sind als irgendeine andere Lösung, werden *dominierte* Lösungen genannt. Mit anderen Worten entspricht die Lösung eines mehrkriteriellen Optimierungsproblems der Suche nach der Menge der *nicht dominierten* Lösungen. Eine Darstellung der Lösungsmenge eines mehrkriteriellen Optimierungproblems mit zwei zu minimierenden Kriterien ist in Abbildung 3.1 gegeben. Es wird deutlich, dass P_2 von P_1

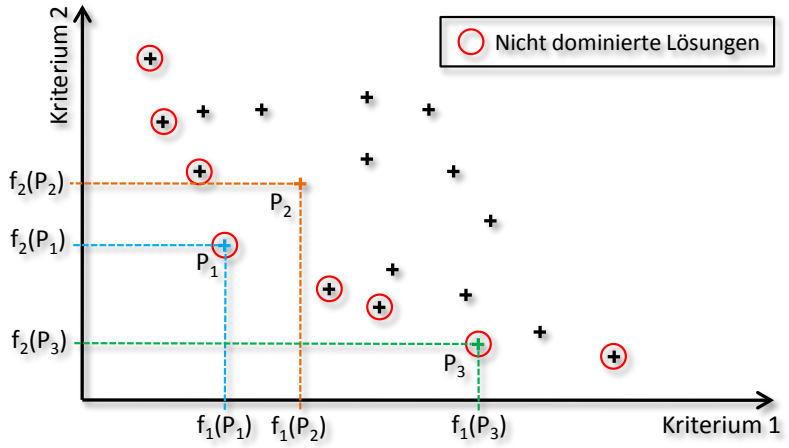


Abbildung 3.1: Darstellung einer Lösungsmenge für ein zweikriterielles Minimierungproblem und der Pareto-Front als Menge der nicht dominierten Lösungen.

dominiert ist, da sowohl $f_1(P_1) < f_1(P_2)$ als auch $f_2(P_1) < f_2(P_2)$ gilt und damit Lösung P_1 in jedem Falle gegenüber Lösung P_2 zu bevorzugen ist. Diese Entscheidung kann jedoch zwischen P_1 und P_3 nicht getroffen werden, da dort zwar $f_1(P_1) < f_1(P_3)$, aber auch andersherum $f_2(P_1) > f_2(P_3)$ gilt und sich damit diese beiden Punkte nicht dominieren. Da es sowohl für P_1 als auch für P_3 keinen anderen Punkt in der Lösungsmenge gibt, der diese jeweils dominiert, gehören diese zur hier rot umrahmten Pareto-Front. Ziel der Optimierung ist deshalb, eine möglichst gute *Konvergenz* zur Front zu erzeugen und gleichzeitig eine hohe *Diversität* in der Menge der Lösungen zu erhalten.

Auf eine weitere formelle Darstellung soll an dieser Stelle verzichtet und auf die umfangreichen Arbeiten von zum Beispiel Deb [48] verwiesen werden. Neben der Vielzahl von klassischen Lösungsverfahren, die zum Beispiel zusammenfassend von Miettinen [125] dargestellt werden, haben sich aufgrund der oftmals sehr komplexen Problemstellung vor allem evolutionäre Verfahren für die Lösung von mehrkriteriellen Problemen etabliert.

Aufgrund der in den letzten 20 Jahren sehr regen Forschungsaktivitäten auf diesem Gebiet, siehe Coello Coello [44], hat sich eine Vielzahl von unterschiedlichen Algorithmen durchgesetzt, deren vollständige Beschreibung den Rahmen dieser Arbeit übersteigen würde. Diese reichen von frühen Entwicklungen wie der Vector-Optimized Evolution Strategy (VOES) von Kursawe [109] und dem Random Weighted Genetic Algorithm (RWGA) von Murata und Ishibuchi [129] über den Multiple Objective Genetic Algorithm (MOGA) von Fonseca und Fleming [72] bis hin zu neueren Entwicklungen wie dem $\mathcal{S}$-Metric Selection-EMOA (SMS-EMOA) von Beume u. a. [31] oder dem Strength Pareto Evolutionary Algorithm (SPEA) von Zitzler u. a. [179]. Für eine umfassende Darstellung sei hier auf das Standardwerk von Coello Coello u. a. [45] verwiesen. Somit sieht sich der praktische Optimierer heutzutage einer kaum durchschaubaren Vielfalt von Algorithmen gegenübergestellt, sodass es schwer fällt, die richtige Wahl zu treffen.
Ein sehr wichtiger Algorithmus, der gleichzeitig ein ganzes evolutionäres Optimierungsparadigma, nämlich das der Pareto-Dominanz-basierten Selektion verkörpert, wurde in dieser Aufzählung zurückgestellt, da er im Folgenden im Detail beschrieben wird. Der „Non-dominated Sorting Genetic Algorithm (NSGA)“ und seine Weiterentwicklung NSGA-II sind heute sehr verbreitet eingesetzte Verfahren für die praktische Lösung von mehrkriteriellen Problemen. In dieser Arbeit wird die mehrkriterielle Betrachtung primär zum besseren Verständnis des Gesamtproblems genutzt, weshalb auf dieses vielfach erprobte Verfahren zurückgegriffen wird. Es ist dabei durchaus möglich, dass mit einem der obigen Verfahren noch bessere Lösungsergebnisse als die hier dargestellten erzielbar sind. Da aber eine vergleichende Analyse der erreichbaren Lösungsqualitäten bei unterschiedlichen mehrkriteriellen Algorithmen nicht im Fokus der Arbeit steht, werden die erreichten NSGA-II-Resultate als ausreichend betrachtet.

3.3.1 Non-dominated Sorting Genetic Algorithm

Wie schon in Abschnitt 2 angedeutet, basiert eine Reihe von Algorithmen auf dem von Goldberg [81] entwickelten Konzept der ebenenweisen Pareto-Selektion. Der 1994 von Srinivas und Deb [167] vorgestellte Non-dominated Sorting Genetic Algorithm (NSGA) realisierte als erster dieses Konzept, indem ein klassischer genetischer Algorithmus um eine speziell an multikriterielle Probleme angepasste Selektion erweitert wurde. Bevor der eigentliche Selektionsprozess beginnt, wird zunächst die gesamte Population auf Basis der Pareto-Dominanz beurteilt: Alle nicht dominierten Individuen der zu bewertenden Population werden in eine Kategorie mit Rang 1 eingeordnet und anschließend aus der Population entfernt. Aus den verbleibenden Individuen wird dann wieder die Menge der nicht dominierten ausgewählt und in eine weitere Kategorie mit Rang 2 eingeordnet. Dieses Verfahren wird so lange durchgeführt, bis alle Individuen einer Kategorie zugeordnet sind, siehe Abbildung 3.2. Dabei wird allen Individuen einer Kategorie ein „virtueller“ Fitnesswert zugeordnet, der proportional zur aktuellen Populationsgröße ist, um ein gleiches Reproduktionspotenzial für diese Individuen abzubilden. Da die Individuen in der Kategorie Rang 1 den größten möglichen Fitnesswert zugeschrieben bekommen haben, werden sie immer mehr Nachkommen als der Rest der Population für die nächste Generation produzieren. Offensichtlich ist dieses Vorgehen nicht sonderlich effizient[3], da die Pareto-Front-basierte Kategorisierung wiederholt ausgeführt werden muss. Weiterhin unterstützt dieser Algorithmus keine elitäre Selektion,

[3]Die Komplexität von NSGA ist nach Deb u. a. [50] für k Kriterien und μ Individuen $\mathcal{O}(k\mu^3)$, während sie bei NSGA-II auf $\mathcal{O}(k\mu^2)$ reduziert ist.

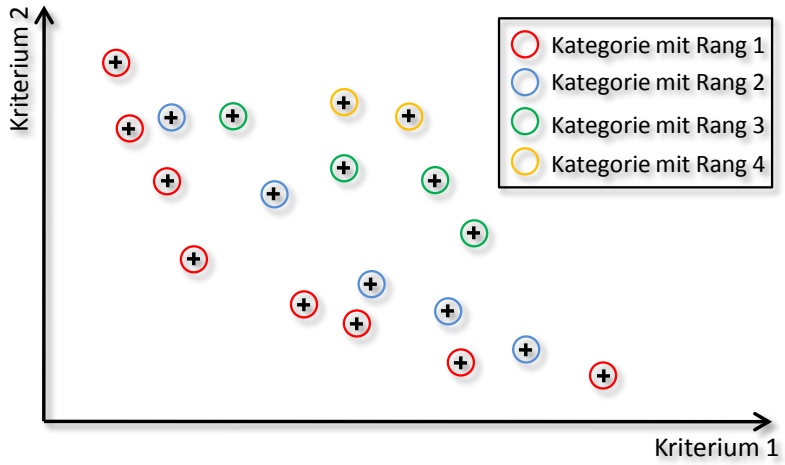


Abbildung 3.2: Einteilung einer Population in Kategorien mit unterschiedlichem Rang nach Pareto-Dominanz.

weshalb häufig gute Lösungen verloren gehen. Ein drittes Problem ergibt sich dadurch, dass in der Originalversion ein spezieller Parameter vorgegeben werden muss *(sharing parameter)*, der starken Einfluss auf die Diversität der Lösungen hat.

Als Konsequenz wurde von Deb u. a. [50] eine effizientere Variante mit der Bezeichnung NSGA-II vorgestellt, die sich durch ein verbessertes Konzept zur Berechnung und Diversitätserhaltung vom Vorgänger unterscheidet. Für jede Lösung werden dazu die Menge der von ihr dominierten Lösungen und die Menge der dominierenden Lösungen bestimmt. NSGA-II berücksichtigt zusätzlich die Dichte der Lösungen, die eine bestimmte Lösung umgeben, indem die durchschnittliche Distanz von zwei Nachbarlösungen für jedes Zielkriterium bestimmt wird. Dieses Verfahren wird *crowding distance* genannt und als Vergleichsoperator während der Selektion zusätzlich verwendet. Somit berücksichtigt die Selektion bei NSGA-II sowohl die Pareto-Kategorisierung als auch die Lösungsdichte, das heißt, nicht dominierte Lösungen werden gegenüber dominierten Lösungen bevorzugt, aber innerhalb einer Kategorie werden Lösungen in weniger dichten Regionen primär selektiert. Die elitäre Selektion wird dabei nicht wie bei anderen Verfahren durch ein Archiv umgesetzt, sondern durch die Kombination von den besten Eltern mit den besten Nachkommen, also quasi als eine Art $(\mu + \lambda)$-Selektion.

Durch diese Verbesserungen, die NSGA-II unter Komplexitätsbetrachtungen viel effizienter als seinen Vorgänger macht, hat dieser Algorithmus eine derart große Verbreitung gefunden, dass er zu einem der populärsten mehrkriteriellen evolutionären Optimierungsverfahren gezählt werden darf. Trotz auch zahlreicher identifizierter Nachteile, wie zum Beispiel Problemen bei der Optimierung mit sehr vielen Kriterien, siehe Knowles und Corne [105], müssen sich heute immer noch alle neuen Entwicklungen auf diesem Gebiet dem Vergleich mit NSGA-II stellen.

3.3.2 Alternative Ansätze zur parallelen Berechnung

Der Vollständigkeit halber sei hier noch erwähnt, dass der hohe Rechenaufwand der CI-Methoden und besonders der mehrkriteriellen evolutionären Algorithmen eine große Herausforderung bei der Lösung praktischer Probleme darstellt. In dieser Arbeit sind es besonders die Simulationen eines Grid-Systems, die teilweise einer sehr langen Berechnungszeit bedürfen, sodass unter anderem die parallele Ausführung auf einem Rechencluster mit bis zu 200 Knoten notwendig wurde. Hauptproblem bei der Ausführung von generationsbasierten evolutionären Algorithmen wie traditionellen Evolutionsstrategien, aber auch NSGA-II ist die geringe Auslastung dieses Clusters durch sogenannte *barriers* und die damit verbundenen langen Optimierungszeiten. Die Evaluation einer neuen Generation kann erst starten, wenn die gesamte vorherige Population bewertet wurde. Dies ist besonders kritisch, wenn die Auswertung eines einzigen Individuums deutlich länger dauert als die der restlichen Individuen.
Um den Cluster besser für die aufwendigen Simulationen nutzen zu können, wurden deshalb alternative evolutionäre Ansätze untersucht, die nicht auf dem Generationenmodell basieren. Das parallele Räuber-Beute-Modell für die mehrkriterielle Optimierung bietet eine vielversprechende Alternative. Um hier eine Anpassung für die praktische Nutzung von Höchstleistungsrechenressourcen zu ermöglichen, wurde das ursprüngliche von Laumanns u. a. [110] entworfene und von Grimme und Schmitt [85] erweiterte Modell vollkommen modularisiert und für den allgemeinen Entwurf von Variationsoperatoren angepasst, siehe Grimme und Lepping [12]. Auf dieser Basis wurden dann einfache Mutations- und Rekombinationsoperatoren und deren Einfluss auf das System genau untersucht und theoretisch erklärt [13]. Weiterhin konnte das Modell auch für die praktische Anwendung weiterentwickelt werden, wobei besonders die kombinatorische Optimierung und Probleme aus dem Bereich des Job-Scheduling im Vordergrund [18] standen. Zuletzt ließ sich ein kombiniertes Verfahren aus Räuber-Beute-Modell und integrierter lokaler Suche [19] als ein hybrides CI-Verfahren umsetzen.
Da diese parallelisierten Verfahren trotz aller Fortschritte noch nicht genügend für reale Probleme erprobt sind und hier Individuen mit besonders langen Chromosomen optimiert werden, wurde dennoch auf den etablierten Standardalgorithmus NSGA-II für die Evaluation zurückgegriffen und die parallele Evaluation unabhängiger Individuen einer gesamten Population *(embarrassingly parallel workload)* realisiert.

4 Evolutionäre Fuzzy-Systeme

UNTERSCHIEDLICHE ELEMENTE gehören in der traditionellen Mengentheorie entweder einer bestimmten Menge an oder nicht. Ebenso nehmen in einer binären Logik Parameter immer entweder den Wert „wahr“ oder „falsch“ (1 oder 0) an, wobei diese Bedingung auch für das Ergebnis eines Inferenzprozesses gilt. Das menschliche Denken ist jedoch in vielen Fällen nicht exakt, sondern fußt vielfach auf einem gewissen Maß von Unsicherheit und Ungewissheit. Die linguistische Formulierung basiert zumeist auf vagen oder subjektiven Termen, wie zum Beispiel „Es ist ziemlich kalt“ oder „Die Person ist sehr alt“. Die Fuzzy-Mengen-Theorie geht nun von der Annahme aus, dass alle Dinge nur zu einem gewissen Grad zutreffen, und reduziert die herkömmliche Logik auf einen Sonderfall. Gerade der Mangel an Präzision ermöglicht das Treffen von Entscheidungen selbst in Situationen, in denen unvollständige oder teilweise widersprüchliche Informationen vorliegen. Fuzzy-Mengen und Fuzzy-Logik erlauben es, durch die Einführung einer kompletten Algebra auf unscharfen Mengen auch ein ungefähres Schließen *(approximate reasoning)* durchzuführen. Die Fuzzy-Logik ermöglicht es also, Argumentationen auf Basis von unsicheren Tatsachen zu führen und neue Tatsachen mit einem gewissen Maß an Sicherheit aus anderen unsicheren Tatsachen abzuleiten.
Es ist wichtig zu bemerken, dass die hier gemeinte Unsicherheit bei Fuzzy-Systemen immer in einem *nicht* statistischen Sinne zu verstehen ist und nicht mit der klassischen statistischen Unsicherheit verwechselt werden darf. Statistische Unsicherheit basiert auf den Gesetzen der Wahrscheinlichkeit, während nicht statistische Unsicherheit auf Unbestimmtheit, Ungenauigkeit und/oder Unklarheiten beruht. Statistische Unsicherheit kann durch Beobachtungen behoben werden, indem zum Beispiel nach einem Münzwurf das Ergebnis unweigerlich bekannt ist, während vor dem Wurf für das Ergebnis jeweils nur eine Wahrscheinlichkeit von 50 % angegeben werden kann. Die nicht statistische Unsicherheit hingegen ist eine inhärente Eigenschaft eines Systems und kann nicht durch weitere Beobachtungen verringert werden.

4.1 Fuzzy-Mengen

Eine Fuzzy-Menge oder auch Fuzzy-Teilmenge g der Grundmenge X ist eine Abbildung

$$g : X \to [0, 1],$$

die jedem Element $x \in X$ seinem Zugehörigkeitsgrad $g(x)$ zu g zuordnet.[4] In vielen Fällen besteht die Grundmenge X aus Werten, die eine reellwertige Variable annehmen kann,

[4] In dieser Arbeit wird konsequent g für den Zugehörigkeitsgrad und $g(x)$ für die Zugehörigkeitsfunktion entgegen der üblichen Notation von μ und $\mu(x)$ als Zugehörigkeitsgrad beziehungsweise -funktion verwendet. Die Variable μ bezeichnet hingegen die Elternpopulation einer Evolutionsstrategie.

sodass X fast immer ein reelles Intervall ist. Eine Fuzzy-Menge g ist dann eine reelle Funktion mit Werten im Einheitsintervall, die beispielsweise durch die Zeichnung ihres Graphen veranschaulicht werden kann.

$$g : \mathbb{R} \to [0,1], \quad x \longmapsto \begin{cases} \frac{x-a}{b-a} & \text{wenn } a \leq x \leq b \\ \frac{c-x}{c-b} & \text{wenn } b \leq x \leq c \\ 0 & \text{sonst} \end{cases} \tag{4.1}$$

Ein Beispiel für eine dreiecksförmige Zugehörigkeitsfunktion in Gleichung 4.1 ist in Abbildung 4.1 dargestellt (mit $a < b < c$). Dabei ist es, wie sich später noch zeigen wird, besonders wichtig, sich bei der Wahl von Fuzzy-Mengen auf konvexe Funktionen zu beschränken, da deren Beschreibung meist mit wenigen Parametern realisiert werden kann.

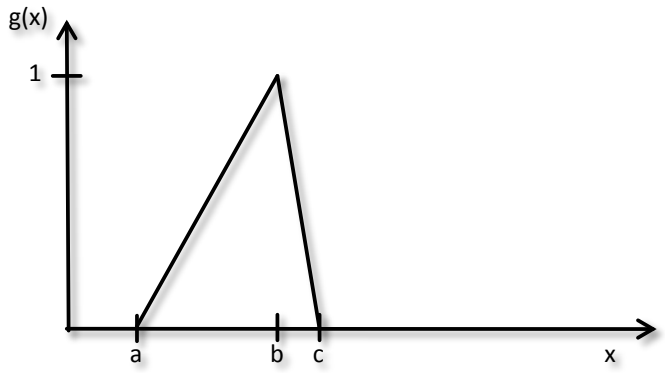
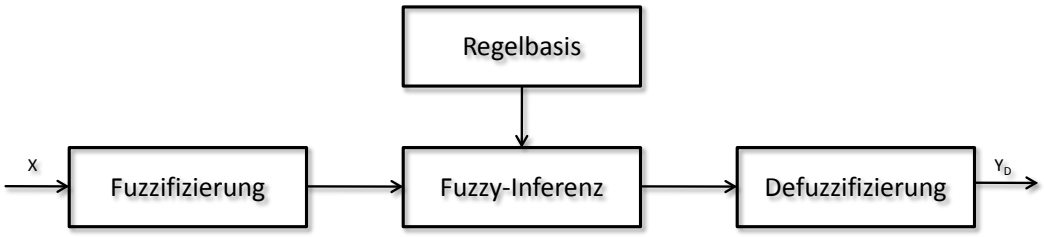


Abbildung 4.1: Darstellung einer dreicksförmigen Fuzzy-Menge nach Gleichung 4.1.

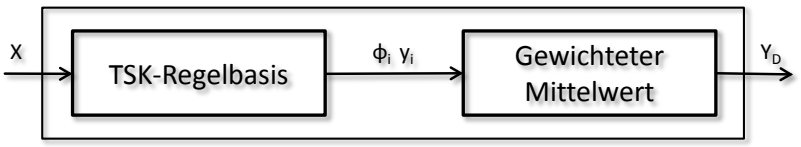
Durch das Konzept der Zugehörigkeitsfunktionen wird erreicht, dass unterschiedliche reelle Werte einer Eingangsvariablen demselben linguistischen Wert zugeordnet werden. Zum Beispiel werden alle Lebensalter größer als 70 Jahre dem linguistischen Wert „alt“ zugeordnet. Durch derartige Abbildungen wird der kontinuierliche Wertebereich einer Variablen granularisiert und somit die Komplexität des Modells reduziert. Hierdurch kann die Problemlösung, wenn die richtige Balance zwischen wünschenswerter Vereinfachung und noch ausreichender Differenzierung gefunden wurde, stark vereinfacht werden, wobei die Balance genau durch die Anpassung der Zugehörigkeitsfunktionen beeinflusst wird. Diese Vereinfachung und Reduktion der Komplexität kann als ein Schlüssel dafür gesehen werden, dass im alltäglichen Leben Menschen in der Lage sind, mit ihrer Umgangssprache Probleme zu bewältigen. Die sich ständig ändernde Ausprägung einer Sprache zeigt, dass die Granularisierung für die Orientierung in einer in gewissen Teilbereichen komplexer werdenden Umwelt ständig adaptiert werden muss.
Das größte Anwendungsgebiet für Fuzzy-Mengen und Fuzzy-Logik findet sich bis heute auf dem Gebiet der Regelungstechnik beziehungsweise bei der Umsetzung von regelbasierten Systemen.

4.2 Konzepte der Fuzzy-Regelsysteme

Die Fuzzy-Regler stellen eine Erweiterung der klassischen regelbasierten Systeme dar, weil sie auf Basis von „WENN-DANN“-Regeln arbeiten und Bedingungen und Entscheidungen von Regeln mittels Fuzzy-Mengen beschrieben werden. Im weiteren Sinne ist ein Fuzzy-Regler ein regelbasiertes System, in dem Fuzzy-Logik als Werkzeug für die Darstellung von verschiedenen Formen von Wissen über bestimmte Probleme sowie für die Modellierung der Interaktionen und Beziehungen zwischen deren Parametern genutzt wird.
Zur Verarbeitung von Regeln, deren Prämissen und Konklusionen Fuzzy-Wahrheitswerte annehmen können, dienen die in der Fuzzy-Logik entwickelten Fuzzy-Operatoren, siehe Kandel und Langholz [103]. Bei einem Fuzzy-Regler werden also die Eingangsgrößen durch geeignete Zugehörigkeitsfunktionen „fuzzifiziert“ und das ursprüngliche im kontinuierlichen Raum angesiedelte Problem wird in eine diskrete Welt der linguistischen Terme transformiert und dort mithilfe der Fuzzy-Logik gelöst (Fuzzy-Inferenz). Über die Zugehörigkeitsfunktionen wird das Ergebnis schließlich wieder in die kontinuierliche Welt zurücktransformiert oder „defuzzifiziert“, siehe Abbildung 4.2(a). Dieses Konzept wurde 1974 von Mamdani und Assilian [119] entwickelt und ist seitdem das am häufigsten eingesetzte Fuzzy-Regler-Verfahren.



(a) Mamdani-Regler



(b) TSK-Regler

Abbildung 4.2: Struktureller Aufbau von Mamdani- und TSK-Fuzzy-Reglern.

Der Vorteil dieses Verfahrens liegt darin, dass es einen natürlichen Rahmen vorgibt, um Expertenwissen in die Form der sprachlichen Regeln umzusetzen. Dieses Wissen lässt sich einfach mit Regeln verbinden, die das Verhältnis zwischen Eingang und Ausgang des Systems beschreiben. Dabei sind seit Jahren entsprechende Entwurfsverfahren für die automatische Extraktion von Regeln aus gegebenen Daten vorhanden, siehe zum Beispiel Slawinski u. a. [160]. Weiterhin erlaubt ein Mamdani-Regler ein hohes Maß an Freiheiten in der Formulierung beliebiger Fuzzifizierungs- und Defuzzifizierungsmethoden sowie der Ausgestaltung der Inferenz-Methode, sodass eine problemspezifische Anpassung leicht und flexibel möglich ist. Nachteile ergeben sich allerdings auch aus einem potenziellen Mangel an Genauigkeit durch die Struktur der sprachlichen Regeln. So wird die Flexibilität der Regler durch die

starre Aufteilung des Eingangs- und Ausgangsbereichs eingeschränkt und bei gegenseitigen Abhängigkeiten zwischen verschiedenen Eingangsvariablen ist es mitunter sehr schwierig, überhaupt geeignete starre Aufteilungen des Eingangsbereichs zu finden. Weiterhin steigt mit höherer Problemkomplexität auch der Bedarf der Berücksichtigung weiterer Eingangsvariablen und entsprechender Regelbeschreibungen, wodurch die Dimensionalität und die Gesamtkomplexität der Fuzzy-Beschreibungen stark ansteigen. Neben der schwierigen Interpretation derartiger Regelsysteme steigen die Anforderungen an die entsprechenden Entwurfsverfahren, geeignete Aufteilungen des Eingangsbereichs zu erzeugen. Mitunter sind dann mehrstufige Verfahren nötig, die, der Erzeugung eines initialen Regelsystems nachgelagert, eine Reduktion der Regelkomplexität berücksichtigen. Weitere Probleme entstehen durch die Verwendung von Fuzzy-Mengen als Ausgangsgrößenbeschreibung und den entsprechenden Methoden zur Defuzzifizierung, wo es in bestimmten Fällen zu sprunghaften Veränderungen der Ausgangsgröße oder starken Ungenauigkeiten durch notwendige Interpolationen kommen kann, siehe auch Michels u. a. [124].
Deshalb wurde 1985 von Takagi, Sugeno und Kang [169] ein Fuzzy-Modell (TSK-Modell) entworfen, bei dem für die Ausgaben nur scharfe Werte verwendet werden. Für die Beschreibung der Eingabewerte werden weiterhin Fuzzy-Mengen genutzt und die Konsequenzen der Regeln werden als Linearkombinationen der Eingangsmengen beschrieben, siehe Kandel und Langholz [103]. Dadurch gestaltet sich die Defuzzifizierung als sehr einfach, weil nur der Mittelwert aus den mit den zugehörigen Erfüllungsgraden ϕ_i der Regeln gewichteten Ausgabewerten y_i gebildet werden muss, siehe Abbildung 4.2(b). Genaue Details der Berechnung sind in Kapitel 12.2 geschildert. Dabei wird im Gegensatz zum traditionellen Ansatz einer Gesamtmodellierung des Systems ein Multi-Modell-Ansatz verfolgt, in dem das globale Systemverhalten durch Einzelkomponenten abgebildet wird. Das gesamte Fuzzy-Modell wird dadurch als Kombination von verbundenen Subsystemen erstellt, die sich jeweils durch geringere Komplexität auszeichnen. Der große Vorteil des TSK-Modells ist dabei seine Repräsentationsfähigkeit. So ist es zum Beispiel möglich, auch hochgradig nichtlineare Systeme mit einer geringen Anzahl von Regeln zu beschreiben. Weiterhin sind TSK-Modelle durch die Darstellung der Ausgabe als explizite funktionale Zusammenhänge besonders für externe Optimierungsverfahren wie evolutionäre Algorithmen geeignet. Nachteile ergeben sich allerdings durch die komplexe Struktur der Regelkonsequenzen, wodurch ihre Verständlichkeit und Interpretierbarkeit stark eingeschränkt ist. Weiterhin führt eine starke Überlappung der Regeln im Bedingungsteil zu einer starken Verwischung der Einzelmodelle. Durch die Mittelwertbildung kann es dabei dann leicht zu Überschwingern in der Ausgabe führen, auch wenn nur geringe Überlappungen im Eingangsbereich vorhanden sind. Es ist deshalb zur Verhinderung derartiger unerwünschter Effekte besonders wichtig, die Zugehörigkeitsfunktionen sorgfältig auszuwählen. Üblicherweise sind Dreiecksfunktionen deshalb nicht empfehlenswert und sollten beim TSK-Modell lieber durch Trapezfunktionen oder andere Alternativen, siehe Kapitel 12.3.1, ersetzt werden.
Für diese beiden Modelle existiert zwar eine Reihe von Lernverfahren, die aber alle auf das Vorhandensein einer sogenannten Wissensbasis angewiesen sind. Es ist also immer notwendig, schon in irgendeiner Form über Expertenwissen zu verfügen, das dann mittels unterschiedlicher Verfahren in Fuzzy-Regeln abgebildet werden kann. Für viele praktische Probleme, wie unter anderem das in dieser Arbeit betrachtete, ist ein derartiges Expertenwissen jedoch nicht verfügbar, sodass die klassischen Fuzzy-Entwurfsverfahren nicht unmittelbar anwendbar sind. Im Sinne einer hybriden Problemlösungsstrategie der Computational Intelligence ist es dann aber immer noch möglich, die Anpassung eines Fuzzy-Systems durch

einen evolutionären Algorithmus durchführen zu lassen. Dabei beziehen sich die kritischen Punkte des Entwurfes sowohl auf die Optimierung der Eingangsbedingungen als auch auf die für gegebene Bedingungen sinnvollen Regelempfehlungen. Im Folgenden werden nur evolutionäre Fuzzy-Systeme auf Basis des TSK-Modells genauer beschrieben. Für weitergehende Ausführungen zum evolutionär gestützten Fuzzy-Entwurf sei auf das umfangreiche Werk von Cordón u. a. [46] verwiesen.

4.3 Evolutionäre Optimierung von Fuzzy-Systemen

Bei einem evolutionären Fuzzy-System wird ein evolutionärer Algorithmus genutzt, um die Struktur, den Gültigkeitsbereich von Regeln und deren jeweilige Regelausgaben extern zu optimieren. Dabei kann die automatische Erzeugung von Fuzzy-Systemen als ein großes Optimierungsproblem angesehen werden, für das sich evolutionären Algorithmen als Lösungsverfahren besonders eignen, da diese in der Lage sind, auch sehr große und vollständig unbekannte Suchräume effizient und unter minimalem externem Zusatzwissen zu handhaben. Zusätzlich sind die generische Codestruktur und die unabhängigen Stellgrößen eines evolutionären Algorithmus ideal geeignet, um eventuell vorhandenes Vorwissen über das Problem, wie zum Beispiel sinnvolle Wertebereiche der Eingangskenngrößen oder eine sinnvolle Anzahl von Regeln pro Fuzzy-Regler, mit in die Problemlösung einfließen zu lassen. Allgemein lassen sich zwei große Entwurfsprinzipien für evolutionäre Fuzzy-Systeme unterscheiden, die sich jeweils auf die Abbildung von Fuzzy-Systemen in einem evolutionären Algorithmus beziehen: der Michigan-Ansatz nach Bonarini [34] und der Pittsburgh-Ansatz nach Smith [162].
Beim Michigan-Ansatz repräsentiert ein Individuum in einem evolutionären Algorithmus genau eine einzelne Regel. Das gesamte Inferenzsystem wird dann durch eine gesamte Population dargestellt. Da natürlich mehrere Regeln an dem Inferenzprozess beteiligt sind, entsteht während der evolutionären Selektionsprozesse unter den Regeln ein ständiges Streben nach guten Regelentscheidungen für bestimmte Situationen, wobei einzelne Regeln nur durch die Interaktion mit anderen Regeln überhaupt auswertbar sind. Durch diese kooperativ-kompetitive Eigenschaft dieses Konzeptes ist es sehr schwer zu entscheiden, welche Regeln letztendlich für den Erfolg oder Misserfolg einer konstituierten Regelbasis verantwortlich zu machen sind. Deshalb bedarf es sehr ausgeklügelter Mechanismen, um aus dem Gesamtresultat den Fitnesswert für Einzelregeln sinnvoll zu extrahieren. Der Michigan-Ansatz wird daher primär bei nicht induktiven Problemstellungen verwendet, bei denen das Problem nicht als Gesamtheit handhabbar ist, sondern besser durch die Anpassung an Teilprobleme lösbar ist. So ist der Michigan-Ansatz auch in den Kontext von symbiotischen oder kooperativen Verfahren einzuordnen. Beispielhafte Studien dazu sind unter anderem bei Juang u. a. [100] und Anwendungen im Bereich Scheduling bei Franke u. a. [7] zu finden.
Der Pittsburgh-Ansatz operiert hingegen auf einer Population von vollständigen Fuzzy-Systemen, wobei in dem Chromosom eines Individuums ein komplettes Regelsystem abgebildet wird. Die evolutionären Operatoren erzeugen dabei durch Variation ein neues Fuzzy-System, das sich im evolutionären Entwicklungsprozess somit ständig verbessert, da es die Eigenschaften der Eltern entsprechend weiterentwickelt. Dieses Lernkonzept ist in seiner allgemeinen Struktur in Abbildung 4.3 dargestellt. Da hier keine Kompositionen von Regelsystemen aus Einzelregeln notwendig ist, kann auch die Zuweisung von Fitnesswerten problemlos durchgeführt werden, wodurch sich dieser Ansatz prinzipiell besser für die An-

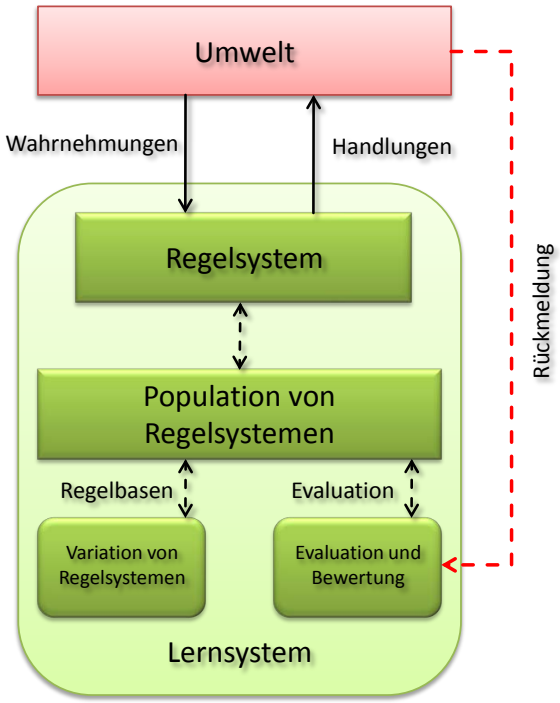


Abbildung 4.3: Evolutionäres Lernkonzept für Fuzzy-Systeme nach dem Pittsburgh-Ansatz.

wendung in einem evolutionären Algorithmus eignet. Ein Nachteil ergibt sich allerdings aus der hohen Zahl von Parametern, die eine Kodierung des ganzen Systems mit sich bringt, und dem sich daraus ergebenden großen Suchraum. Dies ist besonders bei der Behandlung komplexer Probleme kritisch, bei der zur Modellierung eine Vielzahl von Regeln benötigt wird. In solchen Fällen ist es wichtig, gut angepasste Variationsoperatoren zu wählen und die evolutionäre Suche durch die Integration von externem Problemwissen möglichst zu unterstützen. Wenn zunächst kein spezielles Wissen über die Beschaffenheit des zu regelnden Systems vorliegt, ist es ratsamer, die gesamte Regelbasis direkt zu optimieren und symbiotische Effekte nicht zu berücksichtigen. Weiterhin hat sich in der Vergangenheit die Konstruktion eines leistungsfähigen Bewertungssystems für den Michigan-Ansatz als äußerst schwierig und nicht immer zielführend herausgestellt, siehe Franke u. a. [7]. Deshalb werden in dieser Arbeit die evolutionären Fuzzy-Systeme auf Basis des Pittsburgh-Ansatzes erzeugt.
Obwohl klassische und evolutionäre Fuzzy-Systeme bereits in einer Vielzahl unterschiedlicher praktischer Probleme ihre Anwendbarkeit bewiesen haben, so existieren dennoch sehr wenige Arbeiten in Bezug auf Scheduling Computational Grids. Huang u. a. [93] bestimmen eine Wissensbasis für Grid-Computing durch die Erzeugung von Expertenwissen aus intensiv aufgezeichneten Daten existierender Grids. Dazu werden entsprechende Muster für sinnvolle Schedulingentscheidungen aus den Aufzeichnungen extrahiert und dann entsprechend in Fuzzy-Regeln abgebildet. Dieses Vorgehen erfordert aber die Verfügbarkeit sehr detaillierter Zustandsaufzeichnungen (Monitoring-Data) und ist nur in gering bis gar nicht informationsbeschränkten Umgebungen einsetzbar. Fayad u. a. [61] untersuchen ein Grid-Schedulingproblem mit Fälligkeitsterminen, bei dem die Ausführungszeiten von Jobs einer gewissen Unsicherheit unterliegen. Sie nutzen Fuzzy-Mengen, um diese Unsicherheiten zu modellieren, und kombinieren dies mit einer Tabu-Suche, um die Anzahl der fristgerecht

beendeten Jobs zu maximieren. Dazu wird das Vorgehen mittels realer Arbeitslastaufzeichnungen evaluiert. Es wurden dort allerdings nur sequenzielle Jobs berücksichtigt und künstliche Fälligkeitstermine erzeugt. Weiterhin wird in dem Grid-Szenario ein direkter Zugriff der Grid-Scheduler auf die lokalen Ressourcen erlaubt.
Die Problematik des Scheduling sowohl auf Parallelrechnern als auch in Computational Grids sowie Untersuchungen zur Simulation von Grids werden im folgenden Teil dieser Arbeit genauer dargestellt.

Teil II

Job-Scheduling auf Parallelrechnern und in Computational Grids

5 Job-Scheduling auf Parallelrechnern

COMPUTATIONAL GRIDS werden in dieser Arbeit etwas vereinfacht als Zusammenschluss von konventionellen Parallelrechnern verstanden und modelliert. Deshalb wird hier zunächst auf ein Modell und das resultierende Schedulingproblem auf Parallelrechnern eingegangen. Dabei werden weiterhin das dieser Arbeit zugrunde liegende Maschinen- und Jobmodell erläutert, sowie Lösungsverfahren für die Allokation von Jobs zu den Knoten eines Parallelrechners vorgestellt.

5.1 MPP-Systeme und Cluster

Wir betrachten also als kleinste Ausführungseinheiten in einem Computational Grid einzelne Parallelrechner. Diese werden durch unterschiedlichste architektonische Konzepte realisiert, wobei heutzutage die sogenannten *Massively-Parallel-Processor*-(MPP-)Systeme und *Cluster* die größte Verbreitung finden. Obwohl hier nicht genauer auf die verschiedenen Architekturen und Umsetzungen von Parallelrechnern eingegangen wird, sollen doch die wesentlichen Eigenschaften herausgestellt werden, wodurch die dann im Abschnitt 5.2.1 geschilderten Modellannahmen und Vereinfachungen gerechtfertigt erscheinen.
In MPP-Systemen werden meist baugleiche Prozessoren über ein extrem leistungsfähiges Verbindungsnetzwerk miteinander verbunden. Dadurch ist es möglich, die Berechnung von Jobs auf mehrere dieser Prozessoren zu verteilen. Durch die somit erreichte Parallelverarbeitung kann eine kürzere Rechenzeit erzielt werden als bei einer rein sequenziellen Ausführung. Dies geschieht durch die effiziente Ausnutzung der Kommunikationsinfrastruktur, die eine Interprozesskommunikation innerhalb eines Jobs nahezu ohne Verzögerungen ermöglicht. Diese Kommunikation wird durch das „Message Passing“ für den Nachrichtenaustausch bei parallelen Berechnungen auf verteilten Computersystemen realisiert. Die Größe der heutigen MPP-Systeme reicht von 100 und bis zu einigen 1.000 Prozessoren, wobei gerade kleine Installationen in fast allen naturwissenschaftlichen Einrichtungen verfügbar sind. Der Trend zur parallelen Ausführung bestimmter Aufgaben innerhalb eines Programms, der sich zurzeit durch die Einführung von Multi-Core-Architekturen auch auf einfachen Desktop-Systemen durchsetzt, ist im akademischen Umfeld durch den traditionellen Einsatz von MPP-Systemen bereits lange vorhanden.
In den letzten Jahren hingegen wächst die Bandbreite von Kommunikationsverbindungen für Rechnernetze schneller als die Rechengeschwindigkeit von Prozessoren und hat bereits Größenordnungen von mehr als 100 Gigabit/s erreicht. Damit wird der traditionelle Unterschied zwischen Parallelrechnern mit ihren aufwendigen und spezialisierten Verbindungsnetzwerken sowie handelsüblichen Rechnernetzen immer mehr verwischt. Als kostengünstige Alternative für die Umsetzung von parallelen Systemen haben sich deshalb Cluster

etabliert. Diese stellen eine Form von paralleler Rechnerarchitektur dar, die durch ein Ensemble von unabhängigen (zumeist handelsüblichen) Recheneinheiten gebildet wird, in der jeder Knoten einen eigenen Prozessor mit eigenem Arbeits- und Festplattenspeicher besitzt. Diese Knoten werden dann durch ein Verbindungsnetzwerk (zum Beispiel LAN) zu einem koordinierten System zusammengefasst. Cluster werden hauptsächlich für die Bereitstellung von Rechenleistung und die Durchführung von extrem aufwendigen Berechnungen (z.B. Klima- oder CFD-Simulationen) genutzt. Die Anwendung von I/O-intensiven Operationen wie zum Beispiel die Abfrage auf Datenbanken spielen aufgrund der nicht optimierten Kommunikationswege eher eine untergeordnete Rolle. Wegen der einfachen Installation und der Verwendbarkeit von handelsüblicher Hardware sind Cluster sowohl im akademischen als auch im industriellen Umfeld immer häufiger anzutreffen. Die aktuelle Liste der 500 schnellsten Hochleistungsrechner der Welt[5] verzeichnet einen Anteil von ungefähr 84 % Clusterinstallationen und nur ungefähr 14 % traditioneller MPP-Systeme. Für die groß angelegte Bereitstellung von parallel nutzbaren Rechenressourcen kommt den Clusterinstallationen also die vorrangige Bedeutung zu.
Für das weitere Vorgehen im Rahmen dieser Arbeit ist es nun notwendig, geeignete Abstraktionen beziehungsweise notwendige Vereinfachungen vorzunehmen, um ein Modell von parallelen Rechenressourcen als Entitäten in einem Computational Grid zu erstellen.

5.2 Modell eines Parallelrechners

Aktuelle Installationen von Parallelrechnern oder Clustern unterscheiden sich stark in der verwendeten Hardware. Es finden sich Systeme mit unterschiedlich großem Hauptspeicher oder Festplattenspeicher. Die verwendeten Prozessorfamilien lassen sich hingegen auf einige wenige Formen reduzieren. Unter den momentan schnellsten 500 Installationen finden sich ungefähr 80 % Intel-Architekturen und ungefähr 8 % PowerPC- und 9 % AMD-Prozessoren. Daher kann im High-Performance Bereich von einer gerechtfertigten Ähnlichkeit der Maschinen nicht nur innerhalb eines Großrechensystems, sondern auch zwischen diesen Systemen ausgegangen werden. Selbstverständlich gibt es auch innerhalb der Architekturen auch Performanceunterschiede, da nicht alle Intelprozessoren identisch sind. Jedoch kann davon ausgegangen werden, dass für die entsprechenden HPC-Systeme nur die jeweiligen High-End Produkte Verwendung finden. Weiterhin sind für die hier betrachteten Ressourcenallokations- und Schedulingprobleme Unterschiede in der *zusätzlich* vorhandenen Hardware nicht von Interesse und deshalb zu vernachlässigen.

5.2.1 Maschinenmodell

In dieser Arbeit wird ein Parallelrechner als Verbund von m_k identischen Maschinen modelliert, was nach Pinedo [137] einem sogenannten P_m Modell entspricht. Weiterhin wird eine unendlich schnelle Netzwerkverbindung zwischen den Knoten angenommen, die keine besonderen Kommunikationsmuster unterstützt. Deshalb kann ein paralleler Job auf jeder Untermenge dieser m_k Prozessoren ausgeführt werden. Jedoch werden mögliche Partitionen der Maschinen auf einem System nicht erlaubt. Ausführungsrestriktionen auf den Maschinen eines Parallelrechners (machine eligibility constraints) werden also nicht berücksichtigt,

[5] `http://www.top500.org`, Juni 2010.

da sie einer allgemeingültigen Analyse im Wege stehen würden. In realen Systemen werden zwar entweder Hardwarepartitionen oder auch die Priorisierungen von Nutzergruppen durch solche Restriktionen umgesetzt, aber sie führen zwangsläufig zu einer schlechteren Auslastung des Gesamtsystems. Franke u. a. [10, 11] entwickelten bereits Verfahren, die eine Umsetzung von Priorisierungen auch ohne Restriktionen und zugleich ohne Verringerung der Auslastung ermöglichen. Um prinzipiell jede Form von Partitionierungen zu ermöglichen und keine lokale Konfiguration in der Analyse zu bevorzugen, soll daher hier nur die freie Zuteilung von Jobs auf eine Menge von identischen Maschinen betrachtet werden.

5.2.2 Jobmodell

Weitere Überlegungen betreffen die Charakteristiken paralleler Jobs, die zur Verarbeitung eingereicht werden. Feitelson und Rudolph [67] klassifizieren Jobs zum einen danach, ob die Anzahl belegter Prozessoren durch den Benutzer vorgegeben oder durch das System gewählt wird. Zum anderen unterscheiden sie, ob die Anzahl belegter Prozessoren einmal beim Start des Jobs festgelegt wird und anschließend nicht modifiziert werden kann oder ob sie während der Jobausführung variabel ist.

Für sogenannte *rigid* Jobs wird demnach eine feste Anzahl benötigter Prozessoren vom Benutzer vorgegeben, wohingegen die Prozessoranzahl von *evolving* Jobs während der Jobausführung vom Benutzer modifiziert werden kann. Für die sogenannten *moldable* und *malleable* Jobs wird dabei ein Arbeitsvolumen in Prozessorzeiteinheiten spezifiziert. Moldable Jobs werden beim Start vom System selbst (und nicht vorher vom Nutzer) eine feste Anzahl an Prozessoren zugewiesen, wohingegen die Prozessoranzahl von malleable Jobs während der Jobausführung auch noch durch das System variiert werden kann. Die Realisierung einer derart flexiblen Handhabung der Ressourcenanforderungen muss selbstverständlich durch die Jobarchitektur unterstützt werden, was den Programmentwurf deutlich kompliziert. Obwohl nach Cirne und Berman [43] bis zu 98 % der auf Parallelrechner eingereichten Jobs prinzipiell moldable sind, finden sich doch in den vorhandenen Arbeitslastaufzeichnungen, siehe Abschnitt 7.1, nur statische Informationen über die verwendete Anzahl von Prozessoren. Dies legt die Vermutung nahe, dass von realen Systemen die flexible Zuweisung von Prozessoren durch das Schedulingsystem nicht genutzt wird und diese üblicherweise vom Nutzer zum Zeitpunkt der Einreichung fest vorgegeben wird. Malleable Jobs werden hingegen fast ausschließlich vor dem Hintergrund einer konkreten Hardwareumgebung und für einen konkreten, oftmals mathematischen Anwendungskontext behandelt, siehe Kalé u. a. [101]. Für die praktische Anwendung spielen diese Art von Jobs fast keine Rolle, da die Realisierung einer derart flexiblen Architektur in keinem Verhältnis zu dem dadurch erreichbaren Performancevorteil steht.

In dieser Arbeit werden daher starre, parallele Batch-Jobs *(ridgid jobs)* $j \in \tau_k$ angenommen, die von lokalen Nutzern zum Zeitpunkt $r_j \geq 0$ ins System eingereicht werden. Dabei bezeichnet τ_k die Menge aller Jobs, die von einer lokalen Nutzergemeinschaft in das System eingereicht werden. Der Zeitpunkt r_j ist gleichzeitig der frühestmögliche Startzeitpunkt, wobei hier Einreichungszeitpunkt *(submission time)* und Verfügbarkeitszeitpunkt *(release time)* identisch sind, da das System keine zukünftigen Reservierungen *(advanced reservations)* für Jobs unterstützt.

Zum Einreichungszeitpunkt r_j wird weiterhin der Parallelitätsgrad $m_j \geq 1$ fest vorgegeben und ist im System bekannt.[6] Jeder Knoten der parallelen Ressourcen kann dabei ausschließlich von nur einem Job belegt werden *(space-sharing)*, wodurch in einem gültigen Schedule Doppelbelegungen von Ressourcen ausgeschlossen sind.
Die Jobs besitzen eine Ausführungszeit $p_j > 0$, die allerdings erst nach deren Beendigung zum Zeitpunkt $C_j(S_k) \geq r_j + p_j$ im System k bekannt ist. In dieser Beziehung bezeichnet die Abhängigkeit vom Schedule $S_k \in \mathbb{S}_k$, dass $C_j(S_k)$ natürlich je nach Planung der Allokationen variiert. Dabei beschreibt $\mathbb{S}_k$ die Menge aller gültigen Schedules für das System k, siehe Abschnitt 5.3. Sollte es sich allerdings um keine mehrdeutigen Bezüge zu unterschiedlichen Schedules S_k handeln, wird in dieser Arbeit um der besseren Lesbarkeit willen vereinfacht C_j geschrieben.
Zwischenzeitliche Unterbrechungen *(preemptions)* der Ausführungen werden nicht unterstützt. Jeder Job wird deshalb zum Zeitpunkt $(C_j - p_j)$ gestartet und wartet vom Einreichungszeitpunkt bis zum Start für den Zeitraum $(C_j - p_j - r_j)$. Eine Darstellung der zeitlichen Verhältnisse ist in Abbildung 5.1 zu finden. Zum Einreichungszeitpunkt können die

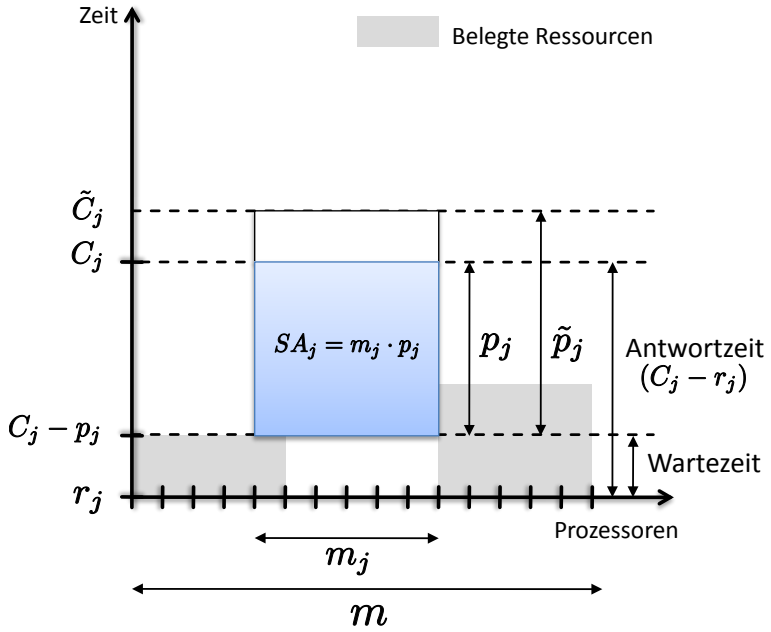


Abbildung 5.1: Zeitliche Verhältnisse innerhalb eines Schedule.

Nutzer optional eine Schätzung der Laufzeit $\tilde{p}_j$ angeben, die dann als maximal angeforderte Laufzeit interpretiert wird. Das heißt, Jobs, deren Laufzeiten unterschätzt sind $(p_j > \tilde{p}_j)$, werden vom System zum Zeitpunkt $\tilde{p}_j$ abgebrochen. Somit kann die Bearbeitungszeit eines Jobs $\tilde{p}_j$ niemals überschritten werden und jeder ins System eingereichte Job wird deshalb in jedem Falle terminiert. Diese Maßnahme ist nötig, um die Blockierung von Ressourcen durch potenziell fehlerhafte Jobs zu verhindern. Dies hat zu Folge, dass Nutzer dazu tendieren, die Laufzeiten ihrer Jobs zu überschätzen, um damit vorzeitigen Abbrüchen entgegenzuwirken, siehe Lee u. a. [111]. Grundsätzlich wäre es möglich, aufgrund der Laufzeitschätzung einen erwarteten Beendigungszeitpunkt $\tilde{C}_j$ nach der Einplanung eines Jobs zu bestimmen und mit

[6] Alternativ wird zum Beispiel von Drozdowski [52] für den Parallelitätsgrad auch $size_j$ verwendet.

dessen Hilfe Schedulingentscheidungen zu verbessern. Die aus den Überschätzungen resultierende allgemein schlechte Qualität der Laufzeitschätzungen macht sie aber nur bedingt zur Verbesserung von Schedulingentscheidungen nutzbar. Im Rahmen dieser Arbeit wird daher bei der späteren Optimierung der Schedulingstrategien im Grid-Kontext auf die Forderung einer Laufzeitschätzung verzichtet.
Abhängigkeiten zwischen den Jobs im Sinne von Workflows werden auch nicht berücksichtigt, da diese auf Parallelrechnern entweder selten zu finden sind oder vom Scheduler nicht unterstützt werden. In heutigen Grids wird die Verarbeitung von Workflows zwar angestrebt, ist aber momentan nur rudimentär umgesetzt. Außerdem sind verlässliche Aufzeichnungen über in Grids eingereichte Workflows bisher nur in sehr geringem Maße verfügbar, da unter anderem bisher kein einheitliches Modell für die Beschreibung von Workflows im Grid-Kontext existiert. Deshalb werden auch hier nur Jobs angenommen, bei denen keine Abhängigkeiten bestehen.
Dieses so beschriebene Maschinen- und Jobmodell wurde im Wesentlichen bereits 1975 von Garey und Graham [80] vorgestellt und wiederholt, zum Beispiel von Hotovy [92] oder Song u. a. [165], als realistisch für existierende Parallelrechnerinstallationen (zum Beispiel bei der IBM SP2) nachgewiesen.

5.3 Schedulingproblem

Aufgrund nicht vorhandener Kenntnis der exakten Laufzeit zum Einreichungszeitpunkt und des fehlenden Wissens über zukünftige Jobeinreichungen handelt es sich bei einem Parallelrechner um ein sogenanntes Online-System. Das resultierende Zuteilungsproblem von Jobs zu Prozessoren ist somit, entsprechend dem vorgestellten Modell, ein Online-Schedulingproblem mit parallelen Jobs und identischen Maschinen. In der Literatur wird dies Problem üblicherweise als *non-clairvoyant online job scheduling* Problem bezeichnet, siehe zum Beispiel Motwani u. a. [127]. Es ist dabei allerdings (wie bei vielen anderen Online-Problemen, siehe Albers [24]) nicht nötig, dass ein Job unmittelbar nach seiner Einreichung allokiert wird. Diese Forderung ist für non-clairvoyant-Probleme wenig sinnvoll, da dies im schlimmsten Falle zu einer sehr schlechten Auslastung und Lastverteilung führen kann. Außerdem wäre ein solches Vorgehen in realen Grids physikalisch nicht realisierbar, da dies den unmittelbaren Zugriff auf entfernt vorhandene Ressourcen benötigen würde.
Formal dargestellt besteht das Schedulingproblem darin, Allokationen S_k für Jobs $j \in \tau_k$ zu $m_j < m_k$ Prozessoren zu finden, sodass diese Jobs ohne Unterbrechung auf ihnen ausgeführt werden. Ein gültiger Schedule erfüllt dabei zunächst die Bedingung in Gleichung 5.1.

$$r_j + p_j \leq C_j(S_k) \; \forall \; j \in \tau_k \tag{5.1}$$

Damit ist sichergestellt, dass alle Einreichungszeitpunkte berücksichtigt werden und kein Job vor seiner Einreichung gestartet wird. Weiterhin dürfen zu jedem Zeitpunkt t nicht mehr als die maximal verfügbaren Prozessoren m_k von den momentan ausgeführten Jobs genutzt werden. Formal muss daher die Bedingung 5.2

$$m_k \geq \sum_{\{j \in \tau_k \mid C_j(S_k) - p_j \leq t < C_j(S_k)\}} m_j \tag{5.2}$$

für alle Zeitpunkte t erfüllt sein. In der von Graham u. a. [83] eingeführten Schreibweise

Maschinenmodell | *Randbedingungen* | *Zielfunktionen*

für Schedulingprobleme kann das hier vorliegende Problem als

$$P_m|m_j, ncv|\sum C_j$$

beschrieben werden. Dabei bezeichnet P_m das parallele identische Maschinenmodell, der Zusatz *ncv* (non-clairvoyant) fehlendes Wissen über zukünftige Einreichungen, m_j die Berücksichtigung von parallelen Jobs und $\sum C_j$ eine mögliche Zielfunktion (siehe Abschnitt 7.4), die–in leichter Abwandlung–in dieser Arbeit vorrangig betrachtet wird.

5.4 Lokales Ressourcen-Management-System (LRMS)

Die Lösung des Online-Problems auf Parallelrechnern wird in realen Installationen durch ein *lokales Ressourcen-Management-System* (LRMS) übernommen. Dies gliedert sich in seiner einfachsten Struktur in drei Komponenten, siehe auch Abbildung 5.2:

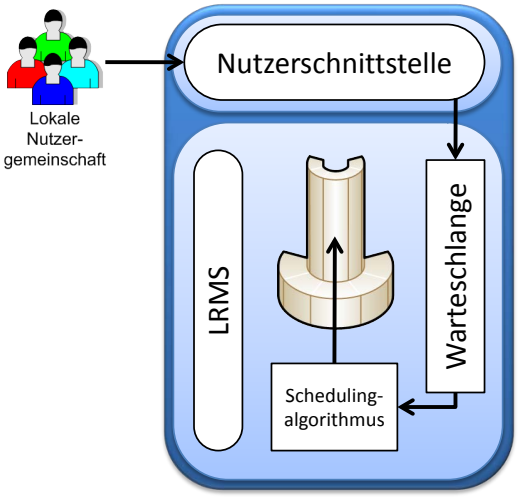


Abbildung 5.2: Aufbau eines lokalen Ressourcen-Management-Systems (LRMS).

Warteschlange (Queue): Alle eingereichten Jobs werden zunächst in die lokale Warteschlange einsortiert, wobei diese grundsätzlich auf einer fest vorgegebenen Ordnung basiert.

Schedulingalgorithmus: Der Schedulingalgorithmus ordnet jeweils einem Job aus der Warteschlange Ressourcen zu, die in die Verwaltungsdomäne des LRMS fallen. Üblicherweise agiert der Algorithmus immer dann, wenn ein Job eingereicht, gestartet oder beendet wird. Die Qualität der erzeugten Schedules hängt natürlich stark von dem Vorgehen des eingesetzten Algorithmus ab. Die gebräuchlichsten Heuristiken werden in Abschnitt 5.4.2 genauer dargestellt.

Schedule: Der Schedule ist der Belegungsplan für Ressourcen über der Zeit. Alle Planungen werden anhand der dort protokollierten Belegungen durchgeführt. Dabei sind Reservierungen und Planung in die Zukunft prinzipiell auch möglich, werden in dieser Arbeit aber nicht weiter betrachtet, da diese Verfahren bisher noch keine weitere Verbreitung gefunden haben, siehe auch Feitelson u. a. [68] oder Ernemann u. a. [1].

5.4.1 Batch-Systeme

Batch-Systeme als praktische Umsetzung lokaler Schedulingsysteme kapseln die Komplexität der zugrunde liegenden Ressourcenverwaltung sowohl für Nutzer als auch für den Administrator des Systems. Sie stellen außerdem eine einheitliche Einreichungsschnittstelle für Jobs zur Verfügung und bieten zusätzlich zumeist Kontroll- und Überwachungsfunktionen (Monitoring) an. Beispiele für solche Systeme sind IBM LoadLeveler, siehe Skovira u. a. [159], Portable Batch System (PBS)/Torque, siehe Henderson [87], oder die Sun Grid Engine. Etsion und Tsafrir [60] geben eine Übersicht der heute verwendeten Batch-Systeme. Die Konfigurationsoptionen moderner Batch-Systeme ermöglichen jedoch auch die Nutzung mehrerer Queues, die Vergabe von Job- und Nutzerprioritäten oder die Einführung von Quoten für einzelne Nutzer, um die Rechenleistung des MPP-Systems beliebig zu verteilen. Da sich die verwalteten MPP-Systeme unterschiedlicher Verwaltungsdomänen in Größe und Aufbau genauso stark unterscheiden können wie die jeweilige Nutzergemeinschaft, können auch die Konfigurationen der Batch-Systeme stark variieren. Ernemann u. a. [1] zeigen anhand von Befragungen, dass in realen Systemen häufig angepasste Lösungen zu finden sind, die zur Umsetzung unterschiedlichster Priorisierungen und Servicezusicherungen genutzt werden. Es wird dort weiterhin deutlich, dass die genaue Konfiguration eines LRMS fast immer der Geheimhaltung unterliegt und daher nicht öffentlich verfügbar ist.

Die in dieser Arbeit entwickelten Methoden sind jedoch generell für jegliche LRMS-Konfiguration anwendbar. Die Ergebnisse und die Güte der erreichbaren Schedulinglösungen können jedoch leicht abweichen, da sie natürlich von den verwendeten LRMS-Konfigurationen abhängen. Für die hier vorgenommenen Untersuchungen wird deshalb auf allgemein verwendbare Standardheuristiken innerhalb des LRMS zurückgegriffen, wie sie heute immer noch vielfach Verwendung finden, siehe Feitelson u. a. [68].

5.4.2 Häufig verwendete Standardheuristiken

Schedulingalgorithmen sind dafür verantwortlich, gültige Schedules für die eingereichten Jobs zu erzeugen. Sie sind aber immer im Hinblick auf eine oder mehrere Zielfunktionen entwickelt, sodass auch nur für bestimmte vorher berücksichtigte Kriterien gute Schedules garantiert werden können, siehe Abschnitt 7.4. Für die hier betrachteten Online-Schedulingprobleme haben sich heuristische Lösungsverfahren etabliert.

An diese Heuristiken werden die Anforderungen gestellt, sowohl den Aufwand für das Design der Algorithmen als auch den Berechnungsaufwand zur Bestimmung einer Schedulingentscheidung möglichst gering zu halten. Dabei werden Schedulingalgorithmen auch immer für einen bestimmten Verwendungszweck entworfen, wobei vorhandene Standardheuristiken als Basiskomponenten auch in komplexeren Strategien eine Wiederverwendung finden können, siehe zum Beipiel Franke u. a. [8]. Ein strukturiertes Vorgehenskonzept zum Entwurf von Schedulingalgorithmen für Parallelrechner ist von Krallmann u. a. [108] vorgeschlagen worden. In dieser Arbeit werden jedoch keine neuen LRMS-Algorithmen entworfen, sondern es wird auf Standardlösungen zurückgegriffen.

Dabei haben sich drei einfache Heuristiken als sehr effektiv erwiesen und finden deshalb (zum Teil in nur leichten Abwandlungen) Verwendung in vielen real eingesetzten Schedulingsystemen, siehe Ernemann u. a. [1]:

First Come, First Served (FCFS) führt den ersten, das heißt den am längsten wartenden Job in der Warteschlange aus, sobald genügend freie Prozessoren verfügbar sind. Im Worst-case resultiert diese Heuristik in einer sehr geringen Auslastung, siehe zum Beispiel Turek u. a. [175], wobei jedoch in der Praxis FCFS zu akzeptablen Ergebnissen, siehe zum Beispiel Grimme u. a. [15], führt. Auch in dieser Arbeit wird FCFS als bevorzugter, lokaler Schedulingalgorithmus verwendet, da er als Beispiel für eine einfache und vielfach eingesetzte Heuristik dient und mit einem Minimum an Informationen zum Treffen von Entscheidungen auskommt. Eine Analyse der erreichten Schedulingqualität auf Basis von realen Arbeitslastaufzeichnungen findet sich in Abschnitt 7.1.

Extensible Argonne Scheduling System (EASY) stellt eine mögliche Variante des sogenannten *Backfilling* dar, bei dem es erlaubt ist, später eingereichte Jobs für die Ausführung vorzuziehen. Backfilling-Varianten benötigen zusätzliche Informationen in Form der Laufzeitschätzung $\tilde{p}_j$, siehe Abschnitt 5.2.2, eines Jobs j. Sei t der aktuelle Zeitpunkt, j der aktuell erste Job in der Warteschlange und $T_j = t$ der Zeitpunkt, zu dem j, bestimmt auf Basis der gegebenen Laufzeitschätzungen, im aktuellen Schedule frühestmöglich gestartet werden kann. Wenn dieses aktuell nicht der Fall ist, also $T_j > t$ gilt, dann wird der erste Job i mit Laufzeitschätzung $\tilde{p}_i$ aus der Warteschlange gestartet, für den genügend freie Prozessoren verfügbar sind und dessen Ausführung den Start von j nicht weiter verzögert. Es muss also die Bedingung $t + \tilde{p}_i \leq T_j$ erfüllt sein (Backfilling-Bedingung). Eine genaue Beschreibung und Analyse ist sowohl bei Lifka [115] als auch bei Feitelson und Weil [70] zu finden.

LIST-Scheduling repräsentiert den allgemeinsten Scheduling-Ansatz: Falls der erste Job in der Warteschlange nicht unverzüglich ausgeführt werden kann und nicht alle Prozessoren belegt sind, so wird die Warteschlange iterativ vom Kopf zum Ende durchlaufen, bis der erste Job gefunden wird, der die noch vorhandenen Prozessoren nutzen kann, oder das Ende der Warteschlange erreicht ist. Existiert ein solcher Job, so wird dieser sofort gestartet. Dabei werden jegliche mögliche Verzögerungen von weiter vorn stehenden Jobs in der Warteschlange in Kauf genommen. Dieses Verfahren wurde bereits 1969 von Graham [82] eingeführt und detailliert untersucht. Dabei kann LIST mit jeder Form von Listensortierung angewendet werden.

Diese hier vorgestellte Architektur und die einfachen Methoden des Ressourcen-Managements auf Parallelrechnern können nun genutzt werden, um ein Modell eines Computational Grid aufzubauen.

6
Job-Scheduling in Computational Grids

SEITDEM IM Jahre 1992 von Smarr und Catlett [161] der Begriff *Meta-Computing* geprägt wurde, gab es zahlreiche Forschungsanstrengungen im Hinblick auf die Konzeption von Architekturen zur gemeinsamen Nutzung weltweit verteilter Ressourcen. Heute ist dieser Begriff durch den von Foster und Kesselman [74] geprägten Terminus *Grid-Computing* verdrängt worden. Aber auch dieser Begriff ist seitdem unterschiedlich interpretiert worden und hat eine Reihe von Metamorphosen durchgemacht, siehe zum Beispiel Altmann u. a. [26]. Jede Auslegung des Begriffes drückt dabei ganz bestimmte Anforderungen von Commmunities aus, die diese jeweils durch Grid-Computing erfüllt sehen wollen. Unter den verschiedenen Modellen ist das bislang am weitesten verbreitete jedoch ein *Computational Grid*. Dies repräsentiert eine Form von parallelverteilten Systemen, die auf einem Zusammenschluss regionaler, nationaler und globaler (heterogener) Rechnersysteme unter Einbeziehung von Daten und weiteren Ressourcen basiert. Dabei sind die Teilnehmer durch ein Hochleistungs-Kommunikationsnetz miteinander verbunden. Es erlaubt somit die Aggregation von autonomen Ressourcen in einer dynamischen und sich ändernden Umgebung. Die im vorigen Kapitel eingeführte Einheit aus Nutzergemeinschaft, Ressourcen und koordinierendem LRMS stellt somit einen elementaren Baustein des Computational Grids dar. Wie schon zu Anfang erwähnt, wird hier der etwas organisatorisch abstrakte Begriff von virtueller Organisation und Community nun auf mehr technische Weise modelliert.

Hier soll aber auf eine Beschreibung aller momentan umgesetzten Architekturen im Sinne einer vollständigen Taxonomie verzichtet und auf die Arbeiten von Berman u. a. [30] verwiesen werden. Da sich die Entwicklungen in dieser Arbeit ausschließlich auf das Job-Scheduling und Ressourcen-Management in Computational Grids konzentrieren, ist es vielmehr notwendig, gemeinsame Anforderungen an die Architekturen im Hinblick auf einen effizienten Jobaustausch zu identifizieren. Ziel ist dabei die Erstellung eines einfachen, aber realistischen Modells für eine allgemeingültige und zukunftsorientierte Grid-Schedulingarchitektur. Diese soll die angesprochene unterschiedliche Komplexität von Grid-Architekturen möglichst gut abbilden und gleichzeitig so einfach sein, dass die erstellten Schedulingkonzepte und -algorithmen auf Basis von realistischen Eingangsdaten evaluiert werden können. Gemeinsam ist allen Grid-Organisationsformen, dass lokal vorhandenen Ressourcen durch eine zusätzlich eingeführte Koordinationsschicht, der sogenannten Middleware, über Schnittstellen in einem größeren Kontext verfügbar gemacht werden. Dieser globale Kontext der Interaktion bleibt der ausschließlich lokal agierenden Nutzergemeinschaft dabei vollständig verschlossen, sodass der Zugriff aus Nutzersicht vollkommen transparent geschieht. Die Nutzergemeinschaft formuliert gegenüber ihrer lokalen Grid-Schnittstelle lediglich Ressourcenanforderungen, die dann von der Infrastruktur autonom umgesetzt werden. Aufgrund der grundsätzlichen Analogie wird der Grid-Gedanke auch gern mit dem Stromnetz *(power-*

grid) verglichen, bei dem ebenso ein transparenter Zugriff auf die Ressource Energie realisiert ist.

Bevor nun die unterschiedlichen Grid-Schedulingarchitekturen genauer spezifiziert werden, ist es nötig, den Begriff der „Nutzergemeinschaft“ *(community)* genauer zu fassen. In dieser Arbeit werden unter einer Nutzergemeinschaft[7] die Zugehörigen einer virtuellen Organisation verstanden. Innerhalb der D-Grid Initiative [176] wird zum Beispiel eine Grid-Lösung für die deutsche Klimaforschung erstellt [84], die es dann der daraus entstehenden Klimaorganisation erlaubt, Ressourcen transparent zu nutzen. Dabei hat eine virtuelle Organisation auch immer ganz spezielle Anforderungen und Anwendungen, weshalb jede Nutzergemeinschaft diesbezüglich ihre ganz eigenen Charakteristiken besitzt.

Eine virtuelle Organisation ist ein Zusammenschluss von Personen, Unternehmen und realen Organisationen, die durchaus veränderlich ist und auch nur vorübergehend bestehen kann. Das Grid ermöglicht nun die Interoperabilität innerhalb einer dynamischen und heterogenen Gemeinschaft von virtuellen Organisationen, bei der jede an einem Grid beteiligte virtuelle Organisation Ressourcen anbieten und über die Nutzung fremder Ressourcen mit anderen virtuellen Organisationen des Grids verhandeln kann. Ein Grid ist prinzipiell heterogen, da die an ihm beteiligten virtuellen Organisationen unterschiedliche Hardware-, Netzwerk- , Basis- und Anwendungssoftware verwenden können. Des Weiteren ist die Beziehung zwischen virtuellen Organisationen von unterschiedlichem Vertrauen geprägt, und der Zugang zu Ressourcen unterliegt unterschiedlichen Bedingungen und Restriktionen. Grid-Computing soll den virtuellen Organisationen ermöglichen, Ressourcen wie Daten, Software, Rechner und Netzwerke mit dem Ziel zu teilen, Probleme kooperativ zu lösen.

Ein weiterer im Folgenden häufig verwendeter Begriff ist die *Site* (engl. für Standort). Eine Site beinhaltet alle Ressourcen, die von einer virtuellen Organisation verwaltet werden und unter ihrer Kontrolle stehen. Sites sind somit die elementare Einheit in einem Grid und führen eine Kapselung von mehreren MPP-Systemen oder Clustern ein. Unter der Annahme von homogenen Rechenressourcen, siehe Abschnitt 5.2, ist eine Site durch einen einzigen Parallelrechner darstellbar. Eine Site ist dabei immer genau einer virtuellen Organisation zugeordnet, die die alleinige Verfügungsgewalt für die Ressourcen hat.

Innerhalb eines Grids ist die Kontrolle über Ressourcen einer Site deshalb auch immer lokal: Eine virtuelle Organisation legt selbst fest, wann, wo, wofür und unter welchen Bedingungen (zum Beispiel zu welchen Kosten) Ressourcen anderen Teilnehmern des Grids zur Verfügung gestellt werden. Informationen über den Status der lokalen Ressourcen werden daher auch sehr restriktiv behandelt und selbst bei Kooperation in einem Grid den Partnern üblicherweise nicht zur Verfügung gestellt. Folglich muss ein Grid unterschiedliche lokale Sicherheitslösungen *(authentication/authorization)* integrieren, um die Trennung innerhalb der Grid-Partner aufrecht zu erhalten. Weiterhin müssen Lösungen zum Finden von Ressourcen *(resource discovery)* vorhanden sein, damit sich die Infrastruktur auch in dynamischen Umgebungen aufrecht erhalten lässt. Für künftige Grid-Lösungen ist es außerdem wichtig, für Dienstqualität *(quality of service, QoS)*, vertragliche Garantien *(service level agreements, SLA)* und Abrechnung *(accounting)* Unterstützung zu bieten, siehe auch Tonellotto u. a. [171]. Die Verwaltung der veränderbaren Zugriffsregelungen *(policies)*, die die Bedingungen der Ressourcennutzung spezifizieren, ist eine weitere Kernaufgabe eines Grids. Letzteres muss für

[7] Alle Beteiligten mit den gleichen (Anwendungs-)Interessen wie Datenanbieter, Forschergruppen und auch Softwarehersteller werden im üblichen Grid-Sprachgebrauch als *Community* bezeichnet.

den Aufbau eines leistungsfähigen Ressourcen-Managements integraler Bestandteil eines im Folgenden erläuterten Grid-Ressourcen-Management-System sein.

6.1 Grid-Ressourcen-Management-System (GRMS)

Obwohl der Begriff Computational Grid, wie oben beschrieben, weit mehr impliziert als nur einen Verbund aus MPP- und Cluster-Systemen, ist diese Form der Föderation doch die bisher am weitesten entwickelte. Zudem umfasst der Begriff des Ressourcen-Managements weitaus mehr als reines Job-Scheduling und beinhaltet auch Daten-Scheduling, Workflow-Scheduling und die Co-Allokation von angeforderten Ressourcen. Wie bereits in Abschnitt 5.2.2 beschrieben, wird sich in dieser Arbeit auf die reine Jobzuweisung innerhalb eines Grids beschränkt, da dies zunächst die wichtigste Voraussetzung für ein effizientes Ressourcen-Management darstellt. Die dazu möglichen Schedulingarchitekturen werden hier nun genauer beschrieben.
Das Job-Scheduling in Grids wird dazu pro Site in zwei Ebenen unterteilt: Das bereits zuvor beschriebene LRMS auf rein lokaler Ebene kapselt die Ressourcen einer virtuellen Organisation und realisiert lokale Schedulingregeln wie Gruppenpriorisierungen oder spezielle Präferenzen des Betreibers. Die zweite, hier neu eingeführte Ebene ist das sogenannte *Grid-Ressourcen-Management-System* (GRMS). Dies ist verantwortlich für die Interaktion mit anderen Grid-Sites und kann prinzipiell Funktionen wie Ressourcensuche, Verhandlungen und natürlich die Migration von Jobs zu anderen Sites umsetzen. Für die strukturelle Anlage dieser zwei Ebenen gibt es Konzepte, die grob in *hierarchische* und *dezentrale* Schedulingarchitekturen klassifizierbar sind.

6.1.1 Hierarchische Grid-Schedulingarchitektur

Hierarchische Schedulingarchitekturen nutzen einen zentralen Scheduler, oft auch Meta-Scheduler genannt, der die Jobs aller Nutzer eines Grid Site-übergreifend und unabhängig von ihrer originären Zugehörigkeit akzeptiert. Der Meta-Scheduler stellt somit für die Nutzer die einzige Einreichungsschnittstelle zum Grid dar, siehe Abbildung 6.1. Er hat nun direkten Zugriff auf die von ihm verwalteten Sites und kann unmittelbar mit deren LRMS interagieren. Die Verteilung der Jobs zu den unterschiedlichen Sites wird zentral vom Meta-Scheduler durchgeführt. Dadurch bleibt zwar die traditionell eingeführte Trennung zwischen Gridebene und lokaler Ressourcenverwaltung bestehen, aber die Autonomie der einzelnen Sites wird aufgehoben. Im klassischen Meta-Scheduling-Modell können Nutzer ausschließlich über den Meta-Scheduler ihre Jobs einreichen und eine direkte Nutzung von lokalen Ressourcen ist nicht vorgesehen. Eine Erweiterung dieses Konzepts sieht die Möglichkeit von mehreren Meta-Scheduling-Ebenen vor, die wiederum hierarchisch die Jobs zu den Ressourcen delegieren. Dabei dient aber auch nur die oberste Ebene als Einreichungsschnittstelle, während die unterliegenden Scheduler Teile eines Grids hierarchisch verwalten.
Dieser Ansatz wurde bisher nur sehr wenig empirisch untersucht: So zeigen zum Beispiel Ernemann u. a. [56], dass sich Antwortzeiten für Jobs durch einen derartigen Ansatz mit einer einfachen statischen Schedulingstrategie bereits deutlich verbessern lassen. Dazu benötigt der Meta-Scheduler aber die Einsicht und die Kontrolle über alle von ihm verwalteten Ressourcen. Weiterhin untersuchen Franke u. a. [77] ebenfalls eine hierarchische Schedulingstruktur und vergleichen diese mit alternativen (auch den später vorgestellten dezentralen) Architekturen.

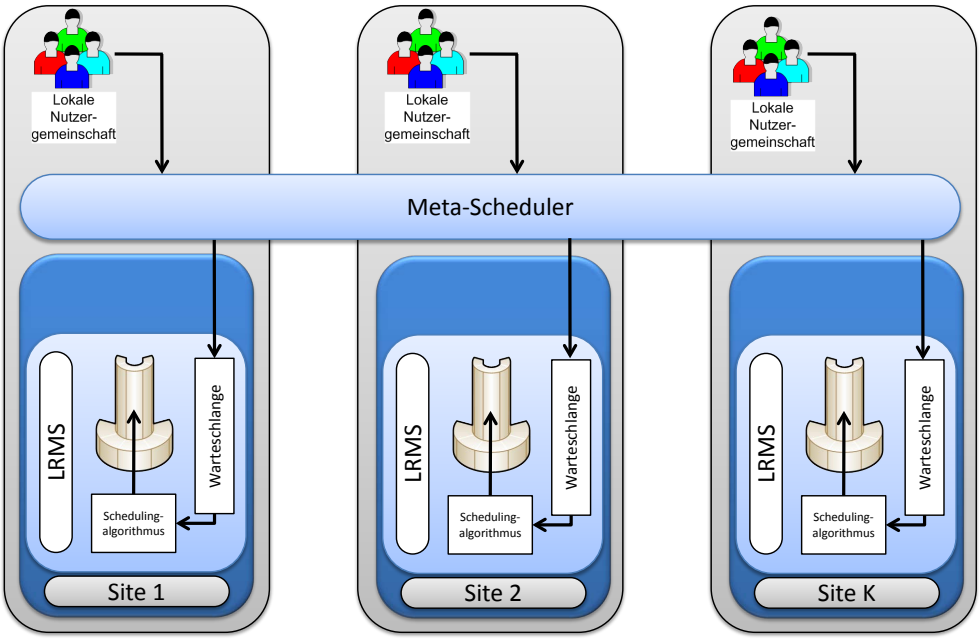


Abbildung 6.1: Hierarchische Grid-Schedulingarchitektur mit drei Sites und einem Meta-Scheduler.

Der Vorteil dieses Ansatzes liegt zum einen in der Möglichkeit, unterschiedliche Schedulingstrategien für den Meta-Scheduler und die lokalen LRMS-Scheduler zu realisieren. Bei einer geschachtelten Meta-Scheduler-Struktur können auch weitere Strategien für jede Ebene eingeführt werden. Dies erlaubt eine strikte Trennung zwischen Grid-Scheduling und LRMS-Scheduling. Aufgrund seiner sehr einfachen Struktur bauen nahezu alle aktuell realisierten Grid-Middleware-Lösungen wie zum Beispiel UNICORE [59] oder die Lösung des EGEE-Projekts [27], siehe auch Abschnitt 6.2, auf diesem Modell auf. Weiterhin stellt dieser Ansatz die übliche Lösung für die an eine spezielle Nutzergemeinschaft angepasste Grid-Scheduling-Infrastruktur dar, siehe Freitag u. a. [79].

Die zahlreichen Nachteile dieser Architektur liegen allerdings in der zentralistischen Struktur begründet. Von Natur aus ist ein derartiges System schlecht skalierbar und der Meta-Scheduler stellt einen „Single Point of Failiure“ dar, der bei Ausfall die gesamte Job-Scheduling-Infrastruktur handlungsunfähig macht. Weiterhin ist es für eine global effiziente Allokation und Verteilung der Arbeitslast notwendig, dass jede Site die Kontrolle über ihre lokalen Ressourcen in die Verantwortung des Meta-Schedulers übergibt. Wie schon Ernemann u. a. [56] zeigen, ist das effiziente Scheduling nur bei Aufgabe der Autonomie möglich. Die administrative Kontrolle wird exklusiv vom Meta-Scheduler übernommen, der nur auf diese Art und Weise eine akzeptable Servicequalität erreichen kann: Im Falle eines hybriden Systems, bei dem lokale Nutzergemeinschaften auch direkt Jobs einreichen können, bestünde für den Meta-Scheduler keine Planungssicherheit. Es könnten nämlich dann immer unvorhergesehen spontane Jobs die eingeplanten Ressourcen besetzen, die auf globaler Ebene dem System nicht bekannt waren. In diesem Falle degeneriert die globale Ressourcenzuordnung zu einer reinen „Best Effort“-Strategie.

In realen Systemen stellt aber nicht nur die fehlende Ausfallsicherheit, sondern auch im Hinblick auf eine zunehmende Konkurrenz zwischen Anbietern von Rechenressourcen der Verlust der Entscheidungskontrolle ein Hindernis bei der Umsetzung einer derartigen Architektur dar, siehe auch Abschnitt 6.3.3.

6.1.2 Dezentrale Grid-Schedulingarchitektur

Ein Konzept, das gerade die oben genannten Nachteile beseitigt, setzt auf eine vollständig dezentrale Grid-Schedulingarchitektur, siehe Abbildung 6.2, im Sinne einer Peer-to-Peer Struktur. Bei diesem Ansatz ist keine zentral koordinierende Instanz nötig und die Grid-

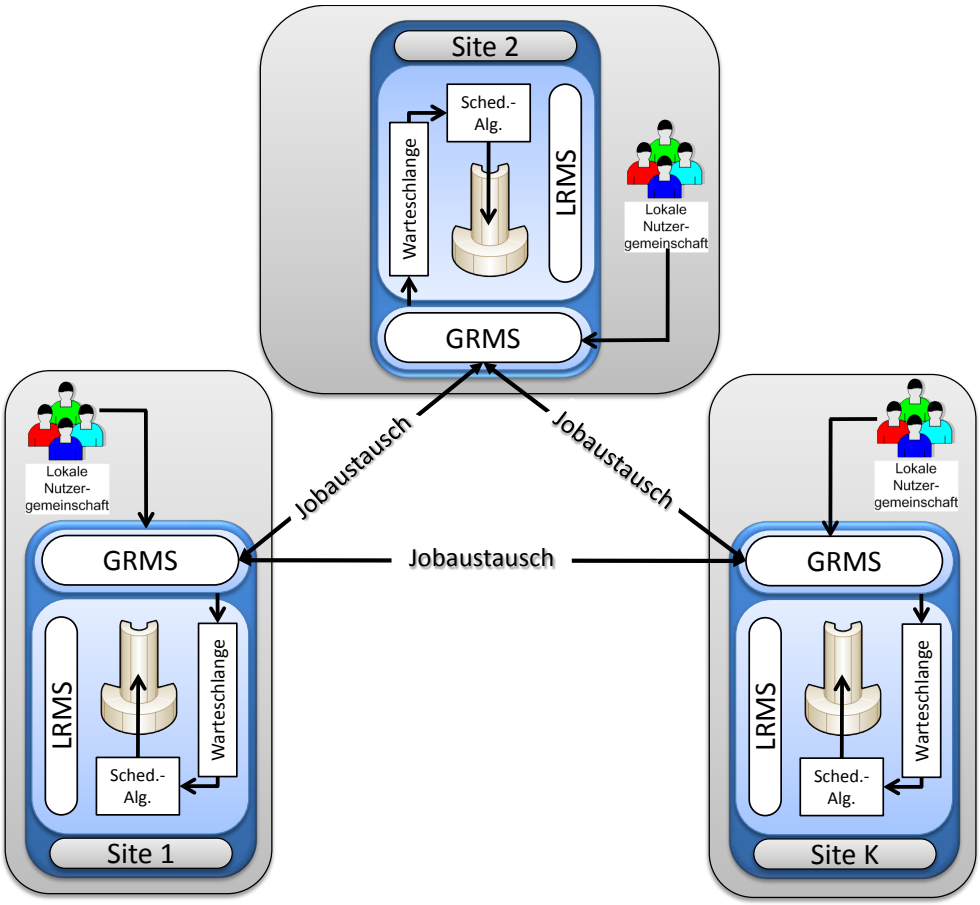


Abbildung 6.2: Dezentrale Grid-Schedulingarchitektur mit drei Sites und jeweils dezentralem Grid-Ressourcen-Management-System (GRMS).

Scheduler interagieren in entweder *direkter* oder *indirekter* Weise miteinander. Die jeweiligen Sites fungieren wiederum als Einreichungsschnittstelle für ihre Nutzergemeinschaften. Dabei wird die bisher vorhandene LRMS-Schicht um eine weitere GRMS-Schicht ergänzt, die dann die Koordinationsaufgaben und die Kommunikation mit weiteren Partnern übernimmt. Eingereichte Jobs werden entweder direkt an das unterliegende LRMS weitergereicht oder, durch Anfrage und Verhandlungsmechanismen gesteuert, zu anderen Sites migriert und dort zur Ausführung gebracht. Grimme u. a. [21] zeigen, dass sich aus dem Peer-

to-Peer Computing bekannte Algorithmen auch vorteilhaft auf ähnliche Strukturen in Grids anwenden lasse.
Wie schon erwähnt, kann die Kommunikation zwischen den verteilten Sites generell auf zwei Arten geschehen: Bei der *direkten Kommunikation* sind Scheduler in der Lage, Jobs unmittelbar untereinander anzubieten oder anzunehmen. Dazu sind Mechanismen zum Auffinden anderer Sites und Mechanismen zum Verwalten aktueller Listen über verfügbare Austauschpartner notwendig. Wenn die Migration eines Jobs zu einer bestimmten Site nicht möglich ist, kann der Scheduler entscheiden, mit einem anderen bekannten Partner um die Ausführung seines Jobs in Verhandlung zu treten.
Bei der *indirekten Kommunikation* wird der Austausch von Jobs durch eine gemeinsame Plattform unterstützt, siehe auch Kapitel 8, auf der Sites Jobs zur Migration anbieten können. Andere Grid-Scheduler können sich nun nach Belieben an dieser Plattform bedienen und Jobs auf ihren Ressourcen zur Ausführung bringen. Obwohl diese Komponente zentral im Grid verfügbar ist, dient sie nur als passives Austauschmedium und die Verantwortlichkeiten bleiben dezentral bei der jeweiligen Site erhalten. Die Entscheidung, welche Jobs nun angeboten und welche angenommen werden, obliegt weiterhin jeder autonomen Site. Sollte die zentrale Plattform ausfallen, bricht zwar der Jobaustausch zusammen, aber die Nutzergemeinschaften können immer noch ihre lokalen Ressourcen ohne Grid-Interaktion nutzen.
Für die dezentrale Schedulingarchitektur existieren bereits einige Forschungsarbeiten, die auch empirische Untersuchungen mit einschließen. So wird in der Arbeit von Lu u. a. [117] beispielsweise der Jobaustausch zwischen Sites anhand eines Lastverteilungsproblems *(load balancing)* modelliert. Hierbei wird für jede Site ein sogenannter Loadindex berechnet, der die momentane Arbeitslast an jeder Site widerspiegelt. Die Loadindizes werden nicht zentral verwaltet, sondern als zusätzliche Informationen an zwischen den Sites wandernde Jobs geheftet oder nach Ablauf einer gewissen Zeitdauer ohne Jobaustausch abgefragt. Bei diesem Ansatz wird die Autonomie der einzelnen Sites eingeschränkt, da sie keine Möglichkeit besitzen, übergebene Jobs abzulehnen. Das Schedulingverfahren basiert mehr auf einem Kollektivgedanken, bei dem es hauptsächlich um die Umverteilung der Arbeitslast zwischen den Sites geht und weniger darum, dass jede der Sites aus ihrer Sicht von dem Jobaustausch profitiert.
Andere dezentrale Ansätze setzen auf die Modellierung des Grids als ökonomisches Modell. So publiziert im Modell von Buyya u. a. [38] jeder Ressourcenanbieter Preise für die Nutzung seiner Ressourcen in Abhängigkeit von Tageszeiten oder bestimmten lokalen Lastsituationen. Der Ressourcennutzer wiederum spezifiziert, in welchem Zeitrahmen und zu welchen Kosten Jobs bearbeitet werden sollen. Die anschließenden Verhandlungen, die automatisch durch Komponenten der Grid-Middleware durchgeführt werden, dienen dazu, den Job im Rahmen der Bedingungen und zum bestmöglichen Preis auszuführen. Ein ganz ähnlicher Ansatz mit einer mehr auf das reine Scheduling fokussierten Kostenfunktion wurde von Ernemann u. a. [57] diskutiert. All diese ökonomischen Modelle bieten eine Vielzahl von einstellbaren Betreiber- und Nutzerpräferenzen und basieren auf der Idee, dass das Zusammenspiel von Angebot und Nachfrage zu einem Marktgleichgewicht führen wird. Die Optimierung des Jobaustauschs und der dazu verwendeten Kostenfunktionen steht jedoch weder im Vordergrund, noch wurde sie ausreichend untersucht. Doch ist gerade die Optimierung des Jobaustauschs von Interesse, da die primäre Motivation für Grid-Computing in der Überwindung lokal limitierter Ressourcen liegt, mit dem Ziel, Jobs effizienter abzuarbeiten und gleichzeitig die vorhandenen Ressourcen besser ausnutzen zu können. Dies könnte über speziell angepasste Kostenfunktionen zwar prinzipiell auch erreicht werden,

aber deren Formulierung und nötige Optimierung gestaltet sich in vielen Fällen als extrem aufwendig, siehe Franke u. a. [77], weshalb bei ökonomischen Ansätzen zumeist nur von Hand eingestellte Kostenfunktionen verwendet werden.
Zusammenfassend ist festzustellen, dass die Vorteile einer dezentralen Schedulingarchitektur zum einen, unabhängig vom jeweiligen Kommunikationsmodell, in einer viel höheren Fehlertoleranz liegen, da der Ausfall einer GRMS-Komponente oder auch einer gesamten Site zwar die Gesamteffizienz des Grids verschlechtern kann, aber nie zu einem Totalausfall des Systems führt. Zum anderen wird durch eine solche Architektur aus demselben Grunde natürlich eine deutlich bessere Skalierbarkeit erzielt. Für die Zukunft ist aus Sicht des Autors deshalb davon auszugehen, dass Grid-Computing sich nur mittels dezentraler Strukturen und unter Wahrung der lokalen Autonomie im großen Stil realisieren lässt.
Der Nachteil dezentraler Strukturen liegt dabei allerdings genau in dem hauptsächlichen Vorteil: Das Fehlen einer globalen Grid-Systemsicht macht das Scheduling deutlich komplexer. Die Entscheidung über die Annahme von angebotenen Jobs oder die Abgabe lokal eingereichter Jobs muss *ohne* konkrete Informationen über den Status anderer Partner gefällt werden können. Alle oben vorgestellten dezentralen Schedulingansätze kommen dieser Forderung nicht nach und sind mit dieser Einschränkung nicht vereinbar. Selbstverständlich können solche Informationen über gewisse Mechanismen für alle Partner aktuell verfügbar gemacht werden, aber typischerweise werden Statusinformationen aus politischem Kalkül lieber vertraulich behandelt, siehe auch Abschnitt 6.3.3. Deshalb ist es notwendig, zur Umsetzung einer solchen dezentralen Schedulingarchitektur besonderen Wert auf das Design leistungsfähiger Entscheidungsstrategien für das GRMS zu legen.

6.2 Eigenschaften existierender Grid-Schedulingsysteme

Alle bisher in der Praxis eingesetzten Softwaresysteme für das Ressourcen-Management in Grids basieren auf dem in Abschnitt 6.1.2 beschriebenen hierarchischen Meta-Scheduling-Ansatz. Unter der Vielzahl von zumeist projektspezifischen Entwicklungen haben einige Systeme, die auch eine allgemein verwendbare Infrastruktur beinhalten, eine weitere Verbreitung gefunden. Dabei hat sich hauptsächlich das *Globus Toolkit* [73] als unterliegende Middleware-Lösung durchgesetzt.
Das Projekt *GridWay* [94] zum Beispiel realisiert einen leichtgewichtigen Meta-Scheduler auf Basis des Globus Toolkit. GridWay bietet eine Einreichungsschnittstelle auf Basis des DRMAA-Standards [172] für Jobs und das Job-Management sowie einfache Schedulingfunktionalität. Weiterhin werden auch die Definition und die anschließende Behandlung von Workflows unterstützt. Die Umsetzung von ausgefeilten Algorithmen steht allerdings nicht im Fokus dieser Entwicklung. Vielmehr wird unterstützt, anhand von gegebenen Anforderungsbeschreibungen passende Ressourcen auszuwählen und auf diesen dann die entsprechenden Jobs zur Ausführung zu bringen.
Ein anders etabliertes System ist der für das sogenannte VIOLA-Projekt (Vertically Integrated Optical Testbed for Large Applications) entstandene *Meta-Scheduling Service* (MSS) [86]. Dieser erlaubt die simultane Reservierung von Rechenzeiten auf mehreren Ressourcen sowie die Reservierung von Bandbreite zwischen den Ressourcen durch den Anwender und verhindert Überschneidungen und dadurch entstehende Engpässe im Rechnerverbund. Der Meta-Scheduler greift zur Reservierung der Rechenressourcen auf das jeweilige LRMS zu und reserviert die Netzkapazitäten. Dabei wird dieses System bevorzugt in die UNICORE-

Middleware integriert. Das unterliegende System bleibt somit autonom und es gibt keine Einschränkungen bezüglich administrativer Präferenzen. Allerdings können Ressourcen gleichzeitig sowohl lokal als auch im Grid-Kontext genutzt werden. Durch die Integration des *Intelligent Scheduling System Model* (ISS), siehe Cristiano u. a. [47], werden auch die Umsetzung komplexerer Schedulingverfahren sowie die Behandlung von Workflows unterstützt.
Obwohl zwischen den vorgestellten Lösungen, die nur eine Auswahl der verbreitetsten Lösungen ist, teilweise große Unterschiede in ihrer Realisierung und in Bezug auf ihre jeweilige Ausrichtung hin bestehen, können dennoch die grundlegenden Gemeinsamkeiten extrahiert werden, die alle aus dem hierarchischen Ansatz resultieren:

1. Alle vorgestellten Systeme nehmen nur einen einzigen Einstiegspunkt (also eine Einreichungschnittstelle) für das von ihnen verwaltete Grid an. Damit hat der Grid-Scheduler auch automatisch die Kontrolle über alle ins Grid eingereichten Jobs.

2. Alle Scheduler setzen die vollständige Kontrolle über die von ihnen verwalteten Ressourcen voraus und haben unbeschränkte Einsicht in den aktuellen Status aller Komponenten.

Damit unterscheidet sich das hier dargestellte Szenario kaum von einem reinen Parallelrechner. Auch dort werden alle Ressourcen zentral von einem Batch-System mit entsprechendem Scheduler verwaltet. Das gleiche gilt für einen klassischen Cloud-Computing-Ansatz [75], bei dem ebenfalls alle Ressourcen unter einer zentralen Verwaltung stehen. In dieser Arbeit wird deshalb für die Schedulingarchitektur innerhalb einer Verwaltungsdomäne als Modell ein Parallelrechner mit entsprechendem Ressourcen-Management-System angenommen.
In einem Computational Grid werden diese Verwaltungsdomänen, also entweder Parallelrechner, hierarchisch gesteuerte Community Grids oder auch Cloud-Computing-Systeme, als unabhängige autonome Teilnehmer miteinander interagieren. In zukünftigen Systemen ist daher anzunehmen, dass

1. es mehrere Einstiegspunkte in ein Grid gibt, die zumeist von den lokalen Ressourcenanbietern und damit Grid-Sites bereitgestellt werden,

2. das Ressourcen-Management autark und dezentral durch Verhandlung zwischen den Anbietern der Ressourcen umgesetzt wird und

3. die LRMS-spezifischen Einstellungen weiterhin Bestand haben und von den jeweiligen Administrationen auch nach Bedarf angepasst werden. Eine Anpassung des LRMS an die Bedürfnisse des Grids wird die Ausnahme bleiben.

Das Modell eines Grids, das diesen für die Zukunft realistisch erscheinenden Anforderungen gerecht wird, ist im folgenden Abschnitt erklärt.

6.3 Modell eines Computational Grids

In diesem Abschnitt wird nun genau das in der vorliegenden Arbeit angenommene Modell eines Computational Grids spezifiziert. Es folgt im Wesentlichen der in Abschnitt 6.1.2 und in Abbildung 6.2 dargestellten Architektur. Dabei stellen alle hier gemachten Modellannahmen eine Erweiterung der schon in Abschnitt 5.2 eingeführten Modellüberlegungen dar.

6.3.1 Site-, Maschinen- und Jobmodell

In diesem Modell besteht ein Grid dabei aus einem losen Verbund von $|K|$ unabhängigen Sites $k \in K$. Jede dieser Sites hat die Verwaltungshoheit über eine gewisse Menge von Prozessoren, die durch einen Verbund von beliebigen Ressourcen bereitgestellt werden. Im Modell wird aber davon ausgegangen, dass die Ressourcen vollkommen homogen sind (P_m Modell) und eine unendliche schnelle Netzwerkverbindung zwischen ihnen besteht. Somit ist dieses Modell identisch mit dem bereits eingeführten Modell eines Parallelrechners; die Ressourcen einer Site k werden folglich durch einen Parallelrechner mit m_k Prozessoren abgebildet. Oftmals wird von Modellen gefordert, dass in Abhängigkeit vom Ausführungsort die Programme aufgrund der unterschiedlichen lokalen Prozessorarchitekturen auch variable Laufzeiten haben. Es ist aber nur unter größten Aufwendungen, die ein eigenes Forschungsfeld bemühen würden, möglich, ein derartiges Modell aufzustellen, da Computerprogramme in der Regel nicht einfach linear in ihrer Ausführungszeit auf unterschiedlichen Prozessorarchitekturen skalieren. Es hängt sehr von der Struktur des betrachteten Programms ab, weshalb komplexe Benchmarks notwendig sind, um die Auswirkungen von veränderten Prozessorarchitekturen auf die tatsächliche Ausführungszeit zu berücksichtigen. Da dies hier nicht geleistet werden kann und aufgrund der Ähnlichkeit aktueller Installationen dadurch keine allzu große Vereinfachung gemacht wird, werden hier alle an einem Grid beteiligten Sites als identische Maschinen betrachtet. Weiterhin sind die angenommenen Jobs identisch mit den bereits in Abschnitt 5.2.2 diskutierten starren parallelen Batch-Jobs, wie sie üblicherweise auf Parallelrechnern eingereicht werden. Allerdings ist es nun nicht, wie zum Beispiel von Ernemann u. a. [55] untersucht, möglich, Jobs über die Grenzen einer Site hinweg aufzuteilen. Dieses sogenannte *Multi-Site-Computing* wird theoretisch durch Meta-Schedulingarchitekturen unterstützt, bei denen zentral Informationen über alle aktuellen Site-Belegungen verfügbar sind. Wird für das Aufteilen von Jobs über mehrere Sites kein zusätzliches Modell für den zu erwartenden Kommunikationsaufwand zwischen den Sites eingebaut, so verhält sich der gesamte Grid-Verbund wie ein großer Parallelrechner mit ($\sum_{k=1}^{K} m_k$) Ressourcen. Damit würde solch ein Grid immer effizienter arbeiten als eine nicht Multi-Site-fähige Architektur. In der Praxis sind solche Aufteilungen von Jobs jedoch sehr schwer umzusetzen, da eine ständige Kommunikation zwischen einzelnen Teilen des Jobs über Firewallgrenzen und Sicherheitsmechanismen hinweg realisiert werden müsste.

6.3.2 Netzwerk und Daten

Im Modell werden unendliche schnelle Verbindungen zwischen den Sites angenommen, sodass die Migration eines Jobs keine Verzögerungen mit sich bringt. Die Verhandlung über Jobs kann dabei durch eine standardisierte Beschreibungssprache wie der *Job Scheduling Description Language* (JSDL) [25] geschehen, durch die der Kommunikationsaufwand bei Verhandlungen minimal bleibt. Obwohl die Festlegung eines konkreten Verhandlungsablaufes und die Spezifikation von geeigneten Protokollen andauernde Prozesse sind, so hat sich doch die WS-Agreement-Spezifikation [136] als Quasi-Standard etabliert. Für dieses Protokoll existieren weiterhin sehr effiziente Implementierungen, wie WSAG4J[8], mit denen jede Form von Verhandlung in sehr kurzer Zeit durchführbar ist.
Die eigentliche Übertragung von ausführbaren Dateien muss dann aber nur einmal bei der wirklichen Migration eines Jobs geschehen. Die realistischen Übertragungszeiten von eini-

[8] `http://packcs-e0.scai.fraunhofer.de/wsag4j/`, November 2010.

gen Sekunden sind aber im Vergleich zu den durchschnittlichen Ausführungszeiten der für die Ausführung im Grid relevanten Jobs[9] zu vernachlässigen. Es kann hier schon vermutet werden, dass Jobs, die vergleichsweise kurze Laufzeiten haben, von effizient arbeitenden Systemen primär nur lokal ausgeführt werden sollten. Die Berücksichtigung von Datenabhängigkeiten und die Co-Allokation von Daten und Jobs sind bei Grids immer wieder diskutiert worden und haben sich inzwischen zu einem eigenen Forschungsgebiet entwickelt. Dabei steht zumeist besonders die Behandlung von Arbeitsabläufen *(workflows)* im Vordergrund, für deren effiziente Abarbeitung Grid-Strukturen als besonders geeignet erscheinen. Obwohl dies zunächst als eine ungerechtfertigte Vereinfachung erscheint, werden sowohl Datentransfers als auch Abhängigkeiten zwischen Jobs in diesem Modell vernachlässigt. Im Wesentlichen sind dafür zwei Gründe zu nennen:
Es können für alle Jobs Daten als Arbeitsgrundlage vorausgesetzt werden und diese sind natürlich auch bei der Migration von Jobs als zusätzlicher Aufwand zu berücksichtigen. Nach Schley [150] kann dieser Prozess allerdings in drei Phasen aufgeteilt werden, wodurch sich ein Job in einen elementaren Workflow aus stellung *(stage-in)*, dem eigentlichen Berechnungsvorgang und Datenauslieferung *(stage-out)* unterteilen lässt. Diese sind aber untrennbar mit einem Job verknüpft, sodass bei der reinen Schedulingentscheidung von einem einzelnen Job mit entsprechenden Datentransfer-Anforderungen und einer gekapselten Gesamtlaufzeit ausgegangen werden kann. Die Laufzeit eines Jobs (und auch die eventuell vorhandene Laufzeitschätzung) wird hier also als Gesamtlaufzeit inklusive Datentransfer angesehen. Da die Netzwerkverbindungen als gleich (oder noch einfacher unendlich) schnell zwischen den Sites angenommen werden, sind Transfers zwischen den Sites immer gleich aufwendig. Eine explizite Planung von Datenabhängigkeiten wird hier nicht behandelt. Die vorgestellten Entscheidungsmethoden können aber auch für Datenscheduling angepasst werden.
In Bezug auf Abhängigkeiten zwischen Jobs sei hier angemerkt, dass solche explizit vom System unterstützt werden müssen und normalerweise erst einmal nicht berücksichtigt werden. Auch die hier verwendeten Arbeitslastaufzeichnungen enthalten keine Informationen über irgendwelche Abhängigkeiten zwischen den Jobs, was auf fehlende Workflow-Unterstützung durch die Originalsysteme hindeutet. Die im Abschnitt 6.2 noch genauer vorgestellten realen Systeme können zwar mit komplexen Workflows umgehen, basieren dazu aber immer auf einer einfachen hierarchischen Schedulingarchitektur. Da hier aber die Entwicklung von Mechanismen für die komplexe Entscheidungsfindung in dezentralen Grids im Vordergrund steht, werden die Abhängigkeiten vernachlässigt. Wie schon oben gesagt, sind die später vorgestellten Algorithmen ebenso für die Integration von Abhängigkeiten erweiterbar.

6.3.3 Informationspolitik

Abschließend sei hier noch die angenommene Informationspolitik innerhalb eines Grids genauer dargestellt, da diese den stärksten Einfluss auf das Design von Entscheidermechanismen und die zu erwartende Effizienz der Algorithmen hat. Hierarchische Architekturen können auch nur dann gute Schedulingergebnisse erzielen, wenn sie über die momentane

[9] Obwohl die Charakteristiken in den Aufzeichnungen, siehe Abschnitt 7.1, schwanken, haben selten mehr als 30 % der Jobs eine Ausführungszeit von weniger als einer Minute, siehe Feitelson [65]. Offensichtlich werden große Rechnerinstallationen überwiegend für die Ausführung von rechenintensiven Anwendungen mit entsprechend großen Laufzeiten genutzt.

Situation an den ihnen überantworteten Sites immer aktuell informiert sind. Dies ist üblicherweise der Fall, da mit der Aufgabe der Siteautonomie auch Informationen gegenüber dem Meta-Scheduler nicht zurückgehalten werden. Das ist ein gangbarer Weg, wenn für eine bestimmte Gemeinschaft eine einfache Grid-Infrastruktur erstellt werden soll, bei der ein Vertrauensverhältnis von vornherein gegeben ist und keine gegensätzlichen Präferenzen für Nutzer vorhanden sind. So findet dieser Ansatz, wie schon beschrieben, häufige Anwendung innerhalb einer virtuellen Organisation. Gleiches gilt für dezentrale Schedulingarchitekturen, bei denen ein Vertrauensverhältnis zwischen allen Partnern herrscht, siehe Lu u. a. [117], und Informationen unmittelbar kommuniziert werden. Im Allgemeinen kann jedoch solch ein unbeschränkter Informationsfluss nicht vorausgesetzt werden, da dies einen empfindlichen Eingriff in die Autonomie der Site-Administration bedeuten würde.
In dem hier angenommenen Grid-Modell wird daher von einer *restriktiven Informationspolitik* für alle Teilnehmer ausgegangen. Dies bedeutet konkret, dass sich der Informationsaustausch zwischen den Sites lediglich auf Jobanfragen beschränkt. Informationen bezüglich lokaler Kenngrößen wie der aktuellen Auslastung oder Anzahl der momentan wartenden Jobs werden nicht *zwischen* den Sites kommuniziert. Auch der häufig in Grids vorausgesetzte zentrale Informationsdienst, in dem der gesamte aktuelle Grid-Status von allen Teilnehmern abfragbar ist, wird in diesem Modell nicht eingesetzt.
Lediglich statische Informationen, wie die Anzahl der maximal zur Verfügung stehenden Prozessoren (m_k) sind allen Teilnehmern bekannt. Diese restriktive Informationspolitik garantiert, dass die Administration einer Site keine Rechtfertigung über ihr Verhalten in Bezug auf die Annahme oder Abgabe von Jobs geben muss. Obwohl die Umsetzung eines effizienten Jobaustauschs unter derartigen Bedingungen größere Herausforderungen mit sich bringt, stellt sie doch eine Anonymität und Autonomie für alle Teilnehmer sicher. Dies wird die Hürden für den Eintritt in einen Grid-Verbund stark herabsetzen.

7 Simulation von Grids

Für die Evaluation von Schedulingalgorithmen ist es unerlässlich, die Einsatzfähigkeit der konstruierten Algorithmen in realen Szenarien zu überprüfen. Theoretische Untersuchungen können üblicherweise anhand von kleinen Beispielen durchgeführt werden, sodass dort Ergebnisse auch ohne aufwendige Simulationen verifiziert werden. Reale Problemgrößen unterscheiden sich jedoch üblicherweise um Größenordnungen sowohl im Hinblick auf die Anzahl der eingereichten Jobs als auch auf den Einreichungszeitraum von den in theoretischen Untersuchungen betrachteten. Deshalb ist bei praktischen Problemen der unmittelbare Test von neu entwickelten Algorithmen nicht durchführbar, da für aussagekräftige Ergebnisse sehr lange Zeiträume von Jobeinreichungen evaluiert werden müssen. Deshalb werden für die hier vorgestellten Verfahren alle Evaluationen simulatorisch durchgeführt.
Die Grundlage von Simulationen bildet, neben der effizienten Abbildung der erstellten Grid-Modelle und Algorithmen in einer Software, die Eingangsdatenmenge, die einen entscheidenden Einfluss auf die Aussagekraft der Evaluation hat. Als Eingabe können entweder durch statistische Modelle erzeugte Daten genutzt werden oder vorhandene Aufzeichnungen von eingereichten Jobs herangezogen werden. Beide Möglichkeiten bringen allerdings unterschiedliche Probleme mit sich, die im Folgenden genauer betrachtet werden. Feitelson und Frachtenberg [66] diskutieren überdies verschiedene methodische Fehler, die sich bei der Evaluation von Schedulingalgorithmen ergeben können. In den in dieser Arbeit betrachteten Szenarien werden zwar ausschließlich Schedulingalgorithmen für Grids entwickelt, aber diese Grids werden aus dem Zusammenschluss von einzelnen parallelen Ressourcenverbünden gebildet. Deshalb ist auch hier die Problematik der Eingabedaten für ein Grid eine Obermenge zu den Problemen auf Parallelrechnern.
Aus diesem Grund werden kurz die bekannten Ansätze zur Modellierung von Jobeinreichungen und die verfügbaren realen Aufzeichnungen von Parallelrechnern vorgestellt. Anschließend wird die eingesetzte Simulationsumgebung beschrieben und es werden Referenzergebnisse für die verwendeten Daten präsentiert.

7.1 Statistische Modelle von Jobeinreichungen

Die statistische Modellierung der eingereichten Arbeitslast eines Parallelrechners und die Analyse ihrer Eigenschaften sind in der Forschung seit bereits mehr als 20 Jahren behandelt worden. Dazu wurden unterschiedliche vorhandene Arbeitslastaufzeichnungen, wie sie zum Beispiel im von Feitelson [65] gepflegten *Parallel Workload Archive* verfügbar sind, in unterschiedlichen Arbeiten analysiert. Die meisten Konzepte basieren dabei auf der Identifikation und Anpassung von statistischen Verteilungen für einzelne Eigenschaften und

Kenngrößen eines Jobs wie Parallelitätsgrad oder Laufzeitschätzung. Das wiederholte *Sampling* auf diesen angepassten, aber oftmals nicht gekoppelten Verteilungen für einzelne Charakteristiken ermöglicht dann die Erzeugung von synthetischen Eingabedaten, siehe zum Beispiel Feitelson [64]. Bei einem derartigen Vorgehen werden alle Jobattribute individuell behandelt. Bekannte und unbekannte Abhängigkeiten werden ignoriert oder können nicht berücksichtigt werden. Dies ist deshalb problematisch, da seit Langem bekannt ist, dass nachweisbare Korrelationen zwischen bestimmten Jobeigenschaften bestehen: So demonstrierten zum Beispiel Lo u. a. [116], dass unterschiedlich starke Korrelationen zwischen den Attributen Parallelitätsgrad und Laufzeit Auswirkungen auf die Ergebnisqualität von Schedulingalgorithmen haben. Auch Li u. a. [114] verarbeiten in ihrem Modell bewusst sowohl Verteilungen als auch Autokorrelationsfunktionen und verbinden sie in einem zweistufigen Verfahren mit besonderem Fokus auf Grids, siehe auch Abschnitt 7.1.2. Allerdings werden für das Gesamtmodell nur die Attribute Laufzeit und Speicherbedarf berücksichtigt. Um ein vollständiges für Simulationen verwertbares Modell zu bekommen, müssen verschiedene Teilmodelle, wie zum Beispiel jenes für Einreichungszeitpunkte [113] mit einem möglichen Modell für Laufzeitschätzungen [131], kombiniert werden. Diese Kombination ist allerdings in der Praxis nicht leicht zu realisieren und Analysen des resultierenden Kombinationsmodells sind dazu nicht verfügbar, weshalb die Qualität des Kombinationsmodells nur schwer beurteilbar ist.
Jann u. a. [97] unterteilen die Jobgröße m_j in Teilintervalle und erstellen einzelne Modelle für die Einreichungszeitpunkte *(interarrival times)* in jedem Intervall. Lublin und Feitelson [118] wiederum nutzen eine zweistufige Hyper-Gammaverteilung, um die Laufzeiten von Jobs zu modellieren, und nutzen zusätzlich die bekannte und zum Beispiel von Downey und Feitelson [51] ausführlich diskutierte Dominanz von Zweierpotenzen beim Parallelitätsgrad der Jobs.
Die Untersuchung und Modellierung der Laufzeitschätzung $\tilde{p}_j$ ist für die Anwendung von Backfilling-Algorithmen, siehe Abschnitt 5.4.2, unerlässlich. Auch wenn reale Nutzer oft die Laufzeit ihrer Jobs nach oben hin überschätzen, siehe auch Abschnitt 5.2.2, muss diese Eigenschaft auch in den entsprechenden Modellen wiederzufinden sein, da dies einen enormen Einfluss auf die Qualität der unterschiedlichen Algorithmen hat. Song u. a. [163] nutzen für die Anpassung von Laufzeitschätzungen Beta- und Gamma-Verteilungen, während die Jobs zusätzlich nach ihrer Laufzeit klassifiziert werden. Ein ähnliches Vorgehen wurde von Tsafrir u. a. [173] verfolgt, wobei die Laufzeitschätzungen als eine Sequenz von Charakteristiken aufgefasst werden, die ebenfalls die echten Laufzeiten mit einschließen. Beide Untersuchungen zeigen, dass es eine starke Verbindung zwischen der geschätzten und der wirklichen Laufzeit eines Jobs gibt.

7.1.1 Statistische Modelle auf Basis von Markovketten

Weiterhin zeigt Feitelson [62], dass Nutzer dazu tendieren, Jobs in Gruppen einzureichen, und somit Charakteristiken zwischen Jobs oftmals gleich oder sehr ähnlich sind. Diese Tatsache motiviert die Berücksichtigung von Selbstähnlichkeit und Autokorrelationen innerhalb von Einreichungssequenzen. Die Anwendung von Markovketten ist dabei unter anderem ein vielversprechendes Konzept, das auch bereits bei den Modellen von Song u. a. [164] erfolgreich eingesetzt wurde. Dort wurden jeweils Markovketten für die Laufzeit und den Parallelitätsgrad erzeugt und es wurde eine Verknüpfung zwischen beiden Ketten über den jeweiligen Korrelationskoeffizienten hergestellt. Dadurch können sequenzielle Abhängig-

keiten zwischen diesen beiden Attributen ziemlich genau beschrieben werden. Allerdings beinhaltet dieses Modell keine Einreichungszeitpunkte und auch keine Laufzeitschätzungen, die essenziell für die Verwendbarkeit in Simulationen sind.

Eine Weiterentwicklung dieses Modellierungsansatzes, die als einziges Modell alle für die Simulation benötigten Jobattribute berücksichtigt, wurde von Krampe u. a. [23, 22] vorgestellt. Dort wird die Erstellung eines umfassenden Modells als Zeitreihenproblem unter Annahme eines zeitlich unveränderlichen generierenden Prozesses betrachtet. Dadurch bietet sich wiederum die Nutzung von homogenen Markovketten als Modellierungskonzept an. Das generelle Vorgehen basiert zunächst auf der Schätzung der Übergangsmatrix einer homogenen Markovkette für die Parallelität, da sie den kleinsten Wertebereich aufweist und dadurch die Anzahl der möglichen Zustände einer Markovkette relativ gering ist. Weiterhin wird eine Dominanz von Zweierpotenzen beim Parallelitätsgrad in allen Arbeitslastaufzeichnungen dazu verwendet, eine Reduktion des Zustandsraumes durch Logarithmierung vorzunehmen. Dies lässt sich aufgrund der gegebenen Eigenschaften durchführen, ohne dass damit ein großer Informationsverlust verbunden ist, siehe Song u. a. [164]. Die Integration der weiteren Parameter erfolgt dann durch empirische Verteilungsfunktionen, die an die jeweiligen Zustandsübergänge in der Markovkette gekoppelt werden. Diese Verteilungsfunktionen werden ebenfalls zuvor an den entsprechenden Zustandsübergängen der Markovkette geschätzt. Durch die eingeführte Kopplung besteht die Möglichkeit, dass sich die in der Originalzeitreihe identifizierten Korrelationen ebenfalls in der Prädiktion wiederverwenden lassen.

Ein weiteres Problem bei der Modellierung von Zeitreihen sind verlässliche Bewertungsmaße für die Qualität der erstellten Modelle. Es ist aus der Statistik bekannt, dass die Anwendung des Kolmogorov-Smirnov-(KS-)Tests, siehe Kotz u. a. [107], für korrelierte Daten wenig aussagekräftig ist. Außerdem existiert bisher kein allgemein anwendbarer Test für die Angepasstheit *(Goodness of Fit)* einer Markovkette. Weiterhin ist ein Vergleich der vom Modell generierten Zeitreihen schwierig, da dort die verschiedenen Jobeigenschaften gleichzeitig beurteilt werden müssen. Ein alternatives Konzept, das den Vergleich über Schedulingalgorithmen realisiert und bei Song u. a. [164] in ähnlichem Kontext bereits verwendet wurde, basiert auf dem Vergleich bei Anwendung bekannter Algorithmen. Dazu werden zunächst Referenzkenngrößen, siehe Abschnitt 7.4, durch die Anwendung von Standardalgorithmen, wie zum Beispiel FCFS oder EASY-Backfilling, aus den Ursprungsdatensätzen erzeugt. Dann werden die gleichen Algorithmen auf die vom Modell erzeugten Daten angewendet. Ein Vergleich der Kenngrößen dient als Bewertungsmaß für die Güte der erzeugten Jobsequenzen, wobei weitere Vergleichswerte, wie zum Beispiel der Korrelationskoeffizient, unmittelbar auswertbar sind.

Das Modell von Krampe u. a. [22] zeigt für die zeitlichen Verläufe der Antwortzeit und Auslastung eine große Ähnlichkeit. Allerdings hat sich gezeigt, dass dieses Modell für kurzfristige Einreichungen keine verlässliche Vorhersage liefert. Dies ist vermutlich darin begründet, dass die erzeugte Übergangsmatrix über alle Jobs geschätzt wurde und damit die Annahme eines einzigen statischen erzeugenden Prozesses zugrunde liegt. Offenbar ist dies eine zu große Vereinfachung, da sich der erzeugende Prozess mit der Zeit ändert. Dabei ist eine Unterscheidung von Tageszeiten, Wochentagen und Wochenenden als sinnvoll anzusehen. Dies führt dann zu einer inhomogenen Markovkette mit zeitlich veränderlichen Übergangsmatrizen, die als Erweiterung für dieses Modell sinnvoll erscheint.

7.1.2 Statistische Modelle für Computational Grids

Im Gegensatz zu den zahlreichen Erkenntnissen bei der Untersuchung und Modellierung der Arbeitslast auf Parallelrechnern steckt die Forschung mit Fokus auf Cluster-Installationen und Grids noch in ihren Anfängen. Medernach [121] präsentiert zum Beispiel ebenfalls ein Markovkettenmodell, um die Einreichungszeitpunkte von Jobs in einem Grid zu beschreiben. Dabei wird ein Erneuerungsprozess *(renewal process)* angenommen, der allerdings in seiner Konsequenz dazu führt, dass zeitliche Abhängigkeiten (also Autokorrelationen) nur teilweise berücksichtigt werden können, siehe auch Jagerman u. a. [96]. Als Weiterentwicklung wenden Li und Muskulus [113] einen Markov-modulierten Poissonprozess an, bei dem somit auch Autokorrelationen berücksichtigt sind. Diese Modelle betreffen aber primär die Einreichungszeiten und sind noch weit davon entfernt, die komplexeren Abhängigkeiten in einem Grid ausreichend genau zu beschreiben.
Vielmehr ist es im Moment noch Gegenstand der Forschung, die genauen Abhängigkeiten in einem Grid mit Bezug auf Netzwerke, Speicher oder auch Nutzerdynamik genauer zu verstehen. So leistet zum Beispiel Li [112] eine detaillierte Analyse der vorhandenen Arbeitslastaufzeichnungen von Grids und Cluster-Installationen. Dabei wird sowohl das Nutzerverhalten als auch die Dynamik einer gesamten virtuellen Organisation weiter beleuchtet. Allerdings kann diese Arbeit nur als Inspiration für weitere Modellierungsanstrengungen dienen; ein vollständig nutzbares Modell ist auch aus diesen umfassenden Erkenntnissen noch nicht extrahierbar.
Die genannten Probleme bei der Erstellung eines umfassenden Arbeitslastmodells und das Fehlen eines allgemein anerkannten vollständigen Modells für Jobeinreichungen auf Parallelrechnern und Grids lassen noch keine Alternative zur Nutzung der vorhandenen Arbeitslastaufzeichnungen zu. Diese werden im nächsten Abschnitt genauer beschrieben.

7.2 Reale Arbeitslastaufzeichnungen

Aufgrund der oben genannten Probleme bei der Anwendung existierender Arbeitslastmodelle werden in dieser Arbeit Evaluationen ausschließlich auf Basis existierender Aufzeichnungen von realen Jobeinreichungen *(workload traces)* durchgeführt. Wie bereits erwähnt, sind in dem von Feitelson [65] bereitgestellten *Parallel Workload Archive* aktuell ungefähr 20 Arbeitslastaufzeichnungen realer Parallelrechner frei verfügbar. Diese beinhalten alle Charakteristiken, wie Einreichungsmuster oder Tageszyklen und Ähnliches, die Schedulingalgorithmen vor größere Herausforderungen stellen. Weiterhin sind detaillierte Informationen über die genaue Rechnerkonfiguration verfügbar und können somit für Simulationen nachgebildet werden. Aus diesem Grunde erfreuen sich diese Aufzeichnungen seit Jahren in der Forschung großer Beliebtheit und haben sich deshalb auch zu einer Art Standard für Schedulingevaluationen entwickelt. Von Chapin u. a. [42] wurde für die vereinfachte Nutzung das *Standard Workload Format* (SWF) eingeführt, das die Bereitstellung aller für die Evaluation von Schedulingalgorithmen auf Parallelrechnern notwendigen Informationen garantiert.
Für Grids wird seit Kurzem zusätzlich von Anoep u. a. [28] das *Grid Workload Archive* aufgebaut, das eine Sammlung von momentan zehn Aufzeichnungen aus realen Grid-Installationen enthält. In Anlehnung an das SWF wurde dazu das *Grid Workload Format* (GWF) formuliert, das noch weitere Informationen wie Abhängigkeiten, Speicherbedarf oder Zugehörigkeit zu einer virtuellen Organisation berücksichtigt. Obwohl diese Daten zunächst

für die Verwendung in dieser Arbeit als sehr geeignet erscheinen, werden sie aus folgenden Gründen nicht genutzt:
Zum einen basiert die in dieser Arbeit getroffene Modellannahme eines Computational Grid auf dem Zusammenschluss von einzelnen bisher nur autark agierenden Parallelrechnerinstallationen. Die dort angenommene lokale Nutzergemeinschaft ist sich also der Teilnahme eines Grid nicht bewusst. Die Grid-Interaktion findet rein auf der Betreiberebene statt und darf sich deshalb zunächst nicht im Verhalten der Nutzer widerspiegeln. Deshalb ist die Verwendung von reinen Parallelrechneraufzeichnungen durchaus sinnvoller. Ein anderer Grund betrifft die in dieser Arbeit ausschließlich vorgenommene Fokussierung auf sogenannte Rechenjobs, bei denen Datenabhängigkeiten und sehr spezielle Anforderungen nicht gegeben sind, siehe auch Abschnitt 5.2.2. Das Scheduling von reinen Rechenjobs ist nur dann besonders schwierig, wenn von einer lokalen Ressourcenknappheit und einer sehr großen lokalen Auslastung ausgegangen wird. Dies ist bei den bisher vorhandenen Grid-Aufzeichnungen nicht der Fall, da dort selten Auslastungswerte von über 20 % erreicht werden. Diese Szenarien bergen kein Potenzial für die Optimierung von Grid-Schedulern, weil lokal bereits genügend Ressourcen zur Verfügung stehen.
Die Daten des Parallel Workload Archive sind in zwei unterschiedlichen Versionen, nämlich als ursprüngliche und als gesäuberte Daten verfügbar. Die Originaldaten enthalten nur kleinere Korrekturen, die hauptsächlich auf Messfehler bei der Aufzeichnung zurückzuführen sind. Wie Feitelson und Tsafrir [69] zeigen, enthalten diese Aufzeichnungen allerdings eine Reihe an Anomalitäten, die entweder durch unbeabsichtigtes administratives Eingreifen oder durch kurzzeitiges Extremverhalten einzelner Nutzer *(workload flurries)* zu erklären sind. Sie entwickelten ein Verfahren, das diese Unregelmäßigkeiten identifiziert und die Aufzeichnungen entsprechend bereinigt. Derart gereinigten Daten werden nach ausgiebigen Analysen von den Bereitstellern zur ausschließlichen Nutzung empfohlen, weshalb auch in dieser Arbeit Evaluationen auf Basis der gereinigten Arbeitslastdaten durchgeführt werden. Eine genaue Darstellung der verwendeten Daten und der erzeugten Referenzergebnisse wird in Abschnitt 7.5 gegeben.

7.3 Das Teikoku Grid-Scheduling-Framework

Bevor nun auf die genauen Referenzergebnisse und die verwendeten Daten eingegangen wird, sollen in diesem Abschnitt die Simulationen von Grid-Umgebungen und die dazu gemachten Entwicklungen genauer betrachtet werden.
Existierende Simulationsframeworks für Grid-Umgebungen unterscheiden sich stark in ihrer Architektur und ihren Möglichkeiten. Das von Buyya und Murshed [40] entwickelte und inzwischen stark verbreitete Framework *GridSim* erlaubt die Abbildung eines kompletten Grids inklusive heterogener Ressourcen. Es unterstützt Modelle von nur einem Prozessor bis zur Abbildung vollständiger Parallelrechner. Es werden dabei Schedulingalgorithmen sowohl nach dem Space- als auch nach dem Timesharing-Muster unterstützt. GridSim ist dabei vor allem für die Simulation marktökonomischer Grid-Umgebungen ausgerichtet, bei denen jedem Nutzer ein sogenannter Ressourcenbroker zur Verfügung gestellt wird. Dieser kann bei Eingabe eines Jobs eine Liste verfügbarer Ressourcen, die zu dem Job passen, vom Grid-Informationsdienst abfragen. Anschließend liefert der Ressourcenbroker diejenigen Ressourcen zurück, für die eine vom Nutzer formulierte Kostenfunktion minimiert wird, oder führt das Scheduling des Jobs bezüglich dieser Ressourcen automatisch aus. Für

die Modellierung von Schedulingansätzen, die nicht dem ökonomischen Paradigma entsprechen, ist dies jedoch nicht ohne große Anpassungen einsetzbar.
Der *MicroGrid-Emulator*, siehe Song u. a. [166], hingegen ist speziell für die Simulation von Anwendungen gedacht, die mithilfe des Globus Toolkit erstellt wurden. Dabei wird eine virtuelle Grid-Umgebung mitsamt unterliegenden Ressourcen emuliert, auf der die Anwendung dann getestet werden kann. Für die Entwicklung eigener Schedulingalgorithmen beispielsweise ist MicroGrid jedoch unpraktisch, da der Realisierungsaufwand innerhalb des Globus Toolkit zu groß ist.
Das von Casanova [41] entwickelte *SimGrid* zeichnet sich unter anderem dadurch aus, dass es beliebige Arbeitslastmodelle und auch reale Aufzeichnungsdaten als Eingabe akzeptiert. Zusätzlich eignet es sich im Speziellen für die Simulation neuer Schedulingalgorithmen. Hierbei ist es jedoch auf eine einzelne zentrale Schedulinginstanz beschränkt und muss für die Simulation dezentraler Jobaustauschverfahren stark modifiziert werden.
Im Rahmen dieser Arbeit wird deshalb das extra zu diesem Zweck entwickelte *Teikoku Grid-Scheduling-Framework*, siehe Grimme u. a. [20], zur Simulation von Grid-Umgebungen genutzt. Dieses im Vergleich zu GridSim eher schlanke Framework eignet sich für die Simulation sowohl zentraler als auch dezentraler Grid-Umgebungen. Neben dem übersichtlichen Aufbau des Frameworks und der speziell auf Geschwindigkeit optimierten Implementierung lassen sich auch maschinelle Optimierungsverfahren und CI-Methoden sehr leicht integrieren.

7.4 Ziel- und Bewertungsfunktionen

Für die Qualitätsbeurteilung von erzeugten Schedules ist es sowohl bei reinen Parallelrechnern als auch im Grid notwendig, sich auf bestimmte Zielfunktionen zu beziehen. Diese können zum einen einfache reguläre Funktionen, wie die Produktionsspanne *(Makespan)*, die Summe der Beendigungszeiten *(Total Completion Time)* $\sum C_j$ oder die maximale Verspätung *(Maximum Lateness)* L_{max} sein, siehe Brucker [37]. Zum anderen sind für die hier vorrangig betrachteten Schedulingprobleme auf Parallelrechnern Funktionen für die Bewertung erforderlich, die auch den Problemkontext und dessen Eigenschaften, wie Feitelson [63] zeigt, berücksichtigen. Zusätzlich sind praxisnahe Zielfunktionen oftmals nicht so einfach zu formulieren, da sie eine komplexe Problemsicht ausdrücken sollen und sich deshalb auch dynamisch, wie Ernemann u. a. [54] zeigen, durch das Wissen über erreichbare Lösungsmöglichkeiten ändern können. Dies führt dazu, dass häufig mehrere Kriterien gleichzeitig betrachtet werden müssen und das Schedulingproblem dadurch multikriteriell wird, siehe T'kindt und Billaut [170]. Da diese Probleme allgemein sehr schwer zu lösen sind, existiert nur für sehr wenige Probleme überhaupt ein optimaler polynomieller Algorithmus.[10] Es werden deshalb entweder Approximationsverfahren genutzt, wie sie zum Beispiel von Mu'alem und Feitelson [128] für parallele Maschinen beschrieben, oder *a priori* Gewichtungen eingeführt. Mit den Gewichten lässt sich dann durch Aggregation der Kriterien ein einkriterielles Problem erzeugen.
Diese grob skizzierten Probleme in Bezug auf mögliche Zielfunktionen erzwingen auch für die in dieser Arbeit betrachteten Problemstellungen die Einführung nötiger Vereinfachungen. In dieser Arbeit wird deshalb vorrangig eine nutzerzentrierte Sicht auf Parallelrechner

[10] Nach T'kindt und Billaut [170] ist bisher nur das Problem $1|d_j|\sum C_j, L_{max}$ bekannt, das sich mit Komplexität $\mathcal{O}(n^3 \log n)$, siehe van Wassenhove und Gelders [177], optimal lösen lässt.

und Computational Grid-Sites verfolgt. Wie bereits eingangs erwähnt, ist die Teilnahme an einem Grid für Betreiber nur dann vorteilhaft, wenn dadurch kürzere Antwortzeiten, siehe Abbildung 5.1, für die Nutzer erreicht werden. Dies ist darin motiviert, dass die Dienstgüte primär an der schnellen Abarbeitung von Jobs gemessen wird. Dabei wird weiterhin davon ausgegangen, dass zunächst alle Nutzer eine gleich große Priorität beim Betreiber besitzen und keine Bevorteilungen formuliert wurden. Es sei jedoch bemerkt, dass dies keine Einschränkung der hier vorgestellten allgemeinen Methodik darstellt, da für die Umsetzung von derartigen Priorisierungen die bereits erwähnten Verfahren von Franke u. a. [7] unabhängig auf LRMS-Ebene genutzt werden können.
Alle weiteren Ziel- und Bewertungsfunktionen werden nun im Detail vorgestellt. Bei allen Evaluationen in dieser Arbeit werden Ergebnisse, die im Austausch zwischen Sites erreicht werden, mit den Ergebnissen bei rein lokaler Ausführung verglichen. Dabei wird im Folgenden wiederum die Menge der in Site k lokal eingereichten Jobs mit τ_k bezeichnet und die Menge der auf Site k ausgeführten Jobs mit π_k. Ferner sei S_k der Schedule der Site k. Somit bezeichnet $C_j(S_k)$ den Beendigungszeitpunkt von Job $j \in \pi_k$ im Schedule S_k auf Site k. Bei rein lokaler Ausführung gilt folglich $\tau_k = \pi_k$.

7.4.1 Produktionsspanne (Makespan)

Die Produktionsspanne oder auch Makespan bezeichnet den Zeitpunkt der Beendigung des letzten Jobs für eine gewisse eingereichte Menge von ausgeführten Jobs π_k, siehe Definition 7.1.

$$C_{max} = \max_{j \in \pi_k} \{C_j\} \tag{7.1}$$

Die Minimierung der Produktionsspanne erreicht bei sequenziellen Jobs auf parallelen Maschinen automatisch einen Ausgleich der Arbeitslast.[11] Die bereits vorgestellten Heuristiken für parallele Jobs, siehe Abschnitt 5.4.2, sind auch hauptsächlich auf die Minimierung der Produktionsspanne hin ausgelegt. Für einen beendeten Schedule, das heißt für eine abgeschlossene Menge von eingereichten Jobs, impliziert dies weiterhin eine Fokussierung auf die Systemauslastung, siehe Abschnitt 7.4.3, da ein geringer Makespan mit einer hohen Auslastung korreliert.
Als Zielfunktion für Online-Probleme wird C_{max}, auch aufgrund seiner relativ einfachen Handhabung, gern für theoretische Untersuchungen herangezogen. So existieren unter anderem bereits eine Vielzahl von Resultaten für LIST-Scheduling, siehe auch Abschnitt 5.4.2: Shmoys u. a. [158] beweisen für das Problem $P_m|r_j, ncv|C_{max}$, dass die untere Schranke in Gleichung 7.2

$$\frac{C_{max}(LIST)}{C_{max}(OPT)} \leq 2 - \frac{1}{m_k} \tag{7.2}$$

gilt und McNaughton [120] zeigte bereits ihre Gültigkeit bei erlaubter Unterbrechung der Jobausführung. Für starre, parallele Batch-Jobs ist schon 1975 von Garey und Graham [80] die Gültigkeit der gleichen Schranke wie für sequenzielle Jobs aus Gleichung 7.2 im Offline-Fall gezeigt worden. Im Online-Fall kann ebenfalls die gleiche Garantie für parallele Jobs und das Problem $P_m|m_j, ncv|C_{max}$ und sogar, wie Naroska und Schwiegelshohn [130] zeigen, für das Problem $P_m|r_j, m_j, ncv|C_{max}$ gegeben werden. Schwiegelshohn u. a. [155] beweisen,

[11] Das Problem $P_2||C_{max}$ ist identisch mit dem PARTITION Problem, siehe Pinedo [137].

dass diese konstante Schranke durch polynomielle Algorithmen jedoch im Kontext eines Computational Grids nicht mehr garantiert werden kann.
Für die hier vorliegenden Online-Probleme ist allerdings die Betrachtung des Makespan insofern problematisch, als dass der letzte eingereichte Job nicht bekannt ist. Bei einem Online-System wird von einem nicht endenden Strom von eingereichten Jobs ausgegangen, weshalb die Beendigungszeit des letzten eingereichten Jobs höchstens für einen lokalen Teil des Problems bestimmbar ist. Weiterhin zielt ein Grid auf einen regen Austausch von Jobs ab, wodurch sich natürlich Ausführungen auf entfernten Sites ergeben. Um Aussagen über die Effizienz dieses Austauschs machen zu können, ist es aber notwendig, die Ergebnisse mit rein lokaler Ausführung zu vergleichen. Dies ist nur möglich, wenn die Jobs einer Einreichungsmenge τ_k von einer Nutzergemeinschaft evaluiert wird. Bei Betrachtung nur einer Site entsteht dabei das Problem, dass die Jobs verschiedener Nutzergemeinschaften gleichzeitig ausgeführt werden. Die Evaluation eines sogenannten virtuellen Schedules, der sich auf verschiedenen Sites verteilt, ist dadurch mit dem Makespan nicht möglich. Eine jobbezogene Zielfunktion, die aus ebendiesen Gründen in dieser Arbeit primär genutzt wird, ist in Abschnitt 7.4.5 beschrieben.

7.4.2 Ressourcenprodukt (Squashed Area)

Eine weitere Kenngröße sowohl für einzelne Jobs als auch für eingereichte Jobmengen ist das Ressourcenprodukt. Dies wird für einen einzelnen Job j als Produkt aus seinem Parallelitätsgrad und seiner realen Ausführungszeit gebildet, siehe Gleichung 7.3.

$$\mathrm{SA}_j = m_j \cdot p_j \tag{7.3}$$

Das Produkt beschreibt, wie extensiv Ressourcen von einem Job genutzt werden. Als Kennzeichen eines Jobs kann dies auch als Gewichtung für weitere Ziel- und Bewertungsfunktionen genutzt werden. In Gleichung 7.4 wird das gesamte Ressourcenprodukt für eine ganze Menge von Jobs, die auf Site k ausgeführt wurde, gebildet.

$$\mathrm{SA}_k = \sum_{j \in \pi_k} m_j \cdot p_j \tag{7.4}$$

Das gesamte Ressourcenprodukt auf Site k (SA_k) quantifiziert die Menge der gesamten genutzten Ressourcen. Dieser Wert wird in dieser Arbeit als Charakteristik für Arbeitslastaufzeichnungen genutzt und dient als Element in weiteren Bewertungsfunktionen.

7.4.3 Auslastung (Utilization)

Zunächst ist zur aktuellen Zustandsbestimmung die aktuelle Auslastung $\mathrm{U}_k(t)$ einer Site k zum Zeitpunkt t von großer Bedeutung. Sie wird nach Gleichung 7.5 berechnet, wobei das Verhältnis von aktuell belegten zu allgemein verfügbaren Ressourcen an einem gegebenen Zeitpunkt bestimmt wird.

$$\mathrm{U}_k(t) = \frac{1}{m_k} \sum_{\{j \in \pi_k \mid C_j(S_k) - p_j \leq t < C_j(S_k)\}} m_j \tag{7.5}$$

Die gesamte Auslastung U_k einer Site k nach der Beendigung aller ausgeführten Jobs $j \in \pi_k$ wird hingegen nach Gleichung 7.6 berechnet.

$$U_k = \frac{\sum\limits_{j \in \pi_k} p_j \cdot m_j}{m_k \cdot \left(C_{max,k} - \min\limits_{j \in \pi_k} \{C_j(S_k) - p_j\} \right)} \tag{7.6}$$

Dabei wird das oben eingeführte gesamte Ressourcenprodukt ins Verhältnis zum maximal verfügbaren Ressourcenprodukt gestellt, das sich durch einen vergebenen Betrachtungszeitraum ergibt. Die Einschränkung auf ein derartiges Betrachtungsfenster ist natürlich bei realen Systemen rein willkürlich, da es sich um einen kontinuierlichen Einreichungsprozess handelt. Im Rahmen der hier durchgeführten Simulationen lässt sich das Betrachtungsfenster aber entsprechend durch den frühesten Startzeitpunkt ($\min_{j \in \pi_k}\{C_j(S_k) - p_j\}$) der Jobs bis zum letzten Beendigungszeitpunkt, dem Makespan $C_{max,k} = \max_{j \in \pi_k}\{C_j(S_k)\}$ auf Site k sinnvoll begrenzen. Diese Zeitspanne wird dann mit der Größe des Parallelrechners m_k multipliziert.

$$U_o = \frac{\sum\limits_{k \in K} \sum\limits_{j \in \tau_k} p_j \cdot m_j}{\sum\limits_{k \in K} m_k \cdot \left(\max\limits_{k \in K}\{C_{max,k}\} - \min\limits_{k \in K}\{\min\limits_{j \in \pi_k}\{C_j(S_k) - p_j\}\} \right)} \tag{7.7}$$

Eine globale Version der Auslastung ist weiterhin in Gleichung 7.7 gegeben, bei der nun über alle Sites eines Grid die Effektivität der Maschinenauslastung bestimmt ist. Dazu wird U_k einfach auf eine globale Sicht erweitert.
Da die in Gleichung 7.6 definierte Auslastung U_k den Makespan mit berücksichtigt, wird ein zeitlicher Bezug hergestellt. Dies ist für die Bestimmung der Veränderung der Auslastung auf einer Site durch Jobaustausch problematisch: Ein Vergleich der U_k-Werte ist nur dann zulässig, wenn $C_{max,k}$ beim Szenario mit und ohne Austausch identisch ist. Andernfalls kann eine hohe Auslastung im Grid-Szenario für eine Site und damit eine hohe Effizient suggeriert werden, obwohl sie vorrangig Jobs zu anderen Sites migriert hat und nur sehr wenige Jobs lokal ausgeführt wurden. Dies hätte dann zur Folge, dass der $C_{max,k}$ Wert sehr klein ist und dadurch ein hoher Auslastungswert zustande kommt. Es ist daher für einen fairen Vergleich der Auslastungen zum einen notwendig, diese auf einen einheitlichen maximalen Makespan für das Szenario mit und ohne Austausch zu beziehen. Zum anderen muss auch der früheste Startzeitpunkt eines Jobs entsprechend angepasst sein.

7.4.4 Änderung des Ressourcenprodukts

Um dem oben beschriebenen Problem zu begegnen, wird zusätzlich die Änderung des Ressourcenprodukts (ΔSA_k) auf einer Site k betrachtet, siehe Gleichung 7.8.

$$\Delta SA_k = \frac{\sum\limits_{j \in \pi_k} m_j \cdot p_j}{\sum\limits_{j \in \tau_k} m_j \cdot p_j} = \frac{SA_k}{\sum\limits_{j \in \tau_k} m_j \cdot p_j} \tag{7.8}$$

Durch ausschließliche Betrachtung des Verhältnisses von lokal eingereichter Arbeitslast ($j \in \tau_k$) und lokal ausgeführter Arbeitslast ($j \in \pi_k$) wird die Makespanproblematik bei Betrachtung der Auslastung umgangen. Aus Betreibersicht spiegelt diese Bewertungsfunktion die

wirkliche Änderung der Auslastung durch Jobmigrationen wider, siehe auch Grimme u. a. [14].

7.4.5 Durchschnittliche gewichtete Antwortzeit (AWRT)

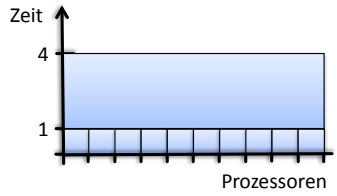
Die wichtigste Bewertungsfunktion dieser Arbeit evaluiert Schedulingresultate aus Betreibersicht und unabhängig von System- und Jobcharakteristiken. Dazu wird die von Schwiegelshohn und Yahyapour [156] eingeführte durchschnittliche gewichtete Antwortzeit *(Average Weighted Response Time, AWRT)* für alle Jobs $j \in \tau_k$ berechnet, die initial auf einer Site k eingereicht wurden, siehe Gleichung 7.9. Die Antwortzeit berechnet sich dabei als Zeitraum $(C_j - r_j)$ zwischen dem Einreichungs- und Beendigungszeitpunkt, siehe Abbildung 5.1.

$$\mathrm{AWRT}_k = \frac{\sum\limits_{j \in \tau_k} m_j \cdot p_j \cdot (C_j(S) - r_j)}{\sum\limits_{j \in \tau_k} m_j \cdot p_j} = \frac{\sum\limits_{j \in \tau_k} \mathrm{SA}_j \cdot (C_j(S) - r_j)}{\sum\limits_{j \in \tau_k} \mathrm{SA}_j} \tag{7.9}$$

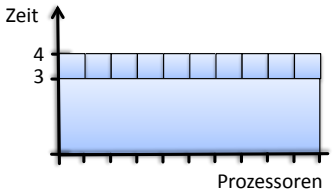
Der gesamte durchschnittlich gewichtete Funktionswert bezieht sich dabei auf das gesamte Grid, weshalb auch die Jobs berücksichtigt werden, die auf entfernten Sites ausgeführt wurden; es werden also jeweils die Jobs einer Einreichungsmenge zur Berechnung herangezogen. Deshalb bezieht sich die Bezeichnung $C_j(S)$ in Gleichung 7.9 auf jeweils diejenige Site, auf der Job j zur Ausführung gebracht wurde.

Besondere Erklärung verlangt die genutzte Gewichtung: Ausgangspunkt ist die bereits oben erwähnte Gesamtbeendigungszeit $\sum C_j(S)$, die die Gesamtsumme aller Beendigungszeiten aller Jobs im Schedule S wiedergibt. Diese Zielfunktion bezieht sich auf jeden einzelnen Job und hat insbesondere eine unerwünschte Eigenschaft: Aus Abbildung 7.1 ist ersichtlich, dass im linken Schedule (a) eine deutlich kürzere Gesamtbeendigungszeit als im rechten Schedule (b) erreicht wird, obwohl in beiden Fällen die gleiche Gesamtbearbeitungszeit benötigt wird. In dieser Zielfunktion werden also sequenzielle Jobs vor parallelen Jobs klar bevorzugt, wenn sie die gleiche Ausführungszeit haben. Ein Nutzer würde deshalb von einer Aufteilung paralleler Jobs in viele sequenzielle profitieren. Ebenfalls wäre es möglich einen langlaufenden Job in viele kurze Jobs mit entsprechenden Abhängigkeiten aufzuteilen. Es ist für eine Zielfunktion allerdings nicht wünschenswert, dass eine bestimmte Aufteilung die Qualität der Lösung beeinflussen kann. Deshalb wurde die Gewichtung mit dem Ressourcenprodukt $(m_j \cdot p_j)$ eingeführt. Sie bewirkt eine Neutralität der Metrik gegenüber der Jobcharakteristiken und bezieht sich dadurch nur noch auf die durch Schedulingalgorithmen oder Austauschstrategien erreichte Qualität der Lösung.

Es sei hier der Vollständigkeit halber noch erwähnt, dass für theoretische Untersuchungen die Summe der Beendigungszeiten und damit auch der Antwortzeiten auf parallelen Maschinen bisher sehr wenig betrachtet wurde. Das Problem $P_m||\sum w_j C_j$ ist für beliebige Job-Gewichte w_j NP-schwer, da sich das PARTITION Problem auf $P_2||\sum w_j C_j$ reduzieren lässt, siehe Pinedo [137]. Für dieses Problem konnte von Kawaguchi und Kyan [104] ein fester Approximationsfaktor von $0,5 \cdot (1 + \sqrt{2}) \approx 1.207$ für LIST-Scheduling mit beliebigen Listen nachgewiesen werden. Für das Online-Problem $P_m|r_j, ncv|\sum w_j C_j$ wurde in einer neueren Arbeit von Schwiegelshohn [154] ein kompetitiver Faktor von 1,25 für sequenzielle Jobs bewiesen.



(a) $\sum C_j = 14; \quad \sum m_j p_j C_j = 130$



(b) $\sum C_j = 43; \quad \sum m_j p_j C_j = 130$

Abbildung 7.1: Beispiel zweier Schedules nach Schwiegelshohn [154] mit einem parallelen und mehreren sequenziellen Jobs.

7.4.6 Migrationsraten

In Bezug auf die Bewertung des Grids ist eine Quantifizierung und Interpretation des erreichten Jobaustauschs von besonderem Interesse. Deshalb werden hier zwei Darstellungen der Jobmigrationen eingeführt. Dazu wird zunächst Matrix M_n, siehe Definition 7.10, eingeführt, die eine Repräsentation der Jobverteilungen im Grid liefert.

$$M_n = \begin{bmatrix} \frac{|\pi_{11}|}{|\tau_1|} & \cdots & \frac{|\pi_{1k}|}{|\tau_1|} \\ \vdots & \ddots & \vdots \\ \frac{|\pi_{l1}|}{|\tau_l|} & \cdots & \frac{|\pi_{lk}|}{|\tau_l|} \end{bmatrix}, \quad \{l,k\} \in [1 \cdots |K|] \tag{7.10}$$

Hier werden die Jobs entsprechend ihrer Ursprungs-Site und ihrer ausführenden Site zugeordnet dargestellt. Dabei bezeichnet τ_k die Menge von Jobs, die auf Site k eingereicht wurden und π_{lk} die Menge der Jobs, die zwar auf Site l eingereicht, aber auf Site k ausgeführt wurden. Deshalb beziehen sich die Zeilen der Matrix M_n auf die ursprünglichen Einreichungsstellen und die Spalten auf die tatsächlichen Ausführungsorte.
Zusätzlich ist es für die Analyse interessant, die Menge der tatsächlich migrierten Arbeitslast zu quantifizieren. Deshalb wird hier, siehe Definition 7.11, zusätzlich die Matrix M_{SA} definiert, die das oben eingeführte Konzept analog auf das Ressourcenprodukt (SA) bezieht.

$$M_{SA} = \begin{bmatrix} \frac{\mathrm{SA}_{\pi_{11}}}{\mathrm{SA}_{\tau_1}} & \cdots & \frac{\mathrm{SA}_{\pi_{1k}}}{\mathrm{SA}_{\tau_1}} \\ \vdots & \ddots & \vdots \\ \frac{\mathrm{SA}_{\pi_{l1}}}{\mathrm{SA}_{\tau_l}} & \cdots & \frac{\mathrm{SA}_{\pi_{lk}}}{\mathrm{SA}_{\tau_l}} \end{bmatrix}, \quad \{l,k\} \in [1 \cdots |K|] \tag{7.11}$$

Beispielhaft wird das Ressourcenprodukt für die Menge π_{kk} nun durch $\mathrm{SA}_{\pi_{kk}} = \sum_{j \in \pi_{kk}} p_j \cdot m_j$ berechnet, siehe auch Gleichung 7.3.

7.5 Referenzergebnisse für vollständige Eingabedaten

In diesem Abschnitt werden nun die verwendeten Arbeitslastaufzeichnungen genauer beschrieben und die mit dem Teikoku Grid-Scheduling-Framework ermittelten Referenzdaten

präsentiert. Alle im Folgenden dargestellten Vergleiche gegenüber der rein lokalen Ausführung aller Jobs beziehen sich dann auf die hier aufgelisteten Resultate.
Alle Daten sind aus den zunächst von Feitelson bereinigten Aufzeichnungen des Parallel Workload Archive entnommen, siehe auch Tabelle 7.1:

1. Der KTH-Datensatz enthält Einreichungen auf einem Parallelrechner mit 100 Prozessoren vom Typ IBM RS/6000 SP. Die Einreichungen wurden im Swedish Royal Institute of Technology in Stockholm von Ende September 1996 bis Ende August 1997, also in einem Zeitraum von 11 Monaten, aufgezeichnet.

2. Vom 430 Prozessoren umfassenden „IBM RS/6000 SP"-System im Cornell Theory Center in Ithaca, NY, stammt der in der Zeit von Ende Juni 1996 bis Ende Mai 1997 aufgezeichnete CTC-Datensatz.

3. Weiterhin wurde der in den zwei Jahren von April 1994 bis September 1996 im Los Alamos National Lab (LANL) aufgezeichnete Datensatz verwendet. Die Maschine war eine CM-5 des US-amerikanischen Herstellers Thinking Machines mit 1.024 Knoten. Bei der vom Betreiber umgesetzten Konfiguration wurden hier nur Jobs mit 32 Prozessoren ($m_j \geq 32$) und mit einem Parallelitätsgrad von Zweierpotenzen ($m_j \in \{32, 64, \ldots\}$) zugelassen. Deshalb sind die Antwortzeiten sehr kurz und es kann von einer annähernd optimalen Allokation ausgegangen werden.

4. Die letzten drei Datensätze entstammen alle von den Parallelrechnern des San Diego Supercomputer Center in La Jolla, CA:
   a) Der SDSC00-Datensatz enthält alle Einreichungen auf eine 128 Prozessor „IBM RS/6000 SP"-Maschine über 24 Monate von April 1998 bis April 2000.
   b) Der SDSC03-Datensatz der „Blue Horizon"-Installation, aufgezeichnet auf einer 1.152 Prozessoren umfassenden „IBM RS/6000 SP"-Maschine über mehr als 24 Monate von April 2000 bis Januar 2003.
   c) Der SDSC05-„DataStar"-Datensatz, der auf einem „IBM *e*Server pSeries 655/690"-System mit insgesamt 1.664 Prozessoren innerhalb der 13 Monate von Anfang März 2004 bis Ende April 2005 aufgezeichnet wurde.

Obwohl die Datensätze teilweise schon vor über 10 Jahren aufgezeichnet wurden, stellen sie realistische Einreichungen lokal angepasster Nutzergemeinschaften dar. Da hier nicht die Strukturen von neuartigen Programmen, sondern nur das Verhalten der Nutzer im Vordergrund steht, stellt die zeitliche Distanz hier keine Qualitätsminderung in Bezug auf ein spezifisches Verhalten dar. Für die Simulation von Grid-Umgebungen ist es wichtig, dass während des gesamten simulierten Zeitraums Nutzergemeinschaften Einreichungen an ihren Systemen vornehmen. Ansonsten würde die unwahrscheinliche Situation auftreten, dass eine einsatzfähige Installation von einem gewissen Zeitpunkt an überhaupt nicht mehr genutzt wird und die Ressourcen dann dem gesamten Grid uneingeschränkt zur Verfügung stehen. Dies würde zu einer extremen Verzerrung im Ressourcenangebot führen und die Ergebnisse wären mit denen rein lokaler Ausführung nicht mehr vergleichbar. Die vorhandenen Datensätze differieren jedoch in ihrer Länge, sodass eine Kürzung aller Datensätze in einem Szenario auf den jeweils kürzesten notwendig erscheint. Datensätze mit gleicher Länge (gekennzeichnet durch angehängte Monatslängen, also zum Beispiel CTC-11 für die

Daten	n	m_k	Monate	SA
KTH-11	28.479	100	11	2.017.737.644
CTC-11	77.199	430	11	8.279.369.703
LANL96-11	57.715	1.024	11	16.701.881.984
LANL96-13	67.043	1.024	13	19.467.626.464
LANL96-24	110.769	1.024	24	35.426.509.344
SDSC00-11	29.810	128	11	2.780.016.139
SDSC00-24	54.006	128	24	6.656.350.775
SDSC03-11	65.584	1.152	11	23.337.438.904
SDSC03-13	78.951	1.152	13	27.410.557.560
SDSC03-24	176.268	1.152	24	54.233.113.112
SDSC05-11	74.903	1.664	11	29.392.365.928
SDSC05-13	84.876	1.664	13	34.347.849.760

Tabelle 7.1: Kennzeichen der verwendeten Arbeitslastaufzeichnungen und ihrer gekürzten Versionen.

ersten 11 Monate des CTC-Datensatzes) können dann beliebig zu einem Grid-Szenario zusammengestellt werden, siehe Tabelle 7.1.

In Tabelle 7.2 sind nun die Ergebnisse einiger in Abschnitt 7.4 eingeführten Zielfunktionen bei Anwendung der in Abschnitt 5.4.2 beschriebenen Algorithmen im LRMS aufgelistet. Offensichtlich erzielt der EASY-Backfilling-Algorithmus auch bei diesen gekürzten Daten-

	FCFS			EASY			LIST		
Daten	AWRT	U	C_{max}	AWRT	U	C_{max}	AWRT	U	C_{max}
KTH-11	418.171,98	68,67	29.381.344	75.157,63	68,72	29.363.626	80.459,27	68,72	29.363.626
CTC-11	58.592,52	65,70	29.306.682	52.937,96	65,70	29.306.682	53.282,02	65,70	29.306.682
LANL96-11	13.886,61	55,65	29.306.900	12.442,02	55,65	29.306.355	13.082,10	55,65	29.307.464
LANL96-13	14.064,40	56,24	33.801.984	12.467,61	56,24	33.802.986	13.178,91	56,24	33.802.986
LANL96-24	12.265,92	55,54	62.292.428	11.105,15	55,54	62.292.428	11.652,44	55,54	62.292.428
SDSC00-11	266.618,82	71,53	30.361.813	73.605,86	73,94	29.374.554	74.527,98	73,96	29.364.791
SDSC00-24	2.224.462,43	75,39	68.978.708	112.040,42	81,78	63.591.452	11.6260,72	82,26	63.618.662
SDSC03-11	72.417,87	68,58	29.537.543	50.772,48	68,74	29.471.588	48.015,99	68,87	29.413.625
SDSC03-13	82.149,66	69,92	34.030.926	54.546,90	70,12	33.932.665	52.012,84	70,14	33.921.090
SDSC03-24	166.189,54	73,22	64.299.555	70.774,86	73,91	63.696.120	72.220,28	73,91	63.695.942
SDSC05-11	70.579,08	60,12	29.380.453	54.953,84	60,17	29.357.277	53.403,51	60,20	29.339.408
SDSC05-13	72.690,44	61,08	33.793.591	56.990,23	61,08	33.793.591	55.250,72	61,08	33.793.591

Tabelle 7.2: Referenzergebnisse der AWRT (in Sekunden), U (in %) und C_{max} (in Sekunden) für verschiedene LRMS-Algorithmen ohne Jobaustausch unterteilt nach Datensätzen.

sätzen im Hinblick auf AWRT und U durchweg bessere Ergebnisse als FCFS. Diese Resultate bestätigen die bereits in früheren Arbeiten auf den Originaldaten von Feitelson und Weil [70] durchgeführten Untersuchungen. Allerdings erzielt der LIST-Algorithmus für einzelne Datensätze (zum Beispiel SDSC03-11 oder SDSC05-11) sogar noch bessere AWRT-Werte als EASY, was auf spezielle Eigenschaften genau dieser Datensätze zurückzuführen ist. Für den Großteil der Daten ist EASY allerdings der beste LRMS-Algorithmus. Allerdings erfordert er auch Laufzeitschätzung als zusätzliche Information. Insgesamt ist festzustellen, dass sich durch zusätzlich verfügbare Informationen Schedulingentscheidungen deutlich verbessern lassen. Aber auch durch das Vorziehen von Jobs, wie es beim List-Scheduling möglich ist, können die Ergebnisse von FCFS geschlagen werden. Trotzdem kann aber generell keine Überlegenheit von LIST gegenüber EASY nachgewiesen werden.

Es ist an dieser Stelle noch notwendig zu erwähnen, dass in dieser Arbeit entgegen vielen anderen Untersuchungen keine Zeitzonen bei den Einreichungen berücksichtigt werden. Wie Lublin und Feitelson [118] zeigen, sind deutliche Unterschiede in der Einreichungsaktivität im Laufe eines Tages nachweisbar: So werden in allen Datensätzen über Mittag zum Beispiel deutlich mehr Einreichungsaktivitäten verzeichnet als während der Nacht. Wenn, wie in dieser Arbeit, keine Zeitverschiebungen zwischen den Standorten berücksichtigt werden, sieht dieser Tageszyklus für alle Datensätze relativ ähnlich aus. Üblicherweise wird in Grids aber genau dieser Effekt ausgenutzt, sodass gerade in der Nachtphase sich befindliche Sites Aufgaben von momentan in Spitzenlast laufenden Sites einer anderen Zeitzone übernehmen können und somit ein globaler Lastausgleich erreicht werden kann. Die Vorteile, die ein solches globales Grid für das Scheduling haben kann, wurden bereits von Ernemann u. a. [58] evaluiert. Für die Zukunft wird aber durch die fortschreitende Automatisierung von Einreichungsmechanismen der in den vorhandenen Daten identifizierte Tagesrhythmus immer weiter verschwinden, siehe Li und Muskulus [113]. Außerdem werden in dieser Arbeit vornehmlich kleine Zusammenschlüsse, also Verbände von zwei bis fünf Teilnehmern, betrachtet, die üblicherweise auch eine örtliche Lokalität aufweisen. Dadurch kann der globale Ausgleichseffekt nicht für das Scheduling genutzt werden. Dies bedeutet natürlich auch, dass im globalen Kontext die hier erreichten Resultate durch Zeitzoneneffekte noch verbessert werden können. Andererseits ist also das hier untersuchte Szenario ohne Zeitzonen der am schwierigsten zu optimierende Fall, da potenziell vorteilhafte Zeitverschiebungseffekte nicht genutzt werden können.

7.6 Referenzergebnisse für aufgeteilte Eingabedaten

Wie schon eingangs erwähnt ist es bei der Erstellung von Grid-Schedulingverfahren besonders wichtig, auch die Robustheit der erstellten Verfahren zu berücksichtigen. Dies ist besonders entscheidend, wenn Verfahren auf Basis von sogenannten Trainingsdaten zunächst angepasst und die optimierten Strategien später in anderem Kontext weiterverwendet werden. Für die im weiteren Teil der Arbeit durchgeführten Untersuchungen in Bezug auf die Robustheit der erstellten Grid-Scheduler reichen deshalb die oben beschriebenen Daten in ihrer ursprünglichen Form nicht aus. Für die Untersuchung der Robustheit ist es zum einen notwendig, Trainingsdaten für die Optimierung der Regelbasis zur Verfügung zu haben. Zum anderen werden Anwendungsdaten benötigt, auf denen die Einsatzfähigkeit des Grid-Schedulers getestet werden kann und die *nicht* Teil der Trainingsdaten waren. Somit kann gezeigt werden, wie sich die Verfahren auf ähnlichen, aber bisher unbekannten Jobeinreichungen verhalten werden. Diesem Ansatz liegt die oft bestätigte Annahme zugrunde, siehe zum Beispiel Krampe u. a. [22] oder Tsafrir und Feitelson [174], dass sich die Charakteristiken der eingereichten Jobs einer Nutzergemeinschaft nicht sprunghaft ändern, sondern einmal angepasste und nutzbare parallele Anwendungen auch über einen längeren Zeitraum genutzt und gepflegt werden.
Da allerdings keine zusätzlichen Arbeitslastdaten mit denen eine solche Evaluation durchführbar wäre, verfügbar sind, wurden die vorhandenen Daten entsprechend aufgeteilt. Dabei war es wichtig, für die Anwendung einen längeren Zeitraum zu berücksichtigen als für das Training, siehe Tabelle 7.3. Für das eigentliche Lernen der Regelbasen mittels eines evolutionären Fuzzy-Systems, siehe Kapitel 12, werden im Folgenden die fünf Monate umfassenden Datensätze herangezogen, während die sechs Monate umfassenden Datensätze für

Datensatz	Monate	n	m_k	AWRT	U	C_{max}
KTH-5	5	11.780	100	488.387,49	64,84	13.765.377
KTH-6	6	16.699	100	99.236,27	68,52	16.420.782
CTC-5	5	35.360	430	57.897,77	63,74	13.009.718
CTC-6	6	41.839	430	59.118,15	67,05	16.346.403
SDSC00-5	5	13.494	128	67.717,23	71,28	12.965.572
SDSC00-6	6	16.316	128	413.957,04	73,38	17.002.360
SDSC03-5	5	32.606	1.152	83.718,68	71,38	13.021.659
SDSC03-6	6	32.978	1.152	62.825,76	66,14	16.577.054
SDSC05-5	5	28.184	1.664	56.925,10	45,94	13.078.215
SDSC05-6	6	46.719	1.664	77.463,52	70,97	16.419.455

Tabelle 7.3: Eigenschaften und Referenzergebnisse der aufgeteilten Arbeitslastaufzeichnungen inklusive AWRT in Sekunden, U in % und C_{max} in Sekunden für rein lokale Ausführung und FCFS als LRMS-Schedulingalgorithmus.

die Anwendung verwendet werden. Im Sinne einer späteren praktischen Anwendbarkeit werden also Methoden vorgestellt, die auf Basis von halbjährigen Aufzeichnungen der Jobeinreichungen für die Zukunft gute Schedulingentscheidungen treffen können.

Teil III

Entwicklung und Analyse von dezentralen Grid-Schedulingverfahren

8 Grid-Scheduling mit indirekter Kommunikation

In diesem Kapitel wird eine erste Architektur für den Austausch von Jobs in einem vollständig dezentralen Grid-Schedulingszenario vorgestellt und simulatorisch evaluiert. Der Austausch wird durch die Einführung einer zentralen Austauschplattform, siehe auch Kapitel 6.1.2, realisiert. Obwohl eine zentrale Austauschkomponente nicht Bestandteil einer final angestrebten Architektur sein kann, stellt sie aber doch einen guten Ausgangspunkt für die Untersuchung des Austauschs in einem Grid dar. Sie erlaubt die Realisierung eines Austauschs ohne Entwicklung von weiteren komplexen Strukturen und Algorithmen. Dabei bietet die Plattform zusätzlich den Vorteil, dass an ihr die Charakteristiken der ausgetauschten Jobs zentral beobachtet werden können.
In Abbildung 8.1 ist die hier untersuchte Architektur dargestellt. Neben der natürlich zentral positionierten Austauschplattform, auf die im nächsten Abschnitt genauer eingegangen wird, ist im Gegensatz zum allgemeinen Modell, siehe Abbildung 6.2, hier die strikte Trennung zwischen LRMS und GRMS aufgehoben. Dies ist notwendig, da die untersuchten Algorithmen auf möglichst einfache Art mit der Austauschplattform interagieren sollen. In Abbildung 8.1 wird durch den bidirektionalen Pfeil zwischen GRMS und der Warteschlange, also der Einreichungsschnittstelle des LRMS, verdeutlicht, dass Jobs auf die Plattform migriert und auch von dort aus wieder entfernt werden können. Alle weiteren Entwicklungen im Rahmen dieser Arbeit berücksichtigen jedoch eine vollkommene sowohl logische als auch technische Entkopplung dieser beiden Komponenten, sodass diese Untersuchung nur als erster Schritt angesehen werden kann.

8.1 Entwicklung von Austauschverfahren

Die Austauschplattform ist als gemeinsam nutzbare Warteschlange realisiert, in die alle teilnehmenden Sites Jobs einreichen und entnehmen können. Dabei ist es auch möglich, Jobs an beliebiger Position aus der Warteschlange zu entnehmen, wie es zum Beispiel zur Unterstützung der Backfilling-Algorithmen notwendig ist. Alle eingefügten Jobs werden statisch nach ihren Einreichungszeitpunkten r_j in der Warteschlange verwaltet; das dynamische Umsortieren der Warteschlange ist dabei nicht zulässig, siehe auch Grimme u. a. [15].
Die Interaktion mit der Plattform erfordert die Entwicklung von Austauschverfahren, die entscheiden, ob ein lokal eingereichter Job auf der Plattform angeboten oder lokal ausgeführt werden soll. Eine einfache Möglichkeit, ein derartiges Verhalten zu erreichen besteht darin, die bekannten lokalen Schedulingalgorithmen abwechselnd sowohl auf die lokale Warteschlange als auch auf die Warteschlange der Austauschplattform anzuwenden. Für alle hier

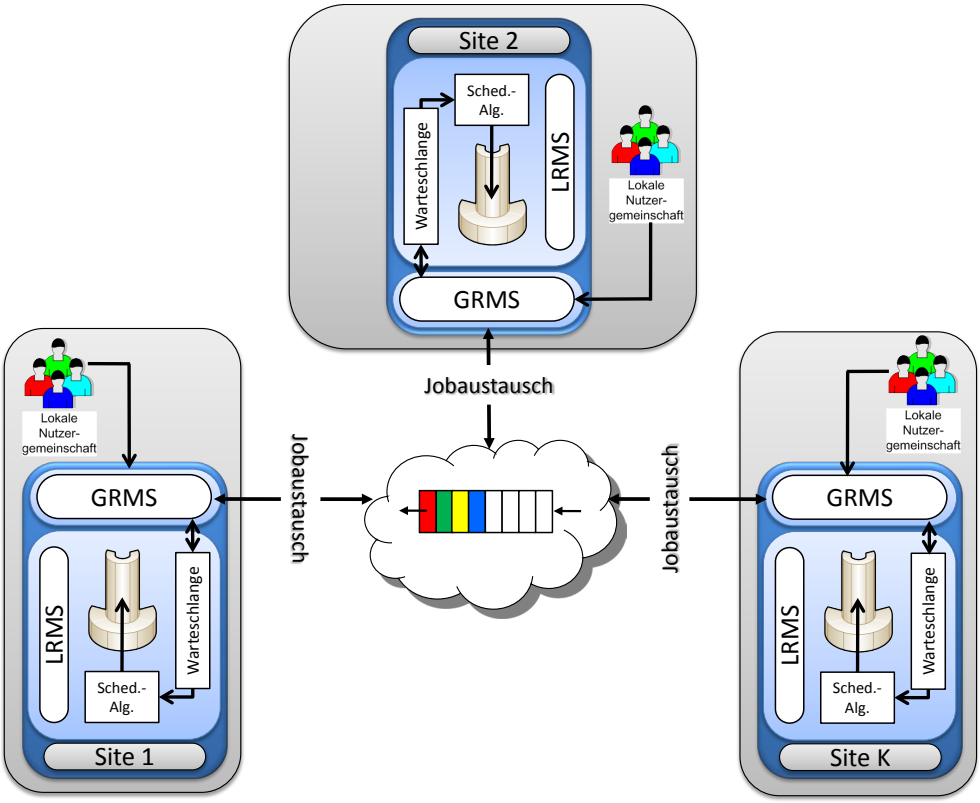


Abbildung 8.1: Dezentrale Grid-Schedulingarchitektur mit zentraler Austauschplattform.

untersuchten Austauschverfahren *(Job-Sharing Algorithms)* wird dazu die folgende allgemeine Regel verwendet:

> *Immer wenn ein Job aus der lokalen Warteschlange nicht unmittelbar auf den lokalen Ressourcen ausgeführt werden kann, wird dieser auf der zentralen Austauschplattform für die entfernte Ausführung angeboten. Dabei wird dieser Job aus der lokalen Warteschlange entfernt.*

Im Folgenden werden die auf Basis dieser Grundregel entwickelten Job-Sharing-(JS) Algorithmen genauer vorgestellt. Es werden drei Algorithmen untersucht, die alle auf Algorithmus 2 als Vorlage beruhen.
Zu Anfang wird bei diesem Konzept eine der vorher zu spezifizierenden Heuristiken FCFS, EASY oder LIST, siehe Kapitel 5.4.2, auf die lokale Warteschlange angewendet, siehe Zeile 1. Danach (siehe Zeile 3), werden alle Jobs, die bereits durch den vorherigen Schritt zwar betrachtet–also auch alle potenziellen Backfilling-Kandidaten–aber nicht gestartet wurden, in die Austauschplattform überstellt. Wenn lokal überhaupt kein Job ausgeführt werden kann, wird derselbe lokale Algorithmus auf die Warteschlange der Austauschplattform angewendet, siehe Zeile 6. Somit sind praktisch in jedem Schritt neben den sowieso in der jeweiligen lokalen Warteschlange zur Auswahl stehenden Jobs auch alle Jobs der Plattform für das LRMS zugreifbar. Folglich ist aufgrund der so angepassten Algorithmen die zuvor schon angesprochene Kopplung zwischen GRMS, LRMS und der Austauschplattform in dieser Architektur sehr eng.

Algorithmus 2: Allgemeine Grundlage für die verwendeten Austauschalgorithmen.

Eingabe: Lokale Warteschlange v, globale Austauschplattform N

```
1 Algorithmus ∈ {FCFS, EASY, LIST} Wende Algorithmus auf v an;
2 forall betrachteten, aber lokal nicht unmittelbar ausführbaren Jobs do
3     Stelle den Job in N zur Verfügung;
4 end
5 if momentan kein Job lokal ausführbar then
6     Wende Algorithmus auf N an;
7 end
```

Für die modifizierte Version von FCFS bedeutet diese Anpassung, dass natürlich auch nur der erste Job in beiden Warteschlangen betrachtet wird. Im Falle des angepassten EASY-Backfilling-Algorithmus hingegen wird das Backfilling komplett getrennt auf beiden Warteschlangen durchgeführt. Die Backfilling-Prozedur, also das Iterieren über die Warteschlange und Auswählen von passenden Jobs, findet nur genau dann statt, wenn dieses Vorgehen lokal zu keinem Start eines Jobs geführt hat. Dabei ist wichtig zu bemerken, dass jeweils nur eine Warteschlange einzeln für das Backfilling betrachtet wird und keine Vereinigungsmenge beider Warteschlangen gebildet wird. Dadurch ist es möglich, dass sich die Backfilling-Bedingung (siehe auch Abschnitt 5.4.2) bei unterschiedlichen Warteschlangen ändert, da jeweils für andere Jobs Reservierungen eingehalten werden müssen. Jobs können also einmal beim lokalen Backfilling in den Pool wandern, weil sie lokal nicht unmittelbar ausführbar sind. Bei der anschließenden Iteration über den Pool könnten diese aber doch gestartet werden, weil der Job an der Spitze der Pool-Warteschlange eine andere Backfilling-Bedingung erzeugt.
Das veränderte LIST-Verfahren iteriert ebenfalls über die Warteschlange, bis ein unmittelbar ausführbarer Job gefunden wurde, und nutzt auf dieselbe Art die Austauschplattform, wenn dies nicht der Fall war. Im Folgenden werden diese modifizierten Varianten entsprechend mit *FCFS-JS*, *EASY-JS* und *LIST-JS* bezeichnet.

8.2 Evaluation

Bevor detaillierte Resultate für verschiedene Bewertungskriterien präsentiert werden, wird zunächst der experimentelle Aufbau beschrieben.

8.2.1 Experimenteller Aufbau

Um die Anwendbarkeit auf verschiedene Szenarien zu untersuchen, wurden acht Konfigurationen auf Basis der bereits eingeführten Arbeitslastaufzeichnungen erstellt, siehe Tabelle 8.1. Die Auswahl und Kombination der Grid-Szenarien stellen eine möglichst breite Evaluationsbasis dar, wobei zunächst nur Grids mit zwei (Szenarien 1-3) beziehungsweise drei (Szenarien 4-8) teilnehmenden Sites untersucht werden. Diese Einschränkung ist deshalb sinnvoll, da es hier primär um eine Verhaltensstudie von dezentralen Grids geht und möglichst viele Effekte genau beobachtet werden sollen. Dies ist in großen Verbänden schwerer möglich und wird erst im späteren Teil der Arbeit durchgeführt. Die Kombinationen sind weiterhin so gewählt, dass sich möglichst lange Zeiträume mit den vorhandenen Da-

Datensatz	n	m	Monate	Grid-Szenario							
				1	2	3	4	5	6	7	8
KTH-11	28.479	100	11	X				X	X	X	X
CTC-11	77.199	430	11						X		
LANL96-11	57.715	1.024	11					X			
LANL96-13	67.043	1.024	13				X				
LANL96-24	110.769	1.024	24		X	X					
SDSC00-11	29.810	128	11	X				X	X	X	
SDSC00-24	54.006	128	24			X					
SDSC03-11	65.584	1.152	11								X
SDSC03-13	78.951	1.152	13				X				
SDSC03-24	176.268	1.152	24		X						
SDSC05-11	74.903	1.664	11							X	X
SDSC05-13	84.876	1.664	13				X				

Tabelle 8.1: Details der genutzten Arbeitslastaufzeichnungen mit den Konfigurationen der untersuchten acht Grid-Szenarien.

ten simulieren lassen. Die simulierte Zeit ergibt sich, wie im Abschnitt 7.1 dargestellt, durch die Länge der kürzesten Arbeitslastaufzeichnungen in einer Simulationskonfiguration. Diese Szenarien repräsentieren Situationen und Konstellationen, wie sie zum Beispiel in der Ressourcenlandschaft des akademischen Sektors anzutreffen sind. Diese Analogien werden im Folgenden kurz in diesem Kontext motiviert, obwohl gleiche Konstellationen auch auf vollkommen anderer, zum Beispiel auch internationaler, Ebene vorzufinden sind.
Szenario 1 repräsentiert dabei den Zusammenschluss zweier kleiner Installationen mit weniger als 128 Prozessoren. Diese sind zum Beispiel häufig innerhalb universitärer Einrichtungen anzutreffen, bei denen an Instituten oder Lehrstühlen vorhandene Clusterinstallationen von 100–150 Rechnern für Kooperationen zusammengeschlossen werden. Das Szenario 2 hingegen repräsentiert den Zusammenschluss zweier großer Rechenzentren mit jeweils mehr als 1000 Prozessoren, die in überregionalen Projekten kooperieren. Im dritten Szenario wird dann wiederum untersucht, wie sich eine kleine Installation im Zusammenschluss mit einem Rechenzentrum, wie es ebenfalls häufig an Universitäten zu finden ist, verhält.
Die anderen fünf Szenarien beziehen sich nun jeweils auf Grids mit drei teilnehmenden Sites: Szenario 4 beschreibt einen überregionalen Verbund von drei großen Rechenzentren als Erweiterung des Szenarios 2. Weiterhin modelliert das Szenario 5 die Situation, dass Ressourcen eines kleinen Instituts in enger Abstimmung mit einem regionalen Rechenzentrum genutzt werden. Im Gegensatz zu Szenario 4 wird in Szenario 6 der Zusammenschluss von drei Institutsrechnern dargestellt, wie er möglicherweise bei einer Forschergruppeninitiative realistisch ist. Dieses Szenario wird wahrscheinlich das am häufigsten realisierte der hier untersuchten Grids sein. Schließlich beschreiben die Szenarien 7 und 8 heterogene Grid-Strukturen, bei denen sich etwa zwei kleine und ein großer Partner (7) oder zwei große und ein kleiner Partner (8) zu einem Grid formieren. Dies ist vor allem deshalb interessant, weil in der Zukunft eher von einer Vielzahl unterschiedlicher Ressourcenanbietern ausgegangen werden kann.

8.2.2 Austausch von Jobs zwischen zwei Sites

In der Tabelle 8.2 sind alle absoluten Ergebnisse der Evaluation für die Grid-Szenarien 1 bis 3 angegeben. Die Kriterien $AWRT_k$, U_k, C_{max} und SA_k sind jeweils für jeden Job-Sharing-Algorithmus angegeben. Weiterhin werden Vergleichswerte dargestellt, die verdeutlichen,

inwieweit sich die durch die neuen Verfahren für Grids erzielte Werte von der rein lokaler Ausführung (siehe Kapitel 7.2) unterscheiden. Allgemein werden Vergleichswerte dabei nach Gleichung 8.1 für beliebige Kenngrößen Λ berechnet.

$$\Delta[\%] = \frac{\Lambda_{\text{lokal}} - \Lambda_{\text{grid}}}{\Lambda_{\text{lokal}}} \cdot 100 \tag{8.1}$$

Dabei werden Vergleiche des jeweiligen lokalen Schedulingalgorithmus zu seiner *eigenen* Job-Sharing-Variante mit Δ gekennzeichnet, während der Vergleich zu dem *jeweils besten* lokalen Algorithmus mit Δ_b bezeichnet werden. Dieser Vergleich ist notwendig, da davon ausgegangen werden kann, dass ein LRMS ohne Grid-Interaktion natürlich auch in optimierter Form eingesetzt wird. Nur wenn die vorgestellten Job-Sharing-Methoden bessere Resultate liefern als ein effizientes LRMS ohne Jobaustausch, ist die Teilnahme an einem Grid aus Betreibersicht sinnvoll. In allen Fällen bedeuten positive prozentuale Angaben für die zeitbezogenen Größen, dass die Werte jeweils kleiner sind und somit eine Verbesserung im Sinne der Zielfunktionsminimierung zu verzeichnen ist.
Um einen systematischen Zugang zu den Ergebnissen in Tabelle 8.2 zu erhalten, werden zunächst die Vergleiche mit dem entsprechenden lokalen Schedulingalgorithmus betrachtet. Für die Konstellation aus zwei kleinen Sites (Szenario 1) werden deutliche Verbesserungen für die AWRT in Bezug auf die rein lokale Ausführung erreicht. Dabei bleibt die Auslastung (U) oder auch das Ressourcenprodukt (SA) nahezu unverändert oder nimmt nur leicht zu. Dies gilt wiederum nicht bei zwei großen Sites (Szenario 2), wo für den FCFS-JS-Algorithmus starke Verschlechterungen der AWRT der Site 1 zu beobachten sind. Die Gründe dafür sind in der Charakteristik der Einreichungen zu suchen: Es ist klar ersichtlich, dass bei rein lokaler Ausführung für den LANL96-Datensatz bereits sehr gute AWRT-Werte erzielt werden, siehe Tabelle 7.2. Dies mag an der schon erwähnten speziellen Eigenschaft liegen, dass dort nur Jobs mit einer Zweierpotenz als Parallelitätsgrad als Einreichung akzeptiert wurden. Dadurch ergibt sich schon eine sehr gute Job-Allokation, die kaum Lücken beinhaltet, sodass diese auch nicht durch Jobs anderer Grid-Partner gefüllt werden können. Im Gegenteil ergibt sich durch zusätzlich aufgenommene Jobs in fast allen Fällen eine deutlich längere AWRT. Als eine Feststellung ergibt sich daraus, dass für Systeme, die aus beliebigen Gründen lokal schon nahezu optimale Allokationen erreichen, auch die Teilnahme an einem Grid-Verbund wenig sinnvoll erscheint. Vielmehr besteht dann die Gefahr, dass durch einfache und statische Austauschstrategien eher noch eine Fragmentierung des lokalen Schedules erzeugt wird.
Ein vollkommen anderes Verhalten ist für einen Verbund aus einer kleiner (SDSC00) mit einer großen (LANL96) Site zu beobachten, siehe Szenario 3. In diesem Falle migriert die kleine Site einen deutlich höheren Prozentsatz, nämlich 15,41 % ihrer gesamten Jobs, zu der großen Site als umgekehrt (nur 2,65 % von LANL96 zu SDSC00). Deshalb ist die AWRT der kleineren Site deutlich verkürzt und ihre Auslastung nimmt um mehr als 26 % ab. Leider ist aus den bereits oben genannten Gründen keine Verbesserung, sondern eine Verschlechterung um mehr als 16 % bei der großen Site zu beobachten. Generell sei hier aber schon vorweggenommen, dass auch die große Site, so sie denn eine ausreichende Auslastung aufweist, in einem derartigen Verbund deutlich profitieren kann. Die Vergleiche der entwickelten Job-Sharing-Algorithmen mit den Ergebnissen für EASY als den besten lokalen Schedulingalgorithmus ergeben strukturell ähnliche Ergebnisse. Zusammenfassend können also die folgenden Erkenntnisse aus diesen ersten Untersuchungen bezüglich Jobaustausch in einem Grid-Verbund gezogen werden:

Szenario	1		2		3	
Daten	KTH-11	SDSC00-11	LANL96-24	SDSC03-24	LANL96-24	SDSC00-24
m	100	128	1.024	1.152	1.024	128
k	1	2	1	2	1	2
FCFS-JS						
$AWRT_k$	12.8043,42	139.380,14	33.487,04	67.584,37	14277,15	40.764,89
U_k	70,12	72,94	64,59	65,44	58,91	55,57
$C_{max,k}$	29.571.802	2.957.587	62.440.050	64.151.102	62.303.391	63.638.145
SA_k	2.073.589.120	2.724.164.663	41.300.491.136	48.359.131.320	37.584.312.832	449.8547.287
Δ $AWRT_k$	69,38	47,72	100,00	59,33	-16,40	98,17
Δ U_k	2,11	0,63	16,31	-10,62	6,07	-26,71
Δ SA_k	2,77	-2,01	16,58	-10,83	6,07	-32,42
Δ $C_{max,k}$	-0,65	2,59	-0,24	0,23	-0,02	7,74
Δ_b $AWRT_k$	-70,37	-89,36	-201,55	4,51	-28,56	63,62
Δ_b U_k	2,04	-2,68	16,31	-11,46	6,07	-32,47
Δ_b SA_k	2,77	-2,01	16,58	-10,83	6,09	-32,42
Δ_b $C_{max,k}$	-0,71	-0,68	-0,24	-0,71	-0,02	-0,07
M_n =	82,86 13,95	17,14 86,05	88,69 9,51	11,31 90,49	97,35 15,41	2,65 84,59
EASY-JS						
$AWRT_k$	66.115,56	59.015,97	13.579,91	54.019,88	12.583,95	41.441,88
U_k	68,95	74,80	63,20	67,24	58,33	60,28
$C_{max,k}$	29.363.626	29.364.791	62.303.135	63.694.187	62.292.428	63.618.367
SA_k	2.024.644.907	2.773.108.876	40.318.878.368	49.340.744.088	37.204.951.541	4.877.908.578
Δ $AWRT_k$	12,03	19,82	-22,28	23,67	-13,32	63,01
Δ U_k	0,34	-0,21	13,79	-9,02	5,02	-26,75
Δ SA_k	0,34	-0,25	13,81	-9,02	5,02	-26,72
Δ $C_{max,k}$	0,00	0,03	-0,02	0,00	0,00	-0,04
Δ_b $AWRT_k$	12,03	19,82	-22,28	23,67	-13,32	63,01
Δ_b U_k	0,34	-0,21	13,79	-9,02	5,02	-26,75
Δ_b SA_k	0,34	-0,25	13,81	-9,02	5,02	-26,72
Δ_b $C_{max,k}$	0,00	0,03	-0,02	0,00	0,00	-0,04
M_n =	79,63 15,04	20,37 84,96	88,69 9,99	13,37 90,01	97,21 17,01	2,79 82,99
LIST-JS						
$AWRT_k$	61.016,52	58.354,18	13.509,25	51.150,19	13.001,56	38.655,67
U_k	68,98	74,84	63,45	67,01	59,11	54,04
$C_{max,k}$	29.363.626	29..339.969	62.317.990	63.694.095	62.303.391	63.588.898
SA_k	2.025.505.127	2.772.248.656	49.172.388.088	49.172.388.088	37.711.932.960	4.370.927.159
Δ $AWRT_k$	24,16	21,70	-15,93	29,17	-11,52	66,75
Δ U_k	0,38	-0,19	14,24	-9,33	6,43	-34,30
Δ SA_k	0,38	-0,28	14,29	-9,33	6,45	-34,33
Δ $C_{max,k}$	0,00	0,08	-0,04	0,00	-0,02	0,05
Δ_b $AWRT_k$	18,82	20,72	-21,65	27,73	-17,08	65,50
Δ_b U_k	0,38	-0,16	14,24	-9,33	6,43	-34,33
Δ_b SA_k	0,38	-0,28	14,29	-9,33	6,45	-34,33
Δ_b $C_{max,k}$	0,00	0,12	-0,04	0,00	-0,02	0,00
M_n =	77,77 19,61	22,23 80,39	83,20 11,35	16,80 88,65	96,62 14,88	3,38 85,12

Tabelle 8.2: Ziel- und Bewertungsfunktionswerte für die Grid-Szenarien 1–3 mit jeweils zwei Teilnehmern ($K = 2$). $AWRT_k$ und $C_{max,k}$ sind jeweils in Sekunden angegeben, während U_k, SA_k und alle vergleichenden Bewertungsfunktionen (Δ, Δ_b und die Matrix M_n) in % gegeben sind.

1. Die Anpassung des lokalen Schedulingalgorithmus in Form einer Job-Sharing Variante und die Einführung einer zentralen Austauschplattform führen zu teilweise deutlichen Verbesserungen der AWRT. Dazu müssen die teilnehmenden Sites allerdings ein lokales Ressourcenproblem haben oder zumindest eine sehr hohe durchschnittliche Auslastung aufweisen. Vorzugsweise bei Teilnehmern mit sehr ähnlicher Größe

werden durch diesen Ansatz gute Ergebnisse auf beiden Sites erreicht. Ein gutes Verständnis für die genaue Dynamik und Struktur des Problems bei unterschiedlichen Sitegrößen und die Identifikation des durch Grid-Computing erreichbaren Verbesserungpotenzials sind für die Weiterentwicklung der Schedulingverfahren unerlässlich. Deshalb werden Lösungen für dieses Problem in Kapitel 10 vorgestellt.

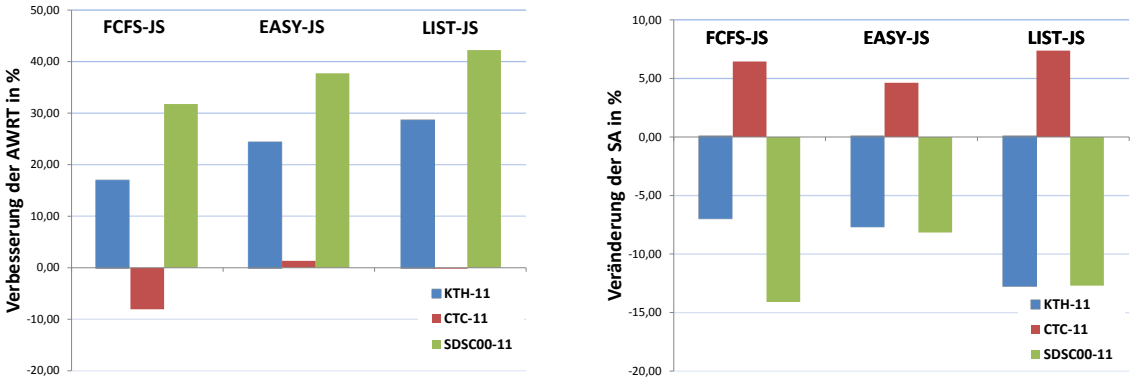
2. Die Kombination von einer stark ausgelasteten mit einer wenig ausgelasteten Site führt zu einem einseitigen Austauschverhalten, das innerhalb des Grid nicht ausgeglichen werden kann. Dies hat in allen untersuchten Fällen starke Verschlechterungen der AWRT auf der wenig ausgelasteten Site zur Folge.

3. Für ungleich große Paarungen wird eine deutlich höhere Anzahl von Jobs von der kleinen zur großen Site migriert. Umgekehrt kann nur eine sehr kleine Anzahl von Jobs auf den kleinen Sites zur Ausführung gebracht werden. Dies mag zum einen daran liegen, dass kleine Sites oftmals gar nicht über genügend Ressourcen zur Ausführung der auf der großen Site eingereichten Jobs verfügen ($m_j > m_k$). Auf der anderen Site ergeben sich nur selten Lücken, die mit großen Jobs der Rechenzentren gefüllt werden können. Umgekehrt ist dies jedoch häufig der Fall, da die Auslastungslücken in den Rechenzentren durch kleine eingereichte Jobs der Institutsinstallationen aufgefüllt werden.

Als Nächstes werden nun die Grid-Szenarien mit drei Teilnehmern untersucht.

8.2.3 Austausch von Jobs zwischen drei Sites

Für die Darstellung der Ergebnisse bei drei miteinander kooperierenden Sites wird hier auf eine detaillierte Auflistung der Absolutwerte verzichtet. Vielmehr werden mittels Diagrammen die Verbesserungen und Verschlechterungen der einzelnen Kriterien verdeutlicht. Die hier diskutierten Diagramme zeigen alle die AWRT-Verbesserungen in Abbildung 8.2(a) und die Veränderungen der SA in Abbildung 8.2(b) verglichen mit EASY als lokalem Schedulingalgorithmus.

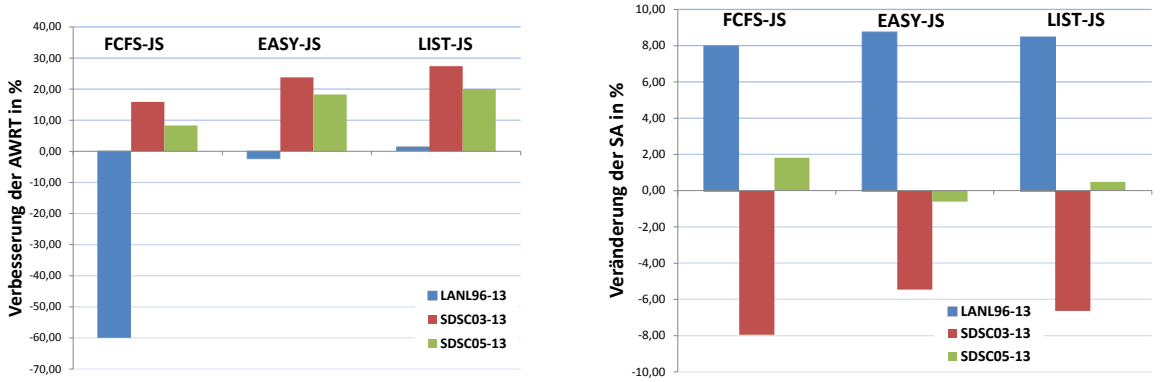


(a) Verbesserung der AWRT in %

(b) ΔSA_b in %

Abbildung 8.2: AWRT und Δ_bSA für Grid-Szenario 6 mit den Sites KTH-11 ($m_1 = 100$), CTC-11 ($m_2 = 430$) und SDSC00-11 ($m_3 = 128$). Dargestellt sind alle drei Austauschalgorithmen FCFS-JS, EASY-JS und LIST-JS.

Es ist auch mit drei teilnehmenden Sites ersichtlich, dass deutlich kürzere AWRT-Werte an allen Sites erreicht werden. Die Auslastung auf der jeweils größten Site, siehe zum Beispiel CTC in Abbildung 8.2(b), erhöht sich jedoch. Dies ist wiederum auf den sich einstellenden Lastausgleich zwischen den Teilnehmern zu erklären. Erstaunlich ist dabei allerdings, dass für EASY-JS und LIST-JS ohne Verschlechterung der AWRT erreicht wird. Dies ist auf die unterschiedlichen Charakteristiken der Einreichungen jeder Nutzergemeinschaft zurückzuführen, die in einem dezentralen Schedulingansatz eben grundsätzlich erhalten bleiben und nicht durch einen Meta-Scheduler schon vor der eigentlichen Schedulingentscheidung vermischt werden. Allerdings stellt sich dieses Verhalten vollkommen automatisch ein und unterliegt keiner steuernden Kontrolle; eine Verbesserung der AWRT kann also mit diesem Verfahren in keiner Weise garantiert werden.
Die Verteilung der Jobs in einem Grid führt zu einer Verringerung der Last an mehreren Sites, während mindestens eine Site mehr Arbeit zu verrichten hat. Dies bedeutet aber durch die dort vorhandenen Überkapazitäten zunächst keine AWRT-Verschlechterung, was als ein weiterer Vorteil von Grid-Kooperation interpretiert werden kann: Durch die Grid-Teilnahme eröffnet sich an weniger ausgelasteten Sites für die Betreiber die Möglichkeit, vorhandene Mehrkapazitäten durch zusätzliche Aufträge auszunutzen. Dies kann zum Beispiel durch das Einwerben weiterer Kunden oder durch die Lockerung von eventuell eingeführten Quotierungen auf Einreichungen geschehen. Es setzt andererseits natürlich voraus, dass zwischen den teilnehmenden Sites eine grundsätzliche Kooperationsvereinbarung getroffen wurde und die Effizienz des Gesamtsystems hierbei erklärtes Gemeinschaftsziel ist.

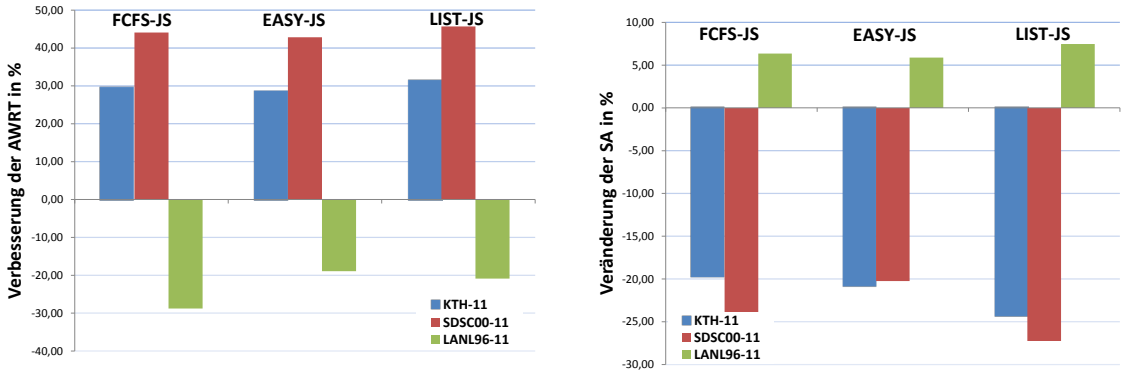


(a) Verbesserung der AWRT in %

(b) ΔSA_b in %

Abbildung 8.3: AWRT und Δ_bSA für Grid-Szenario 4 mit den Sites LANL96-13 ($m_1 = 1.024$), SDSC03-13 ($m_2 = 1.152$) und SDSC05-13 ($m_3 = 1.664$). Dargestellt sind alle drei Austauschalgorithmen FCFS-JS, EASY-JS und LIST-JS.

Gleiche Effekte werden auch bei der simulierten überregionalen Kooperation zwischen grossen Rechenzentren, siehe Abbildung 8.3, beobachtet. Die schon oben erwähnten Besonderheiten in Bezug auf die Einreichungen an der LANL96-Site beeinflussen allerdings auch hier wieder deutlich die Ergebnisse und lassen nur wenig Verbesserungen für diese Site zu. Obwohl für FCFS-JS die AWRT deutlich verschlechtert ist, werden doch für EASY-JS und LIST-JS unveränderte beziehungsweise leicht verbesserte AWRT-Werte gemessen. Eine weitere Begründung dafür ist in dem Einfluss der Backfilling-Varianten auf den Austausch zu finden, auf den in Abschnitt 8.2.4 noch genauer eingegangen wird.

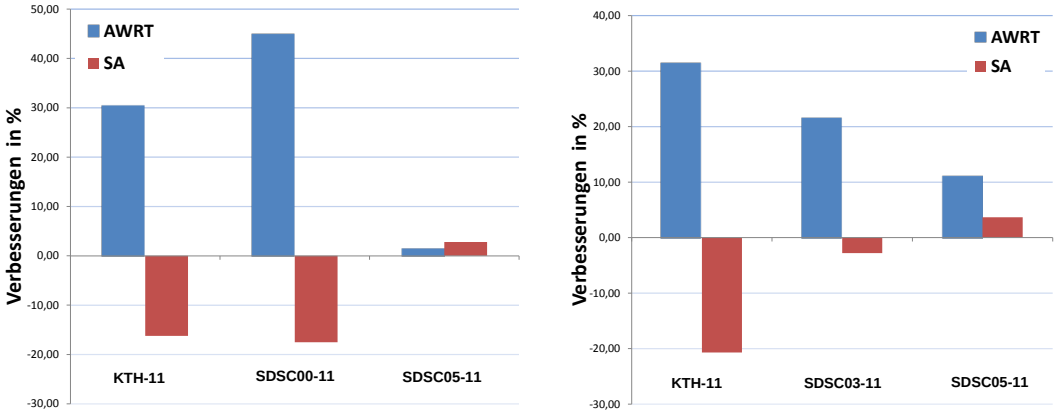


(a) Verbesserung der AWRT in %

(b) ΔSA_b in %

Abbildung 8.4: AWRT und Δ_bSA für Grid-Szenario 5 mit den Sites KTH-11 ($m_1 = 100$), SDSC00-11 ($m_2 = 128$) und LANL96-11 ($m_3 = 1.024$). Dargestellt sind alle drei Austauschalgorithmen FCFS-JS, EASY-JS und LIST-JS.

Analog zu dem Szenario 3 mit zwei teilnehmenden Sites zeigen die Ergebnisse in Abbildung 8.4 für das Szenario 5, bei der zwei kleine und eine große Site kooperieren, deutliche Verbesserungen der AWRT an den kleinen Sites, erreicht auf Kosten der großen Site. Dies liegt dies wiederum daran, dass viele Jobs von den kleinen Sites mit einer insgesamt hohen Auslastung zu der großen Site mit einer relativ geringen Auslastung verschoben werden. In der Konsequenz führt es dann zu einer deutlichen Verschlechterung für die Antwortzeiten an der großen Site.



(a) AWRT und SA_b in % für Szenario 7

(b) AWRT und SA_b in % für Szenario 8

Abbildung 8.5: AWRT und Δ_bSA für die Grid-Szenarien 7 und 8. Dargestellt sind die Ergebnisse für den Austauschalgorithmus LIST-JS im Vergleich zu reinen lokalem Scheduling mit EASY.

Abschließend werden in Abbildung 8.5 die Ergebnisse für die beiden heterogenen Grid-Szenarien 7 und 8 präsentiert. Dabei ist jeweils nur LIST-JS als der beste Algorithmus im Vergleich zu rein lokalem Scheduling mit EASY-Backfilling dargestellt. Es ist hier deutlich

erkennbar, dass sich die AWRT bei allen Teilnehmern teilweise sehr deutlich, wie zum Beispiel beim SDSC00 mit über 40 % in Abbildung 8.5(a), verbessert. Besonders erstaunlich ist, dass selbst für die sehr große SDSC05-Site in Szenario 7 mit 1.664 Prozessoren noch 3 % AWRT-Verbesserung erzielt werden, obwohl sogar ihre Auslastung leicht steigt. Gleiches gilt auch für Szenario 8, siehe Abbildung 8.5(b), bei dem sich nicht nur die kleine KTH-Site, sondern auch die beiden großen Sites deutlich verbessern. Dies ist wohl nicht nur auf die Interaktion zwischen den beiden großen Partnern, sondern auch zum Teil auf die Nutzung der kleinen Site zurückzuführen. Es stellt sich hier heraus, dass sich die lokal schon sehr effizient arbeitende Site LANL96 nicht als Repräsentant einer allgemeinen großen Site eignet. Die Untersuchungen wurden mit LANL96 deshalb durchgeführt, um die Auswirkungen der Auslastungssituation ebenfalls zu berücksichtigen. Auch wenn in einigen Szenarien der Eindruck entsteht, dass die Teilnahme an einem derartigen Grid für große Sites nur nachteilig ist, so zeigen doch die letzten beiden Ergebnisse, dass bei genügend hoher lokaler Auslastung auch große Sites Vorteile von der Grid-Interaktion haben können. Im nächsten Abschnitt werden die Ursachen dafür und die Dynamik dieses Austauschs noch weiter untersucht.

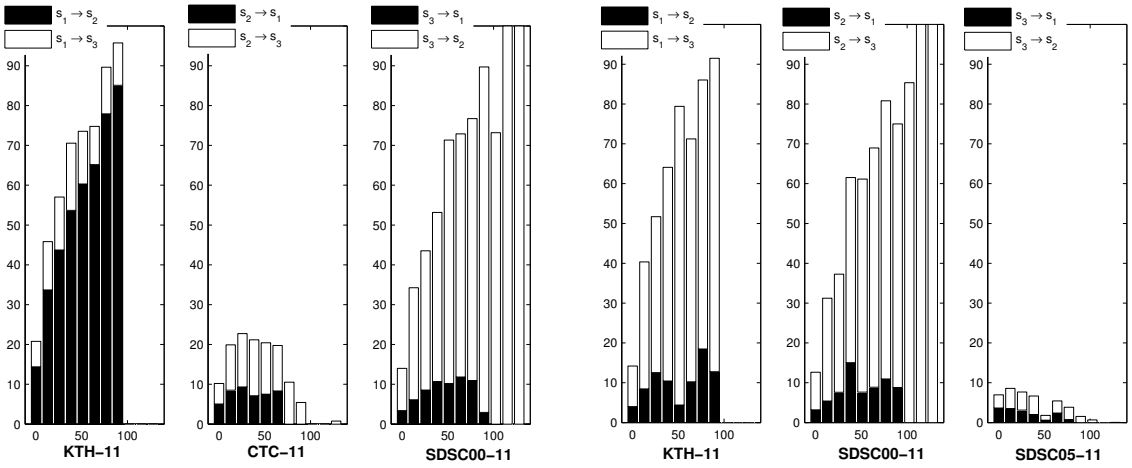
8.2.4 Analyse des Jobaustauschs

Die Tatsache, dass hier für alle Teilnehmer Verbesserungen der AWRT erzielt werden, obwohl teilweise die jeweilige lokale Auslastung noch steigt, ist primär auf die unterschiedlichen Charakteristiken der Einreichungen und die durch die zentrale Plattform erreichbare Vermischung der Jobs zurückzuführen. Inwieweit nun wirklich Jobs migriert werden, soll an zwei Beispielen, nämlich Szenario 5 und 7, verdeutlicht werden. In Tabelle 8.3 sind dazu die Ergebnisse für LIST-JS im Vergleich zu EASY angegeben. Zusätzlich sind die beiden in Kapitel 7.4 bereits eingeführten Matrixdarstellungen für die Migration der Jobs in Hinblick auf Jobanzahl und auf migriertes Ressourcenprodukt mit aufgenommen und die Veränderung der Gesamtauslastung des Grid (U_o) wird ebenfalls mit berücksichtigt. Diese beiden gewählten Grid-Szenarien sind repräsentativ für alle anderen Szenarien und die wichtigsten Effekte lassen sich an dieser Auswahl gut zeigen.
Bei der Betrachtung der Gesamtauslastung fällt sofort auf, dass sie nahezu unverändert bleibt oder leicht zunimmt. Das gesamte Grid wird also im Hinblick auf die Auslastung nicht effizienter, aber eine bessere Lastverteilung wird unter den Partnern erreicht. Damit es allerdings zu so einer verbesserten Lastverteilung bei Anwendung derart einfacher Austauschstrategien kommt, müssen möglichst viele Jobs in der zentralen Austauschplattform angeboten und für die Allgemeinheit zur Verfügung gestellt werden. Dies ist offensichtlich bei der Anwendung von EASY-JS oder LIST-JS eher der Fall, da dort potenziell in jedem Schritt die gesamte Warteschlange durchlaufen wird und auf der Plattform angeboten wird. Somit ist dann auch die Auswahlmöglichkeit für andere Sites größer. Bei FCFS-JS wird ja jeweils nur ein Job in jedem Schritt betrachtet und auch nur ein Job von der Plattform entnommen, wodurch der Anteil der ausgetauschten Jobs sehr viel geringer ist.
Ein weiteres wichtiges Resultat betrifft die Charakteristiken der migrierten Jobs. Aus den Migrationsmatrizen M_n und M_{SA} in Tabelle 8.3 ist zu schließen, dass große Sites dazu tendieren, möglichst kleine Jobs abzugeben, während kleine Sites möglichst große Jobs, das heißt solche mit vielen angeforderten Prozessoren, auf der Plattform anbieten. Dies wird klar, wenn das Migrationsverhalten zwischen Site 1 und 3 im Grid-Szenario 7 betrachtet wird: Site 1 migriert ungefähr 20 % ihrer Jobs zu anderen Teilnehmern, was allerdings insge-

	Grid-Szenario 5			Grid-Szenario 7		
Daten	KTH-11	CTC-11	SDSC00-11	KTH-11	SDSC00-11	SDSC05-11
k	1	2	3	1	2	3
m_k	100	430	128	100	128	1.664
AWRT_k	53.667,36	53.030,53	42.517,13	52.272,68	40.556,50	54.048,42
U_k	59,96	70,54	65,54	57,56	61,83	61,85
$\Delta\ \text{AWRT}_k$	28,59	-0,17	42,24	30,45	44,90	1,65
$\Delta\ \text{U}_k$	-12,75	7,36	-12,57	-16,23	-17,52	2,80
$\Delta\ \text{SA}_k$	-12,75	7,37	-12,70	-16,23	-17,67	2,79
$\Delta\ \text{U}_o$	0,0034	—	—	0,037	—	—
$M_n =$	73,93	18,75	7,32	80,30	4,94	14,75
	5,73	87,74	6,53	4,09	80,51	15,40
	4,42	16,97	78,61	2,82	3,37	93,81
$M_{SA} =$	53,30	35,1	11,56	57,32	8,13	34,55
	5,47	86,99	7,54	5,75	57,34	36,91
	8,35	35,2	56,45	1,27	1,81	96,92

Tabelle 8.3: Detaillierte Ergebnisse für die Grid-Szenarien 5 und 7 mit jeweils drei Sites und der LIST-JS. Die Ergebnisse sind im Vergleich zum lokalen EASY Backfilling dargestellt. Zusätzlich ist das Migrationsverhalten in Form der Matrizen M_n und M_{SA}, siehe Kapitel 7.4, angegeben. Die AWRT_k ist jeweils in Sekunden und alle übrigen Bewertungsfunktionen sind in % angegeben.

samt über 40 % der gesamten auf Site 1 eingereichten Arbeitslast entspricht. Das gegenteilige Verhalten wird für Site 3 beobachtet, die ungefähr 5 % ihrer Jobs zu anderen Partnern migriert. Dies macht dort aber nur einen Anteil von ungefähr 3 % ihrer gesamten Arbeitslast ausmacht. Das gleiche Verhalten ist, allerdings in kleinerem Umfang, ebenfalls für das andere untersuchte Szenario festzustellen.



(a) Grid-Szenario 5 (b) Grid-Szenario 7

Abbildung 8.6: Relativer Anteil der *migrierten* Jobs für die beiden Grid-Szenarien 5 und 7. Die Werte sind unterteilt nach dem Parallelitätsgrad (m_j) und beziehen sich jeweils auf das Vorkommen im Originaldatensatz als Anteile in %.

Um die Aufteilung der abgegebenen Jobs noch weiter zu verdeutlichen, ist in Abbildung 8.6 ein Histogramm der abgegebenen Jobs, unterteilt nach deren Parallelitätsgrad (m_j), gezeigt. Die prozentualen Werte beziehen sich dabei auf die Menge von Jobs mit dem gleichen Par-

allelitätsgrad in der ursprünglichen Einreichungsmenge. Wird also zum Beispiel auf einer Site nur ein Job mit einer bestimmten Anzahl Prozessoren eingereicht und genau dieser eine Job entfernt zur Ausführung gebracht, so entspricht dies einem prozentualen Anteil von 100 %. Es werden dort also nur die abgegebenen Jobs berücksichtigt und lokal ausgeführte Jobs nicht mit dargestellt.
In Abbildung 8.6(a) wird klar deutlich, dass große Sites mehr Jobs von kleinen annehmen, als große Sites an kleine abgeben. Die teilweise drastischen Verbesserungen der AWRT resultieren also offenbar von eine deutlich besseren Ausnutzung der möglichen Lücken in den rein lokalen Schedules. In Abbildung 8.6(b) ist zusätzlich zu erkennen, dass die große Site einen geringen Anteil von Jobs ebenfalls zu den kleineren Partnern umverteilt, was wiederum zu nachweisbaren AWRT-Verbesserungen auf der großen Site führt. Es ist also festzuhalten, dass ein reger Austausch zu stärkeren Verbesserungen führt. Weiterhin kann schon eine effiziente Umverteilung einer geringen Menge von Jobs durchaus starke Verbesserungen der AWRT herbeiführen.

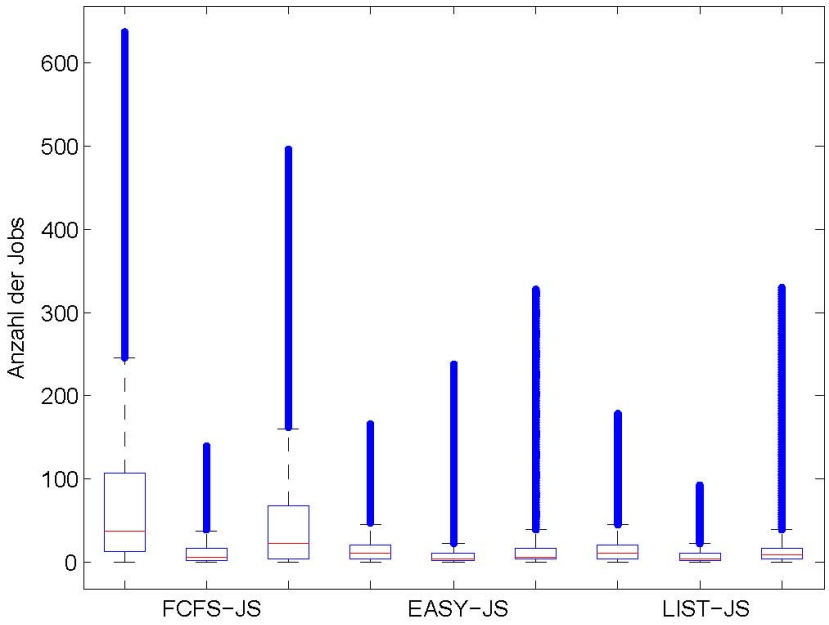


Abbildung 8.7: Box-Diagramm der Größe der Austauschplattform während eines Grid-Betriebs mit zwei teilnehmenden Sites und Anwendung der drei Job-Sharing-Algorithmen. Für jeden Algorithmus sind drei Box-Diagramme abgebildet: Von links nach rechts jeweils KTH-11/SDSC00-11, LANL96-24/SDSC03-24 und LANL96-24/SDSC00-24.

Zum Abschluss der Untersuchung dieses ersten Grid-Szenarios wird die Länge der zentralen Warteschlange während des Grid-Betriebs noch genauer betrachtet. Dazu wurde deren Länge immer dann bestimmt, wenn Jobs zu der Plattform hinzugefügt oder aus ihr entnommen wurden. Die Box-Diagramme in Abbildung 8.7 zeigen jeweils den Median, die 25 % und 75 % Quantile, sowie die 5 % und 95 % Quantile (am Ende der sogenannten Bärte) der gemessenen Daten. Es ist zunächst nachgewiesen, dass die Austauschplattform nicht überfüllt wird und es nicht zu einem ewigen Verweilen von Jobs in der Plattform *(starvation)* kommt. Jobs werden also nur für kurze Zeit auf der Plattform angeboten, bevor sie entwe-

der entfernt oder wieder lokal zur Ausführung gebracht werden. Deutlich ist weiterhin die oben schon angesprochene Eigenschaft von FCFS-JS zu erkennen: Auf der Plattform sammeln sich bei dessen Anwendung vergleichsweise viele Jobs an, was auf den weniger regen Jobaustausch zurückzuführen ist.
Nach der ausführlichen Besprechung dieser dezentralen Grid-Schedulingarchitektur mit indirekter Kommunikation folgt nun die Umsetzung eines vollständig dezentralen Ansatzes mit direkter Kommunikation.

9

Grid-Scheduling mit direkter Kommunikation

BEIM ZUVOR geschilderten Ansatz wurden nur Jobs zum Austausch verfügbar gemacht, die von den Partnern bewusst dazu freigegeben wurden. Das Plattform-Konzept stellt also eine vorrangig egoistisch orientierte Grid-Sicht dar, weil besonders die Jobs der Allgemeinheit angeboten werden, für die lokal gerade keine Verwendung gefunden werden kann. Die Hoffnung besteht darin, dass die Gemeinschaft diese Aufgaben übernimmt und als Gegenleistung von anderen Sites speziell passende Jobs übernommen werden können. Die Analyse in Kapitel 8 zeigt jedoch, dass dieses Verfahren nur bei einem generell sehr regen Austausch Vorteile bietet.

Der nächste Schritt auf dem Weg zu einem vollständig entkoppelten und dezentralen Schedulingsystems überführt nun das Konzept des Austauschs mittels zentraler Plattform in ein vollständig dezentrales Szenario. Die zentrale Plattform wird dabei zwar als Komponente aufgegeben, aber logisch in die Warteschlangen der einzelnen Partner verlagert und durch eine ständige Kommunikation zwischen allen Teilnehmern zugreifbar gemacht. Dadurch wird dann erstmals eine *direkte* Kommunikation zwischen den Partnern ermöglicht.

Als Umsetzung einer vorrangig altruistischen Verhaltensweise im Grid wird hier ein Ansatz vorgestellt, bei dem zu jedem Zeitpunkt die Sites Jobs anfordern, die dazu genutzt werden, möglichst viele lokale Ressourcen auszulasten. Dabei werden weder präferierte Bindungen zwischen Grid-Sites berücksichtigt noch irgendwelche Lastenverhältnisse in die Entscheidung einbezogen. Dazu ist es allerdings notwendig, alle Informationen über die gerade lokal verfügbaren Jobs global zu kommunizieren. Die Informationspolitik ist nur insofern restriktiv, als dass keine weiteren Informationen über den jeweiligen Systemzustand veröffentlicht werden müssen. Generell setzt dieser Ansatz jedoch ein starkes Vertrauensverhältnis zwischen allen Partnern voraus. Ein derartiges Vorgehen ist von dem in der Literatur als „Workstealing“ bezeichneten und im Bereich des Prozessor-Schedulings häufig verwendeten Verfahren inspiriert, siehe zum Beispiel Blumofe und Leiserson [33].

Weiterhin kann dieses Verfahren nur effizient funktionieren, wenn der lokale Schedulingalgorithmus in der Lage ist, Jobs durch ein Backfilling-Verfahren auszuwählen und diese in den lokalen Schedule einzubauen. Wie die Analysen gezeigt haben, kann durch ein reines FCFS-Verfahren nur ein unzureichender Jobaustausch erreicht werden, da in jedem Zeitschritt immer nur ein einzelner Job betrachtet wird. Deshalb wird die Autonomie der jeweiligen Administratoren insofern eingeschränkt, als dass die Anwendung eines Backfilling-Verfahrens unabdingbar ist. In der hier dargestellten Untersuchung, die auch bei Grimme u. a. [16] zu finden ist, wird ausschließlich EASY-Backfilling als Algorithmus im LRMS eingesetzt.

9.1 Austauschstrategie mit aktiver Jobanfrage

Als Ausgangspunkt für die Untersuchungen dient das vollständig dezentrale Grid-Modell, wie es bereits in Kapitel 6.1.2 und Abbildung 6.2 dargestellt wurde. Allgemein kann die hier beschriebene Strategie in zwei Teilentscheidungen separiert werden, die im Folgenden als *Politiken* bezeichnet werden:

- Die *Akzeptanzpolitik* beschreibt das Verhalten einer Site im Hinblick auf die Annahme von eingereichten Jobs. Dabei wird zunächst nicht unterschieden, ob diese Jobs von einer lokalen Nutzergemeinschaft eingereicht oder von externen Grid-Teilnehmern dem System angeboten wurden. Eine Unterteilung dieser beiden Mengen ist zwar prinzipiell möglich, wird hier aber zunächst nicht berücksichtigt; alle dem GRMS angebotenen Jobs werden gleich behandelt.

- Die *Distributionspolitik* beschreibt hingegen, welche der lokal ins System eingegebenen Jobs für eine entfernte Ausführung infrage kommen. Dies kann bedeuten, dass diese Jobs bewusst anderen Partnern zur Ausführung angeboten werden können. Alternativ bestimmt die Distributionspolitik auch die Auswahl der Jobs, die bei externer Anfrage an die anfragende Site zurückgegeben werden. Dabei obliegt der Ursprungs-Site bis zuletzt ein Vetorecht über die Abgabe eines Jobs, auch wenn bereits eine Übereinkunft nach Verhandlung erreicht wurde.

Die Akzeptanzpolitik der hier angewendeten Austauschstrategie ist schematisch in Abbildung 9.1 dargestellt und arbeitet nach dem folgenden Verfahren: Alle von der jeweiligen lokalen Nutzergemeinschaft eingereichten Jobs werden wie üblich an das Ende der Warteschlange angefügt ①. Immer, wenn nun der erste Job in der Warteschlange nicht unmittelbar ausgeführt werden kann,[12] wird eine Anfrage an die anderen Teilnehmer nach dort schon eingereichten aber noch nicht ausgeführten Jobs aktiv gestellt ②. Hier wird mittels der jeweiligen Distributionspolitik eine Menge von Jobs für eine entfernte Ausführung erstellt. Die angefragte Site übermittelt ihre ausgewählten Jobbeschreibungen an die anfragende Site, die dann wiederum mittels des EASY-Algorithmus Jobs zur Übernahme auswählt. Das heißt, es wird überprüft, ob einer der fremden Jobs unmittelbar gestartet werden kann ③ und dabei der geschätzte Startzeitpunkt des lokal an erster Stelle stehenden Job nicht verzögert wird (EASY-Bedingung, siehe auch Kapitel 5.4.2). Ist so ein Job gefunden und gibt die ursprünglich verantwortliche Site ihr Einverständnis zur Übernahme des Jobs durch die anfragende Site, dann wird der reale Job migriert, in die lokale Warteschlange eingefügt ④ und dort in das lokale Schedulingverfahren einbezogen. Die Prozedur wird sequenziell für alle angebotenen Jobs und für alle anderen Grid-Sites durchgeführt ⑤. Erst wenn alle Teilnehmer angefragt und alle jeweils angebotenen Jobs überprüft wurden, beginnt der nächste reguläre lokale Schedulingschritt. Es sei noch bemerkt, dass bei diesem Verfahren einmal migrierte und dann nicht direkt gestartete Jobs natürlich im nächsten Schritt auch wieder anderen Sites angeboten werden. Dieses Konzept wird aus folgenden Gründen so umgesetzt: Zunächst würde beim unmittelbaren Start von migrierten Jobs (und diese Jobs sind direkt ausführbar, da sie nach genau diesem Kriterium ausgewählt wurden) das lokale Schedulingkonzept vollständig außer Kraft gesetzt, da so zum Beispiel im Rahmen von EASY-Backfilling keine

[12] Das Intervall der Überprüfungen kann natürlich im Rahmen spezieller LRMS-Konfiguration auch nach bestimmten festen Abständen gewählt sein.

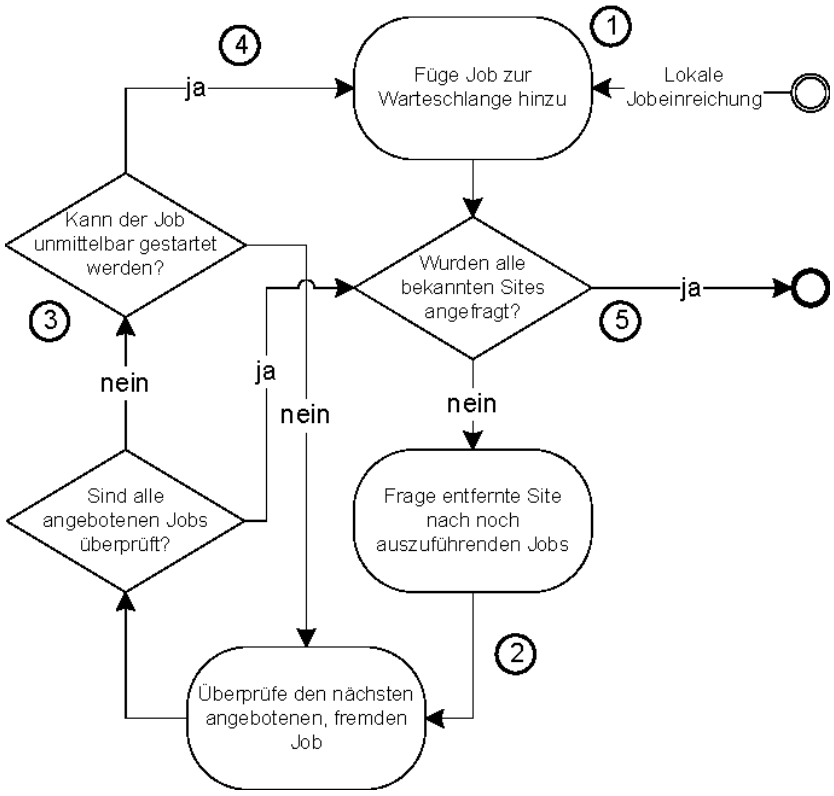


Abbildung 9.1: Ablaufdiagramm der Anfrage-basierten Austauschstrategie.

Garantien mehr für die Startzeit des ersten Job gegeben werden könnten.[13] Weiterhin ist es durch die mehrfache Migration möglich, einmal getroffene Entscheidungen zu revidieren und andere Sites zu finden, bei denen die Jobs schneller zur Ausführung gebracht werden können. Dadurch wird insgesamt ein reger Jobaustausch erzeugt, was, wie schon im Kapitel 8 gezeigt, vorteilhaft für das gesamte Grid ist.

Im Hinblick auf die Autonomie der teilnehmenden Sites ist es noch wichtig zu bemerken, dass die jeweils angebotene Menge von aktuell wartenden Jobs natürlich von der angefragten Site durch die Distributionspolitik beliebig eingeschränkt werden kann. Eine Site kann dabei zum Beispiel nur hochgradig parallele Jobs ($m_j \approx m_k$) für die entfernte Übernahme anbieten, während sequenzielle Jobs immer lokal gerechnet werden. Weiterhin können „geheime“ Jobs durch ein derartiges Verfahren für die Allgemeinheit ausgeblendet werden. Da aber in dieser Arbeit für alle Jobs keine Ressourcen- oder Maschinenrestriktionen *(machine eligibility constraints)* berücksichtigt werden, sind bei jeder Anfrage alle momentan wartenden Jobs für die entfernte Ausführung auswählbar. Es wird lediglich eine Vorfilterung nach der maximalen Anzahl verfügbarer Ressourcen vorgenommen (m_k) um sicherzustellen, dass alle migrierten Jobs auf der anfragenden Site ausführbar sind.

[13] Wahrscheinlich wird beim Einsatz von EASY der gerade ausgewählte Job auch direkt gestartet, da nach der Migration lokal über die gesamte Warteschlange iteriert wird und damit auch der migrierte Job am Ende der Warteschlange berücksichtigt wird. Die Ordnung der Warteschlange bleibt dabei allerdings bestehen, was die korrekte Anwendung von EASY sicherstellt.

9.2 Evaluation des Austauschs durch aktive Anforderung

Ähnlich wie im vorherigen Kapitel wird auch hier nun das vorgestellte Konzept anhand von realen Arbeitslastaufzeichnungen und exemplarischen Grid-Szenarien auf seine Einsatzfähigkeit hin evaluiert. Dabei werden wiederum die in Abschnitt 7.4 eingeführten Bewertungskriterien genutzt.
In Tabelle 9.1 sind die drei untersuchten Szenarien dargestellt, wobei die Szenarien 5 und 8 bereits im vorherigen Kapitel für die Untersuchung herangezogen und dort schon näher beschrieben wurden. Zusätzlich wird hier ein größerer Verbund, nämlich das Szenario 9, mit insgesamt fünf Teilnehmern berücksichtigt. Allgemein wird davon ausgegangen, dass sich die Problemstellung in einem Grid mit mehr Teilnehmern eher vereinfacht und bessere Antwortzeiten für die Jobs zu erwarten sind. Dies liegt darin begründet, dass zwar mehr Jobs in das System eingereicht werden, trotzdem aber durch die größere Menge an zur Verfügung stehenden Ressourcen eine bessere Allokation ermöglicht wird. Mit anderen Worten gesagt finden sich bei mehreren Teilnehmern eher freie Lücken in den Schedules als bei wenigen Grid-Teilnehmern. Trotzdem ist für diese Evaluation das Szenario 9 insofern von Bedeutung, als dass bei dem hier vorgestellten Ansatz latent die Gefahr von Jobzirkulationen entsteht: Da die Jobs immer jeweils zwischen den Warteschlangen wechseln können, siehe Abschnitt 9.1, kann es zu einem ständigen Verweilen von Jobs kommen. Diese Gefahr ist umso größer, je mehr aktiv anfragende Sites sich im Grid befinden.

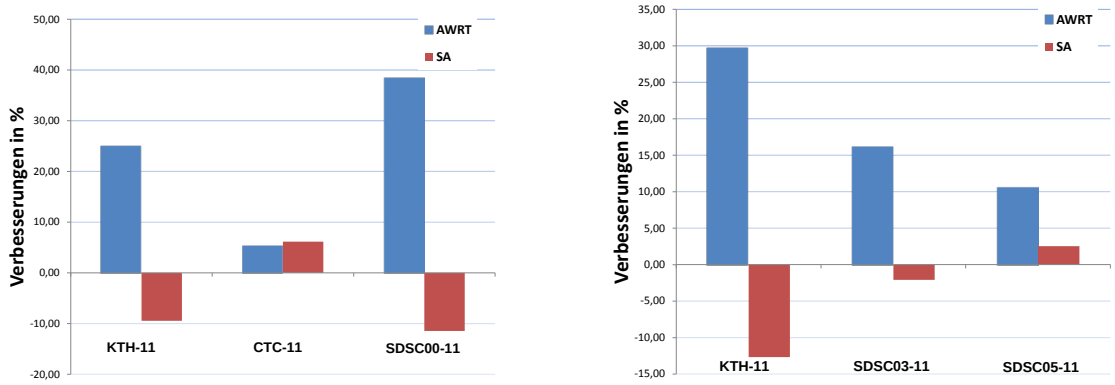
Daten	n	m	Monate	Grid-Szenario		
				5	8	9
KTH-11	28.479	100	11	X	X	X
CTC-11	77.199	430	11	X		X
SDSC00-11	29.810	128	11	X		X
SDSC03-11	65.584	1.152	11		X	X
SDSC05-11	74.903	1.664	11		X	X

Tabelle 9.1: Details der genutzten Arbeitslastaufzeichnungen mit den Konfigurationen der untersuchten drei Grid-Szenarien.

Die im vorherigen Kapitel schon ausführlich analysierte LANL96-Site wird hier nun aus den dargestellten Gründen nicht mehr betrachtet, sodass ausschließlich Szenarien berücksichtigt sind, bei denen sich lokal hoch ausgelastete Teilnehmer zu einer Grid-Föderation zusammenfinden.

9.2.1 Austauschverhalten

Die Ergebnisse für Grid-Szenario 5 sind in Abbildung 9.2(a) beziehungsweise Tabelle 9.2 wiederum als Verbesserungen für AWRT und SA relativ zu der rein lokalen Ausführung dargestellt. Offensichtlich profitieren alle drei Nutzergemeinschaften deutlich von der verkürzten AWRT. Wie zu erwarten war, ist die Arbeitslast an den kleinen Sites leicht gesunken, während sie an der großen Site (CTC) leicht ansteigt.
Im Vergleich zu den Ergebnissen mit zentraler Austauschplattform, siehe Abbildung 8.2, fällt auf, dass die Verbesserung bei aktiver Nachfrage für die große Site (hier CTC) noch deutlich positiver ausgefallen ist. Durch dieses Verfahren wird also eher ein Vorteil für alle Sites erreicht, da durch die aktive Anfrage die Arbeit noch effizienter verteilt werden kann.



(a) Grid-Szenario 5 (b) Grid-Szenario 8

Abbildung 9.2: Darstellung der Verbesserung von AWRT und SA für die beiden untersuchten Grid-Szenarien mit jeweils drei teilnehmenden Partnern im Vergleich zu rein lokaler Ausführung und EASY-Backfilling als LRMS-Schedulingalgorithmus.

Ähnliche Ergebnisse zeigen Abbildung 9.2(b) und Tabelle 9.2 für Szenario 8, wo verkürzte AWRT-Werte für alle Teilnehmer festzustellen sind. An der Auslastung ist abzulesen, dass auch hier die großen Sites Arbeit von der kleinen Site übernehmen, dies jedoch ohne Einbußen mit Hinblick auf die AWRT geschieht. Im Gegenteil können die AWRT-Werte an den großen Sites trotz Mehrbelastung um mehr als 10 % verkürzt werden. Die Ergebnisse sind dabei sehr ähnlich zu denen in Abbildung 8.5(b) unter Einsatz der zentralen Plattform. Aus der Tabelle 9.2 ist außerdem ersichtlich, dass die Gesamtauslastung des Grid nahezu unverändert bleibt.

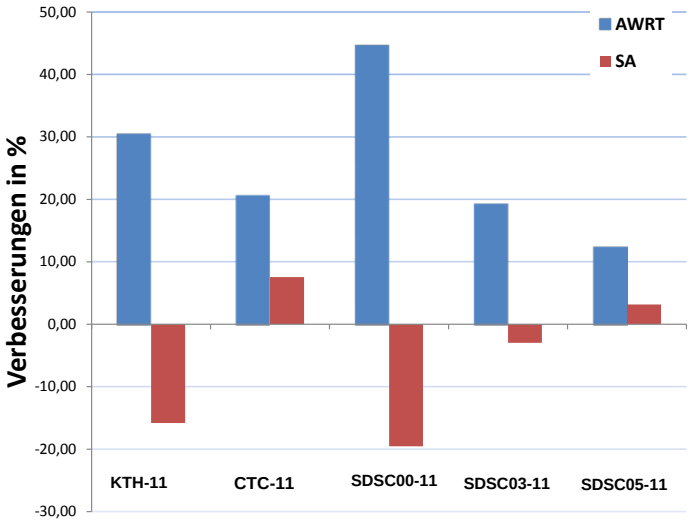


Abbildung 9.3: Darstellung der Verbesserung von AWRT und SA für das Grid-Szenario 9 mit fünf teilnehmenden Partnern im Vergleich zu rein lokaler Ausführung und EASY-Backfilling als LRMS-Schedulingalgorithmus.

Das oben beobachtete Verhalten spiegelt sich nun sehr ähnlich auch bei dem größeren Grid-Szenario 9 mit insgesamt fünf Sites wider, siehe Abbildung 9.3. An den kleinen Sites nimmt die Arbeitslast dabei wieder um bis zu 20 % ab, während sie an den großen Sites um bis zu 10 % zunimmt. Deutliche Verkürzungen der AWRT werden jedoch bei allen Sites um mindestens 10 % erreicht. Auffällig ist allerdings, dass die Arbeitslast auch bei der großen SDSC03-Site leicht abnimmt, während sie bei der relativ kleinen CTC-Site leicht zunimmt. Dies zeigt, dass durchaus nicht immer die kleinen Sites auf Kosten der großen ihre Arbeitslast verschieben, sondern dass ein Ausgleich durchaus unter allen Teilnehmern hergestellt wird. Genauer stellt sich dieser Zusammenhang bei der Betrachtung des Migrationsverhaltens dar.

9.2.2 Migrationsverhalten

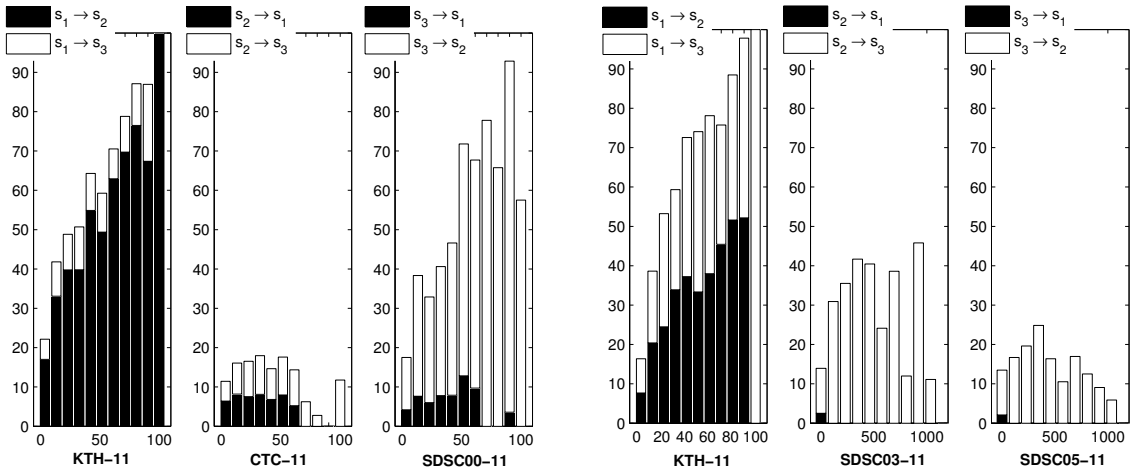
In Abbildung 9.4 und Tabelle 9.2 ist nun das Migrationsverhalten genauer aufgeschlüsselt. Dazu sind die beiden Migrationsmatrizen M_n und M_{SA} dargestellt, an denen deutlich ersichtlich ist, dass der Anteil der schließlich lokal ausgeführten Jobs den der tatsächlich migrierten Jobs deutlich übersteigt. Dies wird besonders bei Betrachtung der Hauptdiagonale der Matrix M_n für beide Szenarien deutlich, wo mindestens 70 % der Jobs als lokal ausgeführt markiert werden.

	Grid-Szenario 5			**Grid-Szenario 8**		
Daten	KTH-11	CTC-11	SDSC00-11	KTH-11	SDSC03-11	SDSC05-11
k	1	2	3	1	2	3
m_k	100	430	128	100	1.152	1.664
AWRT_k	56.098,42	50.453,40	45.128,33	53.168,72	42.486,89	49.091,71
U_k	62,18	69,73	66,55	59,30	67,50	61,67
$\Delta\ \text{AWRT}_k$	25,36	4,69	38,69	29,26	16,32	10,67
$\Delta\ \text{U}_k$	-9,51	6,13	-11,22	-13,70	-1,80	2,49
$\Delta\ \text{SA}_k$	-9,51	6,13	-11,35	-13,30	-1,86	2,39
$\Delta\ \text{U}_o$	0,0372	—	—	-0,092	—	—
$M_n =$	72,72	21,38	5,90	77,14	11,26	11,59
	6,62	87,53	5,85	2,20	83,71	14,09
	5,13	19,29	75,59	1,87	12,01	86,13
$M_{SA} =$	52,16	38,18	9,66	53,95	23,03	23,02
	6,83	84,92	8,25	1,11	69,71	29,18
	7,48	35,46	57,06	1,37	20,99	77,64

Tabelle 9.2: Detaillierte Ergebnisse für die Grid-Szenarien 5 und 8 mit jeweils drei Sites und der aktiven Anforderung von Jobs. Die Ergebnisse sind im Vergleich zum lokalen EASY-Backfilling dargestellt. Zusätzlich ist das Migrationsverhalten in Form der Matrizen M_n und M_{SA}, siehe Kapitel 7.4, angegeben. Die AWRT_k ist jeweils in Sekunden und alle übrigen Bewertungsfunktionen sind in % angegeben.

Im Hinblick auf die migrierte Arbeitslast, dargestellt in M_{SA}, ist eine deutliche Korrelation zwischen der migrierten Arbeit und dem Größenunterschied der zwei Sites festzustellen. Mit steigendem Größenunterschied tendiert der größere Partner dazu, mehr Arbeit von dem kleineren Partner an sich zu ziehen. Dieses Verhalten ist besonders in Szenario 5 nachgewiesen, wo die CTC-Site ungefähr 35 % der Arbeit sowohl vom KTH als auch vom SDSC00 übernimmt. Gleichzeitig tauschen jedoch ungefähr gleich große Sites auch ungefähr die gleiche Arbeitslast aus; dies ist zum Beispiel in Szenario 8 zwischen den großen Sites (SDSC03 und SDSC05) zu beobachten.

Weiterhin ist in Abbildung 9.4 wieder die Aufteilung der ausgetauschten Jobs unterteilt nach Parallelitätsgrad (m_j) und normalisiert auf das jeweilige Vorkommen in der ursprünglichen Arbeitslastaufzeichnung dargestellt, siehe auch Kapitel 8.2.4.



(a) Grid-Szenario 5 (b) Grid-Szenario 8

Abbildung 9.4: Relativer Anteil der *migrierten* Jobs für die beiden Grid-Szenarien 5 und 8. Die Werte sind unterteilt nach dem Parallelitätsgrad (m_j) und beziehen sich jeweils auf das Vorkommen im Originaldatensatz als Anteile in %.

Dabei zeigt sich nicht nur deutlich, dass Jobs von kleinen Sites zu großen Sites verschoben werden, sondern auch, dass ein reger Austausch zwischen den *großen* Partnern stattfindet. Wiederum resultieren die Verbesserungen an der KTH-Site daraus, dass dort *alle* hochparallelen Jobs migriert werden können und die SDSC00-Site einen hohen Anteil ihrer großen Jobs verschiebt. Ebenfalls ist in Abbildung 9.4(b) zu sehen, dass SDSC03 fast die Hälfte seiner sehr großen Jobs auch migriert und dadurch AWRT-Vorteile erreicht.
Allgemein ist festzustellen, dass auch mit diesem aktiven und vollkommen dezentralen Verfahren die Ausnutzung der Ressourcen durch die Grid-Teilnahme deutlich verbessert wird und dadurch für alle Partner verbesserte AWRT-Werte erreicht werden. Durch den Einsatz eines Backfilling-Algorithmus auf lokaler Ebene wird mittels einer aktiven Anfragemöglichkeit offensichtlich sehr viel öfter ein Job gefunden, der die sich momentan ergebenden lokalen Lücken im Schedule auffüllen kann.
Abschließend sei hier noch die bereits angesprochene Problematik der potenziellen Jobzirkulation untersucht. Generell ist durch den Einsatz von EASY zwar davon auszugehen, dass akzeptierte Jobs auch direkt zur Ausführung gebracht werden, jedoch gibt das vorgestellte Verfahren keine Garantie dafür, da die Jobs regulär in die Warteschlange eingefügt werden. Mit den Ergebnissen in Tabelle 9.3 ist jedoch nachgewiesen, dass die ständige Migration von Jobs zwischen Warteschlangen nur sehr selten auftritt. Auch wenn einige Jobs sehr häufig vor der endgültigen Ausführung migriert werden (siehe rechte Spalte), überwiegt doch die unmittelbare Ausführung, was durch einen Mittelwert nahe 1 ausgedrückt ist.
Auch wenn einige Jobs bis zu 100-mal zwischen den Partnern wechseln, bevor sie schließlich an einer Site gestartet werden, werden Jobs doch im Durchschnitt nur einmal migriert. Dies führt zu der Annahme, dass in der Warteschlange keine große Fluktuation in Bezug auf die momentan wartenden Jobs herrscht. Wenn also ein Job zur Migration akzeptiert wurde, hat

Datensatz	Mittelwert	Standardabweichung	Max. Anzahl von Migrationen
Grid-Szenario 5			
KTH-11	0,78	2,45	59
CTC-11	0,82	3,27	73
SDSC00-11	0,71	2,47	80
Grid-Szenario 8			
KTH-11	0,52	2,13	85
SDSC03-11	0,66	2,75	96
SDSC05-11	0,97	3,93	112
Grid-Szenario 9			
KTH-11	0,51	2,07	88
CTC-11	0,57	2,48	75
SDSC00-11	0,48	2,00	97
SDSC03-11	0,83	2,94	68
SDSC05-11	1,25	4,58	112

Tabelle 9.3: Statistische Eigenschaften des beobachteten Austauschverhaltens eingereichter Jobs. Analysiert wurde die Anzahl der Migrationen von der Einreichung bis zur letztendlichen Ausführung der Jobs. Zusätzlich ist die maximale Anzahl von Migrationen bis zur endgültigen Ausführung für mindestens einen Job des jeweiligen Workload angegeben.

sich die Situation in der Warteschlange bis zum nächsten Aufruf des lokalen Schedulingalgorithmus offensichtlich nicht signifikant geändert. Deshalb können auch am Ende eingefügte Jobs durch das Backfilling-Verfahren unmittelbar zur Ausführung gebracht werden.
Zusammen mit den beobachteten Verbesserungen der AWRT ist in Bezug auf die hier analysierten Szenarien auf Basis realer Arbeitslastaufzeichnungen davon auszugehen, dass die Migration von Jobs mit dem hier vorgestellten aktiven Anfrageverfahren keine künstliche Verzögerung der Jobausführung zur Folge hat.
Bevor nun effizientere Strategien für den Einsatz in vollständig dezentralen Umgebungen mit starken Informationsrestriktionen entwickelt und evaluiert werden, soll zunächst das Verhalten eines Grid mit unterschiedlichen Partnern beleuchtet und dabei das mögliche Verbesserungspotenzial mittels mehrkriterieller evolutionärer Optimierung identifiziert werden.

10

Bestimmung des Optimierungspotenzials mit mehrkriterieller evolutionärer Optimierung

STATISCHE KONZEPTE für das Scheduling in dezentralen Grid-Architekturen wurden in den beiden vorherigen Kapiteln entworfen und analysiert. Es ist dabei bereits darauf hingewiesen worden, dass einige Grundannahmen, wie die Beachtung einer restriktiven Informationspolitik sowie der vollständige Verzicht auf zentrale Komponenten, noch nicht umgesetzt, beziehungsweise zunächst bewusst vernachlässigt wurden. Ziel des gesamten Teils III dieser Arbeit ist es, zunächst die Dynamik und das Potenzial von dezentralen Schedulingsystemen allgemein zu analysieren und zu quantifizieren, wobei bisher zunächst die Beobachtung von beispielhaft eingesetzten Heuristiken durchgeführt wurde. Die betrachteten Ansätze führen (unter aufgeweichten Randbedingungen) dabei für fast alle Szenarien zu deutlichen Verbesserungen der Antwortzeiten und einer Verschiebung der Lastsituation. Die erreichten Ergebnisse wurden jeweils im Vergleich zu der rein lokalen Ausführung mit EASY-Backfilling betrachtet und die entsprechenden Verbesserungen wurden identifiziert.

Die Teilnahme am Grid motiviert sich primär durch eine potenzielle Verkürzung der Antwortzeiten für die einzelnen lokalen Nutzergemeinschaften. Je größer die Verbesserung der AWRT gegenüber lokaler Ausführung ist, desto größer ist die Motivation, Grids in der hier vorgestellten Weise zu betreiben. Ein effizientes Schedulingsystem muss also für alle teilnehmenden Partner in der Lage sein, die maximale Verbesserung gegenüber rein lokaler Ausführung zu erreichen. Es stellt sich also die Frage, wie groß die maximale Verbesserung für Grid-Teilnehmer in einem dezentralen Grid sein kann, beziehungsweise welche Auswirkungen die wechselseitige Übernahme von Jobs in einem derart gekoppelten System auf die Antwortzeiten hat.

Die vorherigen Ergebnisse haben gezeigt, dass eine geschickte Aufteilung der Arbeitslast unter beiden Sites zu Verbesserungen der AWRT führen kann, obwohl sich die Lastsituation einseitig zulasten eines Partners ändert; die unterschiedliche Beschaffenheit der Jobeinreichungen führt in der Durchmischung trotzdem zu verbesserten Schedules. Verschiebt aber eine Site einen großen Anteil ihrer Jobs zu einer anderen Site, so führt dies auf dort zunächst zu einer Mehrbelastung, was in deutlich verschlechterten Antwortzeiten für die überbelastete Site resultieren kann.

Hier besteht also die Gefahr eines Ungleichgewichts, was auf Dauer von dem überlasteten Teilnehmer nicht akzeptiert werden kann. Nimmt die Überlastung noch weiter zu, so ergibt sich die Situation, dass alle verschobenen Jobs nicht mehr schneller als ohne Verschiebung berechnet werden können. Es wäre also in diesem Falle sogar günstiger für die entlastete Site, Jobs nicht zu verschieben, weil die Lastsituation an der entfernten Site sich als ungüns-

tig darstellt. Dies führt dann in der Konsequenz zu einer Verschlechterung und zu einer Instabilität des gesamten Grid, die es zu vermeiden gilt. Weiterhin spielen in diesem Zusammenhang natürlich nicht nur die Charakteristiken der Einreichungen eine große Rolle, sondern auch die unterschiedliche Größe der Sites. Gerade im Hinblick auf einen angestrebten Lastausgleich ist es notwendig, die Ressourcenkapazitäten mit zu berücksichtigen, um die Auswirkungen auf die AWRT und ein mögliches Verbesserungspotenzial zu identifizieren.
Die gegenseitige Abhängigkeit der Antwortzeiten von der jeweiligen Lastzuordnung führt dazu, dass die AWRT der beiden Sites nicht mehr unabhängig betrachtet werden können und sich diese vielmehr als konkurrierende Zielfunktionen darstellen. Die Frage nach der im Grid kürzesten überhaupt erreichbaren AWRT unter gegebenen Jobeinreichungen und Maschinengrößen kann also nicht eindeutig beantwortet werden, sondern führt zu einem mehrkriteriellen Optimierungsproblem, bei dem die Lösung als Pareto-Front, also die Menge aller nicht dominierten Lösungen, der beiden Zielfunktionen anzugeben ist. Das Austauschverfahren oder das Schedulingkonzept, durch das eine derartige Aufteilung erreicht werden kann, ist bisher vollkommen unbekannt und auch nicht relevant. Es soll vielmehr versucht werden, die Arbeitslast *a priori* so aufzuteilen, dass sie bei rein lokaler Abarbeitung zu Pareto-optimalen AWRT-Lösungen führt. Die Lösung des mehrkriteriellen Optimierungsproblems, also die Aufteilung der gesamten Arbeitslast auf die einzelnen Grid-Sites, wird hier *offline* durchgeführt, während die eigentliche Auswertung der vorgenommenen Arbeitslastaufteilung dann *online* simuliert wird.
Die Lösung eines derartigen Problems ist extrem aufwendig, da bei Berücksichtigung der realen Aufzeichnungen Partitionen für annähernd 50.000–150.000 Jobs erstellt werden müssen. In diesem Kapitel wird deshalb nun zum ersten Mal eine Methode der Computational Intelligence in Form des mehrkriteriellen evolutionären Algorithmus NSGA-II, siehe Deb u. a. [50], genutzt, um die Pareto-Fronten zu approximieren. Es ist klar, dass die Nutzung von NSGA-II nur eine Approximation der Fronten liefern kann und es keinen Nachweis für ihre Optimalität gibt. Die Interpretation der generierten Fronten ermöglicht dabei nicht nur tiefere Einblicke in die Struktur eines Grid-Verbunds, sondern verdeutlicht auch im Hinblick auf gegebene Zielfunktionen die Grenzen des durch dezentrales Grid-Scheduling Erreichbaren. Da hier die Aufteilung offline vor der Durchführung des eigentlichen Online-Schedulings getroffen wird, dies aber bei einem realen System vollständig online geschehen muss, gelten die hier approximierten Ergebnisse, siehe auch Grimme u. a. [17], als untere Schranke.

10.1 Methodik und Algorithmus

Das Vorgehen zur Approximation der Pareto-Front gliedert sich in zwei wesentliche Schritte:

1. Erzeugung einer beliebigen Aufteilung der Arbeitslast. Dazu sind dem System (im Gegensatz zum realen System) bereits zu Beginn alle eingereichten Jobs bekannt (Offline-Entscheidung). Diese Aufteilung ist einmal fest vorgegeben und wird für die Bewertung während Schritt 2 nicht weiter modifiziert.

2. Die Aufteilung aus Schritt 1 wird nun lokal abgearbeitet und im Sinne eines realen Schedulingsystems simuliert (Online-Entscheidungen). Dabei findet kein Jobaustausch zwischen den Partnern statt.

Die Idee ist also, den Austausch als durch ein beliebiges Verfahren realisiert anzusehen und lokal nur die Konsequenzen als veränderte Jobeinreichungen im Online-Verfahren zu simulieren. Diese Zuordnung wird nun von dem mehrkriteriellen evolutionären Algorithmus übernommen, der die Pareto-optimalen Lösungen iterativ ermittelt.

10.1.1 Kodierung

Ein Individuum $\mathbb{I} = [a_1, a_2, \ldots, a_{l(\mathbb{I})}]$ repräsentiert die Aufteilung der Originaldatensätze $k = 1 \ldots K$ mit jeweils n_k Jobs in K Partitionen. Dabei bezeichnet K wiederum die Anzahl der teilnehmenden Grid-Sites und $l(\mathbb{I}) = \sum_{k=1}^{K} n_k$ die resultierende Länge, also die Anzahl der Parameter des Individuums $\mathbb{I}$.

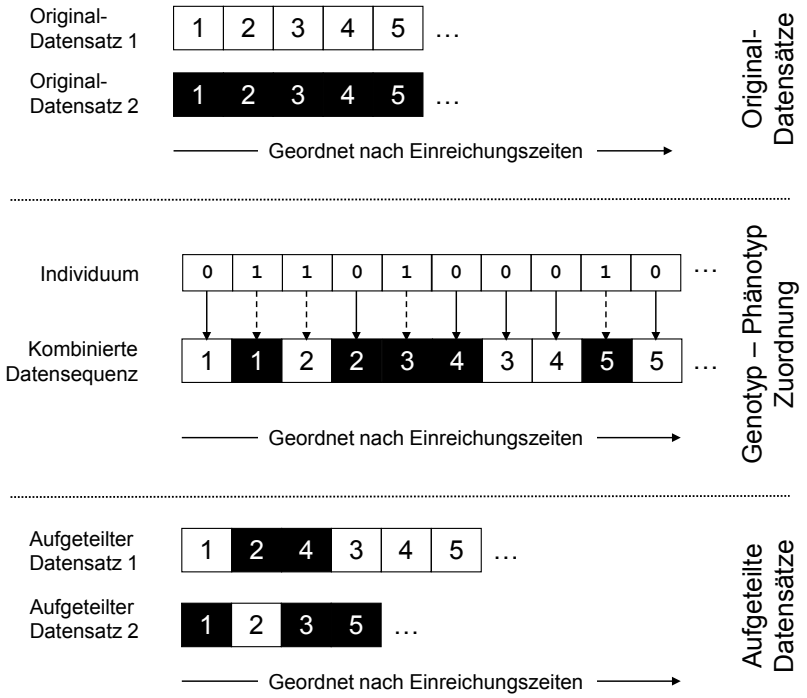


Abbildung 10.1: Kodierungskonzept für Individuen und die resultierende Aufteilung der Arbeitslastaufzeichnungen.

Durch die Verschmelzung der ursprünglichen Arbeitslastaufzeichnungen wird zunächst *eine* Jobsequenz erzeugt, in der alle insgesamt ins System eingereichten Jobs nach deren Einreichungszeiten r_j sortiert sind. Diese Sequenz wird als Referenzdatensatz betrachtet, auf den nun das Individuum die jeweiligen Partitionen auf die Grid-Sites abbildet. Dazu muss in einem Individuum jedem Job eine Site zugeordnet werden, weshalb jedes Individuum genau $l(\mathbb{I})$ Gene enthält.
Jede Grid-Site wird nun wiederum durch eine natürliche Zahl aus dem Intervall der möglichen Allele $a \in [0, K-1]$ repräsentiert. Wie später noch genauer erläutert werden für die hier durchgeführten Untersuchungen lediglich zwei Sites ($K = 2$) angenommen, wodurch sich eine binäre Repräsentation ergibt. Die Zuordnung zwischen Geno- und Phänotyp ergibt somit dann die erwünschte Aufteilung, wobei die Ordnung nach Einreichungszeiten

wiederum bestehen bleibt; es ergeben sich also gültige und damit auswertbare Jobsequenzen. Nachteil dieser Kodierung sind die große Länge der resultierenden Individuen und der sich daraus ergebende sehr große Suchraum.

Das gesamte Vorgehen ist in Abbildung 10.1 für zwei Sites dargestellt. Es bleibt noch der häufig vorkommende Fall zu berücksichtigen, dass ein Job j aufgrund seiner Ressourcenanforderungen nur auf der ursprünglichen Site ausgeführt und nicht migriert werden kann $(m_j > m_k)$. In diesem Fall wird der Job auf seiner ursprünglichen Site ausgeführt und die im Individuum vorgesehene Aufteilung für diesen Job ignoriert.

Die speziellen Eigenschaften des Problems und die daraus resultierende Kodierung erfordern auch die Konzeption und Anpassung spezieller Mutations- und Rekombinationsoperatoren, die im Folgenden genauer vorgestellt werden.

10.1.2 Mutationsoperatoren

Üblicherweise wird für die Variation im NSGA-II-Algorithmus vorwiegend Rekombination verwendet, während die Mutation eine eher untergeordnete Rolle spielt. Untersuchungen von Rothlauf [145] zeigen aber, dass gerade in kombinatorischen Problemen sowie bei der Anwendung von binären Repräsentationen die Erzeugung von möglichst diversen Lösungskandidaten durch Mutation einfacher zu erreichen ist. Deshalb wird auch hier zunächst eine *Zufallsmutation* verwendet, bei der jedes Gen $a_i \in \mathbb{I}$ mit der Wahrscheinlichkeit P_{rand} zu

$$a_i' = [a_i + \lfloor (K+1) \cdot \mathcal{U}(0,1) \rfloor] \bmod K \tag{10.1}$$

verändert wird. Dabei bezeichnet in Gleichung 10.1 der Ausdruck $\mathcal{U}(0,1)$ eine gleichverteilte Zahl zwischen 0 und 1. Diese Mutation fokussiert sich ausschließlich auf den Austausch zwischen unterschiedlichen Partitionen und kann zu besseren Zuordnungen von Jobs zwischen den Partitionen führen, wodurch dann insgesamt eine bessere Lösungsqualität erreicht wird. Es ist dabei allerdings zu befürchten, dass durch den reinen Austausch zwischen den Partitionen keine diverse Front erzeugt werden kann, da die Auswirkungen von einzelnen Jobs nicht ausreichen werden. Dazu ist es vielmehr notwendig, wirkliche Veränderungen in der Lastsituation zu initiieren, durch die dann auch extremere Lösungen erzeugt werden. Deshalb wird hier noch zusätzlich eine Mutation eingeführt, durch die eine einseitige Verschiebung von Jobs zwischen den Partitionen erreicht wird.

Diese sogenannte Verschiebungsmutation ist in Algorithmus 3 detailliert dargestellt. Zunächst wird normalverteilt die Anzahl von Jobs n_s bestimmt, die einseitig verschoben werden sollen (Zeile 2). Dann wird zufällig eine Site k_s bestimmt, zu der im Folgenden alle n_s ausgewählten Jobs verschoben werden (Zeile 4). Es werden nun in Zeile 3 so lange Verschiebungsoperationen ausgeführt, bis genau n_s Verschiebungen ausgeführt wurden. Dazu werden gleichverteilt Positionen im Individuum bestimmt (Zeile 7). Wenn der Job nun nicht schon der Ziel-Site k_s zugeordnet ist und somit eine Verschiebung im Rahmen dieser Mutation durchgeführt werden kann (Zeile 8) wird das entsprechende Gen verändert und somit der Ziel-Site zugewiesen (Zeile 9). Es zählen dabei nur die Fälle, in denen eine Verschiebung wirklich durchgeführt werden kann, und nur dann wird n_s entsprechend erhöht.

Algorithmus 3: Verschiebungsmutation

Eingabe: Individuum $\mathbb{I}$, Verschiebungsweite σ, Größe des Grid K

Anzahl der zu verschiebenden Jobs ermittelt aus Normalverteilung mit Standardabweichung σ.
$n_s = \lfloor |\mathcal{N}(0,\sigma)| + 0.5 \rfloor$;
Bestimme zufällig Site k_s, zu der alle n_s Jobs verschoben werden.
$k_s = \lfloor (K+1)\mathcal{U}(0,1) \rfloor$;
Garantiert eine Anzahl von genau n_s verschobenen Jobs.
while $n_s > 0$ **do**
    p := gleichverteilte Zufallsposition innerhalb des Individuums $\mathbb{I}$;
    if $\mathbb{I}[p] \neq k_s$ **then**
        $\mathbb{I}[p] = k_s$;
        $n_s = n_s - 1$;
    end
end

Dieser Algorithmus benötigt also drei Eingabeparameter:

1. Das zu mutierende Individuum $\mathbb{I}$,
2. die Schrittweite σ, durch die die Standardabweichung der Normalverteilung bestimmt ist und somit die Stärke der Verschiebung über n_s kontrolliert wird und
3. die Anzahl K der teilnehmenden Grid-Sites.

Um beide Mutationsvarianten in NSGA-II zum Einsatz zu bringen und gleichzeitig die Gewichtung der Mutationen kontrollieren zu können, wird zwischen beiden Operatoren mit der Wahrscheinlichkeit P_{shift} gewechselt. In jeder Generation wird dabei jedes Individuum mit der Wahrscheinlichkeit P_{shift} durch die Verschiebungsmutation und mit der Wahrscheinlichkeit $(1 - P_{shift})$ durch die Zufallsmutation verändert. Dabei wird für letztere jedes Gen wie beschrieben mit der Wahrscheinlichkeit P_{rand} mutiert.

10.1.3 Rekombinationsoperator

Als vorwiegend genutzte Variationsart kommt beim NSGA-II Rekombination zur Anwendung, wie zum Beispiel das zu diesem Zweck extra von Deb und Agrawal [49] entwickelte Simulated Binary Crossover (SBX). Dieser Operator ist in diesem Kontext aber nicht anwendbar, da er nur für reellwertige Suchräume konzipiert ist und es sich bei dem vorliegenden Problem um ein rein ganzzahliges handelt. Weiterhin ist die Nutzung dieses speziellen Operators in diesem Kontext auch inhaltlich nicht gerechtfertigt, da dessen Einfluss bei der Anwendung auf Schedulingprobleme noch nicht ausreichend untersucht ist. Außerdem wird bei der hier vorgestellten Lösung des Problems primär auf Mutation gesetzt, weshalb nun für die Rekombination der Einsatz eines Standardverfahrens, wie unter anderem von Michalewicz [123] zusammengefasst, ratsam erscheint.

Dabei ist das häufig verwendete *Uniform Crossover* (UCX), bei dem jedes Bit zweier Eltern mit gleicher Wahrscheinlichkeit verändert wird, nicht geeignet, da dies in einem reinen Austausch von Jobs resultiert und keine Veränderung der Partitionsgrößen erzeugt. Wie schon erwähnt, ist dies aber essenziell notwendig, um eine Lastverlagerung zu erreichen und eine divers approximierte Pareto-Front zu erzeugen. Ein Wechsel der Zuordnung zwischen zwei Eltern resultiert im Durchschnitt aber in der gleichen Anzahl wie der zu einer Partition zugeordneten Jobs.

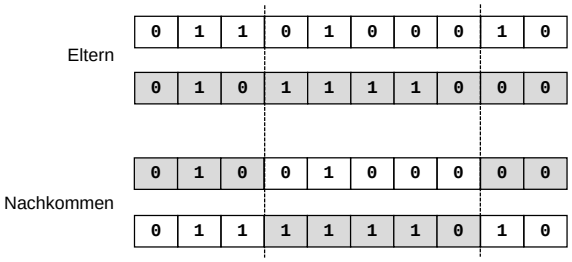


Abbildung 10.2: Zwei-Punkt-Crossover-(TPX-)Rekombination bei binärer Repräsentation.

Die Anwendung des sogenannten n-Punkt-Crossover hingegen ist dazu geeignet, beide erwünschten Veränderungseigenschaften zu kombinieren und dadurch sowohl gleichzeitig einen Austausch zwischen Partitionen als auch veränderte Zuordnungen zu erzeugen. Dabei lassen sich aber Jobsequenzstrukturen von den Eltern auf die Nachkommen im Sinne einer Vererbung übertragen. In dieser Arbeit wird nun ein einfaches Zwei-Punkt-Crossover (TPX) verwendet, bei dem zufällig zwei Crossoverpositionen ermittelt und die Gene entsprechend Abbildung 10.2 für die Erstellung von Nachkommen rekombiniert werden. Die Rekombination wird dabei mit der Wahrscheinlichkeit P_{recomb} durchgeführt.

10.1.4 Zielfunktion

Das mehrkriterielle Optimierungsproblem (MOP), das sich durch die veränderte Lastsituation an den verschiedenen Sites ergibt, lässt sich mittels der schon in Abschnitt 7.4 eingeführten AWRT_k der auf Site k eingereichten Jobs nach Gleichung 10.2 angeben.

$$\mathrm{MOP} := \min \begin{pmatrix} \mathrm{AWRT}_1 \\ \vdots \\ \mathrm{AWRT}_K \end{pmatrix} \tag{10.2}$$

Es handelt sich hier also um ein $l(\mathbb{I})$-dimensionales MOP mit K Zielfunktionen. Dabei kommt zum Ausdruck, dass alle Sites in einem Grid prinzipiell zwar kooperieren wollen, aber jede Site für ihre Nutzergemeinschaften die größte Verbesserung ihrer AWRT gegenüber rein lokaler Auslastung erzielen will. Dies geht nicht ohne gegenseitige Konkurrenz, da eine Verbesserung des einen Partners eine Verschlechterung des anderen zur Folge haben wird. Wie genau aber dieser Stabilitäts- und Lastzusammenhang in einem Grid geartet ist, wird durch die Lösung dieses MOP zum ersten Mal sichtbar und damit auch beschreibbar.

10.1.5 Einschränkung des Suchraumes

Die einfache Kodierung der Partitionen hat den Nachteil, dass die Individuen sehr lang sind und dadurch der Suchraum außerordentlich groß ist, was wiederum eine Beeinträchtigung

des mehrkriteriellen evolutionären Algorithmus darstellt. Weiterhin ist für die Auswertung eines einzelnen Individuums die Simulation eines gesamten Grid notwendig, um die Funktionswerte für alle einzelnen Kriterien zu bestimmen, was auch bei Nutzung des Teikoku Grid-Scheduling Frameworks, siehe Kapitel 7.3, einen beträchtlichen Zeitaufwand bedeutet. Deshalb wird zum einen die Anzahl der teilnehmenden Partner auf $K = 2$ begrenzt (siehe auch Abschnitt 10.2), da dies ausreicht, um die wesentlichen Eigenschaften zu untersuchen. Zum anderen wird der zu untersuchende Suchraum auf den wirklich relevanten Bereich eingeschränkt.
Wenn sehr viele Jobs zu einer Site migriert werden, ist zu erwarten, dass sich dort die AWRT über alle Maßen hinweg verschlechtert, da selbst das beste Schedulingsystem nicht in der Lage ist, diese Überlast durch gute Allokationen zu kompensieren. Derartige Resultate für die AWRT sind jedoch in keiner Weise akzeptabel und auch von keinem praktischen Interesse. Deshalb wird angenommen, dass Individuen, die für nur ein Kriterium AWRT-Werte erzeugen, die um mehr als 30 % schlechter als bei rein lokaler Ausführung sind (siehe Tabelle 7.2), für die weitere Verwendung ungeeignet. Hier wird deshalb der Suchraum auf Werte von

$$\mathrm{AWRT}_k \in [0 \ldots 100.000] \; \forall \, k = 1 \ldots K \tag{10.3}$$

beschränkt, sodass den Individuen mit größeren AWRT-Werten eine unendlich hohe Fitness gegeben wird, was wiederum effektiv ihren Ausschluss vom weiteren evolutionären Prozess bedeutet.

10.1.6 Erzeugung einer Startpopulation

Eine weitere Maßnahme, um die Effizienz der Suche in einem derart großen Suchraum zu steigern, besteht in der geschickten Initialisierung der Individuen. Da hier nur solche Partitionen von Interesse sind, bei denen ein besseres Resultat als bei rein lokaler Ausführung erreicht wird, ist EASY-Backfilling als gute initiale Lösung nutzbar. Es werden also zwei Partitionen erzeugt, die zunächst eine gleiche Aufteilung wie bei rein lokaler Ausführung (also wie die ursprüngliche Zuordnung) umsetzen. Dann werden zufällig 5.000 Jobs ausgetauscht, wodurch eine gewisse Diversität in der Startpopulation erzeugt wird, die aber dann immer noch schwach gestreut um den EASY-Referenzpunkt liegen. Von dort aus ist es für den NSGA-II-Algorithmus deutlich einfacher, in Richtung der Pareto-Front zu konvergieren, als wenn dieser an beliebigen Stellen im Suchraum initialisiert wird.

10.2 Evaluation

Nachdem nun das genaue Prinzip der hier verfolgten mehrkriteriellen Optimierung mittels NSGA-II beschrieben ist, werden die erzeugten Ergebnisse dargestellt und besprochen. Zuvor wird allerdings noch der experimentelle Aufbau in allen Details vorgestellt.

10.2.1 Simulationsaufbau

Die verwendeten Daten basieren erneut auf den realen Arbeitslastaufzeichnungen und sind der Vollständigkeit halber entsprechend der ausgewählten vier Szenarien in Tabelle 10.1 aufgeführt. Es wurde bei der Auswahl der Szenarien Wert darauf gelegt, dass möglichst alle unterschiedlichen Sitegrößen kombiniert werden: Setup I repräsentiert deshalb wiederum

Szenario	Daten	m	n	Gesamte Jobanzahl	AWRT (EASY)	AWRT (PLATF)	AWRT (AA)
I	KTH-11	100	28.479	58.289	75.157,63	63.011,46	61.448,95
	SDSC00-11	128	29.810		73.605,86	58.772,48	57.918,61
II	KTH-11	100	28.479	105.678	75.157,63	58.056,65	55.462,28
	CTC-11	430	77.199		52.937,96	51.742,31	51.347,65
III	KTH-11	100	28.479	94.063	75.157,63	58.497,60	55.527,87
	SDSC03-11	1.152	65.584		50.772,48	50.809,59	50.601,24
IV	SDSC03-11	1.152	65.584	140.487	50.772,48	43.731,50	44.264,10
	SDSC05-11	1.664	74.903		54.953,84	48.628,57	46.498,92

Tabelle 10.1: Eigenschaften und Referenzergebnisse der untersuchen Grid-Szenarien. Die AWRT-Werte (in Sekunden) sind dabei für die rein lokale Ausführung (EASY), den Austausch mittels zentraler Plattform (PLATF) und die aktive Anforderung (AA) angegeben.

den Zusammenschluss zweier kleiner Institutsinstallationen, während Setup II die Kollaboration eines Instituts mit einem universitären Rechenzentrum wiedergibt. Weiterhin modelliert Setup III einen Zusammenschluss von kleinen Institutsinstallationen in wechselseitigem Austausch mit einem regionalen Rechenzentrum. Schließlich wird in Setup IV die überregionale Zusammenarbeit von Rechenzentren betrachtet. Neben dieser Motivation aus durchaus realen Szenarien sind aber vor allem der Einfluss unterschiedlicher Größen und der jeweils an die lokalen Beschaffenheiten angepasste Bedarf der Nutzergemeinschaften im gegenseitigen Austausch von Interesse. Die hohen Anforderungen an den multikriteriellen evolutionären Algorithmus werden nochmals bei der resultierenden Gesamtzahl der Jobs, siehe Tabelle 10.1, deutlich. Weiterhin sind die Ergebnisse angegeben, die durch die bereits im vorherigen Teil dieser Arbeit entwickelten Verfahren erreicht werden: Dazu sind die Ergebnisse bei rein lokaler Ausführung ohne Austausch (EASY), bei Austausch mittels zentraler Plattform, siehe Kapitel 8, bei Austausch durch aktiver Jobanforderung und eine „Accept When Fit (AWF)“-Strategie mit angegeben. Letztere wird zwar erst in Kapitel 11 beschrieben, aber ihr Verhalten wird der Vollständigkeit halber und aufgrund ihrer großen Bedeutung hier schon untersucht.

Besonders die aktive Anfrage kann für die Qualitätsbewertung der zu approximierenden Pareto-Front nützlich sein: Durch die Kombination von Nachfrage und der EASY-Backfilling-Strategie ist davon auszugehen, dass die somit erreichten Ergebnisse ein Pareto-Optimum darstellen. Wenn für beide lokalen Schedulingsysteme ständig alle Jobs durch Anfrage zur Verfügung stehen, ergibt sich eine holistische Sicht auf das Grid und alle darin verfügbaren Jobs. Zudem ist durch das Backfilling der Scheduler in der Lage, eine freie Wahl unter all diesen Jobs durchzuführen.

Für die Konfiguration des NSGA-II-Algorithmus wurde eine Populationsgröße von $\mu = 70$ Individuen verwendet und für jedes Szenario über 200 Generationen evaluiert. Als Rekombinationsoperator wird, wie in Abschnitt 10.1.3 erläutert, TPX mit einer Wahrscheinlichkeit von $P_{recomb} = 0,9$ verwendet. Weiterhin wird als Mutation, wie in Abschnitt 10.1.2 dargestellt, die Verschiebungs- und Zufallsmutation mit gleicher Wahrscheinlichkeit ausgeführt. Dabei wird für die Verschiebungsmutation eine Schrittweite von $\sigma = 2.000$ verwendet, während bei der Zufallsmutation jedes Gen mit einer Wahrscheinlichkeit von $P_{rand} = 0,1$ verändert wird. Insgesamt wird eine Form der Mutation in jeder Generation ausgeführt, was einer Mutationswahrscheinlichkeit von 1,0 entspricht.

Die Auswertungen wurden auf einem Cluster bestehend aus 120 Standard Pentium IV 2,4 GHz Prozessoren und Linux-Betriebssystem durchgeführt. Auf dieser Installation benötigt jede Generation ungefähr 15 Minuten für die Bewertung, wenn die ganze Population parallel ausgewertet wird. Dies führt zu einer Gesamtsimulationszeit von ungefähr zwei Tagen pro ausgewerteten Grid-Szenario. Alle Ergebnisse wurden dabei mittels einer angepassten Kombination von Sastrys [147] C++-Implementierung der GA-Toolbox für MATLAB und dem JAVA-basierten Teikoku Grid-Scheduling Framework, siehe Kapitel 7.3, erzeugt.

10.2.2 Ergebnisse

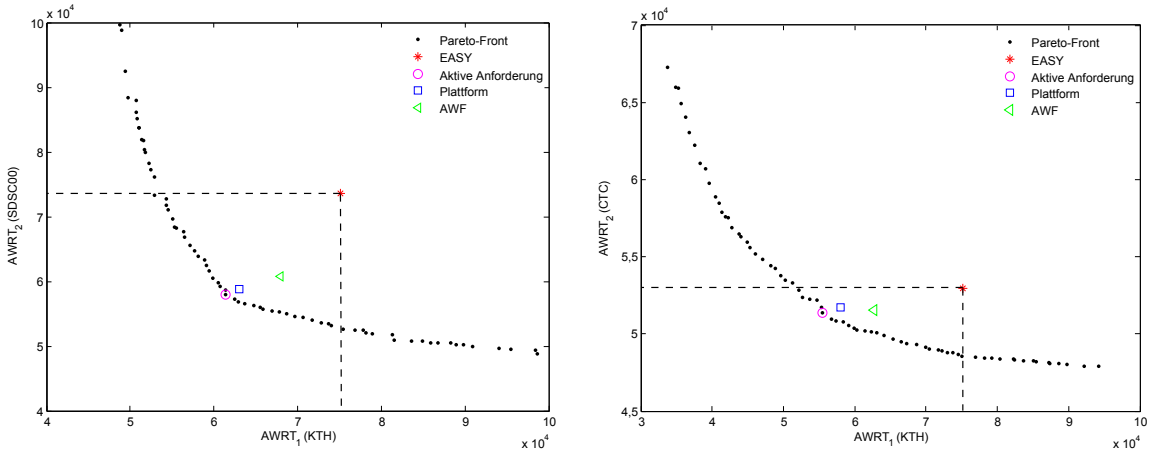
Die erhaltenen Ergebnisse sind, unterteilt nach dem jeweiligen Szenario, in vier Abbildungen dargestellt. Zusätzlich zu der mit NSGA-II approximierten Pareto-Front sind in jeder Abbildung die Ergebnisse bei Anwendung der Plattform-Strategie, der aktiven Anfragestrategie und des reinen EASY-Verfahrens für jeweils beide Zielfunktionen $AWRT_1$ und $AWRT_2$ eingezeichnet. Zusätzlich ist noch der sogenannte Verbesserungsbereich eingezeichnet, der in den Diagrammen oben links durch den EASY-Referenzpunkt und seitlich durch die jeweiligen Schnittlinien mit den Koordinatenachsen begrenzt ist. Alle Teile der approximierten Pareto-Front, die innerhalb dieses Bereichs liegen, stellen eine Verbesserung für beide Partner dar. Durch die Lage des EASY-Referenzpunktes in Bezug auf die eigentliche Front stellt sich heraus, wie stark und gleich aufgeteilt das Verbesserungspotenzial innerhalb eines Grid-Verbunds sein kann. Mit dieser Methodik sind zwei Grenzen für die Vorteile des Grid-Computing identifiziert:

1. Der nicht kooperative Fall, der hier durch die reine Anwendung von EASY repräsentiert ist, stellt eine obere Schranke dar. Alle Werte, die schlechter als dieser Punkt für auch nur ein Kriterium sind, sind nicht akzeptabel, da für mindestens einen Partner die Teilnahme an einem Grid nachteilig wäre.

2. Die Pareto-optimalen Lösungen innerhalb des Verbesserungsbereichs markieren die untere Schranke. Da diese offline erzeugt wurden, ist davon auszugehen, dass sie von online Algorithmen nur in besonderen Fällen überhaupt erreichbar sind.

Sinnvoll einsetzbare Grid-Schedulingverfahren müssen also Ergebnisse innerhalb dieses Verbesserungsbereichs erzeugen.[14]
In allen hier untersuchten Fällen wurden divers abgedeckte Pareto-Fronten innerhalb des spezifizierten Suchbereiches approximiert. Offensichtlich existiert jenseits der schon bekannten heuristischen Einzellösungen eine Vielzahl von Kompromissen, die bisher unbekannt waren. Wie schon oben angemerkt, kann dabei die Strategie mit aktiver Anfrage als ein Pareto-optimaler Punkt vermutet werden. In der Tat zeigen alle Abbildungen, dass dieser Punkt auch mit der Front erreicht wird. Deshalb ist auch insgesamt von einer hohen Approximationsqualität auszugehen. Es zeigt aber auch, dass NSGA-II selbst durch die hohe Dimensionalität des Problems nicht beeinträchtigt ist. Die beiden Abbildungen 10.3(a) und 10.4(b) zeigen die Ergebnisse für die beiden ähnlichen Szenarien I und IV, bei denen jeweils zwei ungefähr gleich große Sites kombiniert wurden. In beiden Fällen entsteht eine konvexe Pareto-Front über den gesamten Bereich des eingeschränkten Suchraumes. Die heuristischen Lösungen liegen dabei ziemlich genau in der Mitte der jeweiligen Front. In

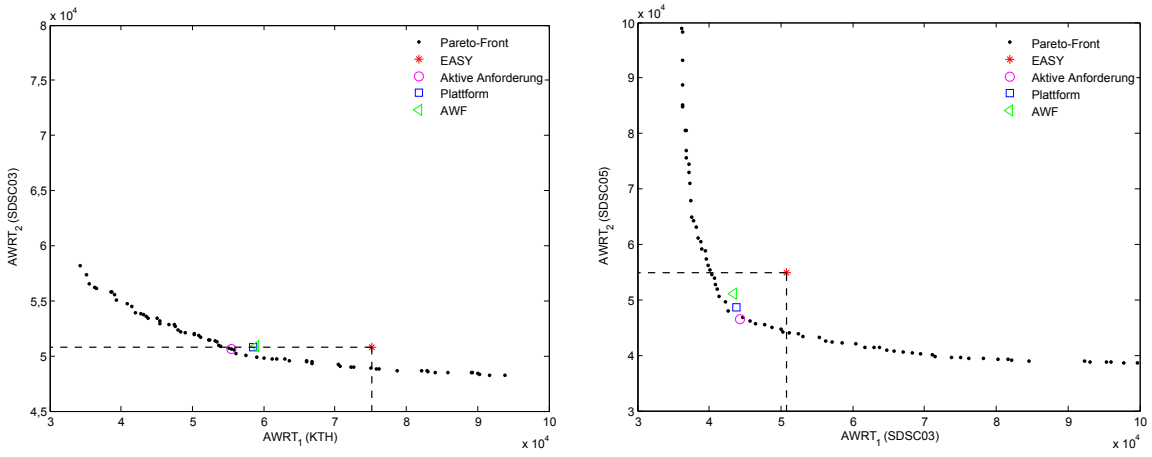
[14] Dabei kann natürlich der Referenzpunkt auch durch einen anderen LRMS-Algorithmus vorgegeben sein (wie zum Beispiel FCFS im späteren Teil dieser Arbeit).



(a) Grid-Szenario I (b) Grid-Szenario II

Abbildung 10.3: Darstellung der mit NSGA-II approximierten Pareto-Front nach 200 Generationen, des Verbesserungsbereichs und der Ergebnisse unterschiedlicher Austauschstrategien.

den Abbildungen 10.3(b) und 10.4(a) sind ähnliche Ergebnisse dargestellt. Hier zeigt sich jedoch, dass der EASY-Referenzpunkt verlagert ist, sodass auch der entsprechende Verbesserungsbereich einseitig zugunsten der kleinen Site verschoben ist. Das Potenzial für eine Verbesserung der AWRT ist, wie schon zu erwarten war, offenbar sehr viel kleiner für die große Site bei Interaktion mit einer kleinen Site. Es ist dabei zu beachten, dass die Achsen in diesen beiden Abbildungen für die größere Site ($AWRT_2$) in einem kleineren Auszug skaliert sind. Bei der Betrachtung von Ergebnissen der Strategie mit zentraler Plattform fällt



(a) Grid-Szenario III (b) Grid-Szenario IV

Abbildung 10.4: Darstellung der mit NSGA-II approximierten Pareto-Front nach 200 Generationen, des Verbesserungsbereichs und der Ergebnisse unterschiedlicher Austauschstrategien.

auf, dass hier offensichtlich ein perfekter Kompromiss zwischen beiden Partnern hergestellt

wird. Durch die nur temporäre Sicht auf alle Jobs kann natürlich die Front dabei nicht ganz erreicht werden. Allerdings ist die allgemeine Qualität der durch diese Heuristik erzeugten Schedules bereits sehr zufriedenstellend.
Die AWF-Strategie hingegen liefert nur deutlich schlechtere Ergebnisse. Ohne an dieser Stelle dafür eine genaue Begründung dafür zu liefern, sei zumindest so viel erwähnt, dass AWF unter starken Informationsbeschränkungen funktioniert und Entscheidungen ohne weitere Kenntnis der noch im Grid verfügbaren Jobs trifft. Es zeigt sich hier schon die Herausforderung, dass es bei weniger zur Verfügung stehenden Informationen ausgeklügelter Algorithmen bedarf, um effizientes Grid-Scheduling zu ermöglichen.
Durch die erzeugten Fronten konnte das Verhalten verschiedener Heuristiken innerhalb des möglichen Verbesserungsbereiches beispielhaft gezeigt werden. Dabei ist allgemein zu beobachten, dass die beiden vorgestellten Austauschstrategien stark dazu tendieren, die Last zwischen den Partnern auszugleichen. Dies resultiert dann in Ergebnissen, die im mittleren Bereich der Front liegen. Besonders in den Szenarios II und III, bei denen Partner mit stark unterschiedlicher Größe beteiligt sind, siehe Abbildungen 10.3(b) und 10.4(a), sind die Lösungen aber nur fair, wenn die gesamte Front betrachtet wird. In genau diesen Fällen aber ist der gegenseitige Verbesserungsbereich so verlagert, dass diese „fairen“ Lösungen nur für den kleineren Partner Verbesserungen bringen und der große Partner oftmals nur ohne Verschlechterung am Grid beteiligt ist. Mit anderen Worten sollen gute Grid-Schedulingverfahren nicht einen Lastausgleich mit globaler Sicht erzeugen, sondern auf das spezifische Verbesserungspotenzial der einzelnen Partner angepasst sein. Verfahren, die genau mit diesen Gedanken konzipiert sind und dazu auch aufwendigere Verfahren zur Optimierung einsetzen, werden im Teil IV vorgestellt.
Trotz der Wichtigkeit der hier erhaltenen Ergebnisse werden für künftige Evaluationen nur jeweils relative Vergleiche zur rein lokalen Ausführung herangezogen und es wird auf eine Auswertung in der Front verzichtet. Dies hat zum einen den Grund, dass es zu rechenaufwendig ist, für alle möglichen Szenarien und unterschiedlichen LRMS-Algorithmen die jeweiligen Fronten zu erzeugen. Zum anderen müssen auch Szenarien mit mehr als zwei Teilnehmern untersucht werden, für die dann die Approximationsqualität durch die beschriebenen Eigenschaften von NSGA-II bei den hier betrachteten Problemgrößen (es müssten annähernd 100.000 Jobs kodiert werden) nachlassen wird. Dies stellt sich als ein Hindernis bei der Bewertung von realen Problemgrößen heraus, da die Erzeugung von möglichst optimalen Offline-Lösungen nur mit entsprechendem Aufwand realisierbar ist. Diese würde zudem die Entwicklung ganz spezieller Variationsoperatoren erfordern, die besonders auf eine dann noch genauer zu identifizierende Problemstruktur angepasst werden müssen. Im Rahmen dieser Arbeit wurde auf die weitere Verfeinerung der mehrkriteriellen Algorithmen deshalb verzichtet. Für die Untersuchung des Lastverhaltens und die daraus gewonnenen Erkenntnisse in Bezug auf das mögliche Optimierunspotenzial von Grid-Verbänden reichen die durchgeführten Studien bereits aus. Auf Basis dieser Erkenntnisse werden im folgenden Teil nun die Schedulingverfahren mittels CI-Methoden optimiert.

Teil IV

Realisierung und Optimierung von dezentralen Grid-Schedulern mit CI-Methoden

11

Architektur eines dezentralen Grid-Schedulers

Bei allen bisher in dieser Arbeit vorgestellten Architekturen wird der Jobaustausch mittels einfacher, aber dennoch effektiver Strategien realisiert. Es gibt allerdings einige Kernprobleme bei der Umsetzung dieser Konzepte, die weitreichende Folgen für die Verbreitung von Computational Grids mit sich bringen.

Zunächst können, wie an den entsprechenden Stellen dargestellt, die guten Schedulingergebnisse mit derart einfachen Strategien nur dadurch erreicht werden, dass diese untrennbar ins LRMS integriert sind. Dies ist bei den bisher realisierten Verfahren notwendig, da Entscheidungen des LRMS auch auf die Interaktionen mit der Grid-Ebene abgestimmt sein müssen. Die Konfiguration des LRMS und die Site-spezifischen Anpassungen können deshalb in vielen Fällen so nicht mehr beibehalten werden.

Ein weiteres Problem stellt die Wahrung der Autonomie in Fragen des Schedulings dar: Obwohl hier keine „Entmündigung“ durch einen Meta-Scheduling-Ansatz geschieht, sind die bisherigen Verfahren leider nur dann effektiv einsetzbar, wenn auf LRMS-Ebene ein bestimmter lokaler Schedulingalgorithmus eingesetzt werden kann und damit die Konfigurationsautonomie beschränkt ist (zum Beispiel Backfilling-Varianten bei der Strategie mit aktiver Anfrage). Dies führt zu einem Verlust der Schedulingautonomie einer Site und ist daher ein weiterer Aspekt, der die Einstiegshürde für die Grid-Teilnahme von Ressourcenanbietern erhöht.

Die vielleicht größte Problematik besteht darin, dass viele sensitive Informationen über den aktuellen Systemzustand und die Anforderungen der eingereichten Jobs öffentlich verfügbar gemacht werden müssen, damit einfache Strategien gute Schedulingergebnisse erzielen können. Da Daten über aktuelle und vergangene Auslastung, Priorisierungen von Nutzern oder auch die lokale Schedulingeffizienz für den Wettbewerb relevante Faktoren eines Ressourcenanbieters darstellen, werden sie ausschließlich vertraulich behandelt und können nicht für die Umsetzung von Schedulingentscheidungen auf Grid-Ebene nutzbar gemacht werden. Sollte eine offene Informationspolitik dennoch gefordert werden, so stellt dies nicht selten eine unüberwindbare Hürde für die Teilnahme am Grid dar.

Es wird deshalb in diesem Teil der Arbeit als Herausforderung verstanden, Verfahren zu entwickeln, die selbst in einem dezentralen Szenario mit starken Informationsrestriktionen mindestens genauso gute Schedules im Hinblick auf eine gegebene Zielfunktion erzeugen können, wie dies im unkooperativen Fall für die einzelnen Sites möglich ist. Wird dies nämlich nicht erreicht, so ist die Motivation der Sites in ihrer Rolle als Ressourcenanbieter an einem Grid teilzunehmen, sehr gering, da mindestens eine teilnehmende Site dadurch Nach-

teile hat. Um eine allgemeine Akzeptanz der dezentralen Scheduler zu ermöglichen, müssen folgende Anforderungen erfüllt sein, siehe auch Fölling u. a. [3]:

Vollständige Trennung von globaler Grid- und LRMS-Ebene
Ressourcenbesitzer und Administratoren haben üblicherweise ihre eigenen Betriebskonzepte und setzen oftmals komplexe und über die Jahre angepasste Priorisierungen ihrer Nutzer im LRMS um. Deshalb ist es für die Akzeptanz der Systeme, gerade bei bisher im Grid unerfahrenen Administratoren, unerlässlich, die LRMS-Kontrolle nicht zu beeinträchtigen. Die Grid-Interaktion muss somit vollständig auf Middleware-Ebene geschehen und dabei sowohl für Nutzer als auch für Administratoren vollständig transparent sein.

Unterstützung von Umgebungen mit starken Informationsrestriktionen
Obwohl fast alle Grids einen Informationsdienst für das Auffinden von Sites und die globale Organisation integrieren, werden doch jegliche Informationen über die Maschinen und Ressourcen selbst höchst vertraulich behandelt. Dies gilt besonders für Daten über die Effizienz und den Betriebsstatus der Systeme, wie aktuelle Auslastung, momentane Priorisierungen im LRMS, Antwortzeiten oder vorhergehenden Jobdurchsatz. Solche Informationen können als wettbewerbsrelevant angesehen werden, da die Betreiber der Sites zwar im Grid-Kontext kooperieren, aber generell in Konkurrenz zueinander stehen.

Umsetzung von situationsabhängigen, adaptiven Schedulingentscheidungen
Die Berücksichtigung des aktuellen *lokalen* Systemzustandes[15] sowohl in Bezug auf die Ressourcen als auch in Bezug auf den eingereichten Job ist essenziell für die Umsetzung von Entscheidungen über dessen Annahme oder Abgabe. So erscheint zum Beispiel für das Erreichen einer kurzen AWRT die Annahme von weiteren entfernten Jobs umso weniger sinnvoll, je höher die annehmende Site durch schon eingereichte Jobs belastet ist.

Robustheit und Stabilität in sich verändernden Umgebungen
Auch unter Berücksichtigung von künftigen, noch nicht bekannten (und normalerweise auch nicht voraussagbaren) Jobeinreichungen ist es entscheidend, dass die Scheduler gute Entscheidungen treffen können. Dazu müssen flexible und adaptive Mechanismen integriert sein, die auch auf künftige Arbeitslastanforderungen dynamisch reagieren können. Weiterhin müssen Veränderungen der Grid-Landschaft, wie der Ausfall einer oder mehrerer Sites oder das vollständig egoistische Verhalten eines Partners, nicht nur detektiert werden, sondern auch durch angemessene Veränderung der Entscheidungen beantwortet werden, um so weder die eigene noch die allgemeine Effizienz des Grid zu gefährden.

Architekturen, Entscheidungsstrategien und Algorithmen, die genau diese geforderten Eigenschaften haben und dabei trotzdem sehr gute Schedulingergebnisse bei gleichzeitig hoher Robustheit der Lösungen erzeugen, werden in den nun folgenden Kapiteln entworfen und optimiert. Dazu wird zunächst die grundlegende Architektur der Austauschschnittstelle beschrieben und die Interaktion beziehungsweise Trennung der eingeführten Schichten

[15] Hier sei betont, dass nur lokale Informationen für die lokale Entscheidung genutzt werden und keine lokalen Informationen anderer Sites. Somit ergibt sich hier kein Widerspruch zur vorherigen Forderung.

erläutert. Die grundlegenden Annahmen für eine allgemeine Austauschstrategie werden dann in Abschnitt 11.2 festgelegt und in Abschnitt 11.3 an einer Greedy-Strategie, die als Vergleichsstrategie für folgende Optimierungen dient, umgesetzt und untersucht. In den Kapiteln 12 und 13 werden die Beschreibung der Realisierung mittels Fuzzy-Systemen und die Optimierung mit evolutionären und co-evolutionären Algorithmen dargestellt.

11.1 Architektur der Austauschschnittstelle

Die Architektur der im verbleibenden Teil dieser Arbeit ausschließlich angenommenen Austauschschnittstelle ist eine konsequente Erweiterung der schon in den Abbildungen 5.2 und 6.2 dargestellten Konzepte. Dabei ist eine strikte Trennung zwischen dem rein lokalen Ressourcenmanagement (LRMS) und der globalen Grid-Interaktionsschicht (GRMS) umgesetzt, siehe Abbildung 11.1. Die Kommunikation mit anderen Sites geschieht also ausschließlich über diese Schnittstelle. Hinsichtlich der Informationspolitik ist die Schnittstelle sowohl Teil

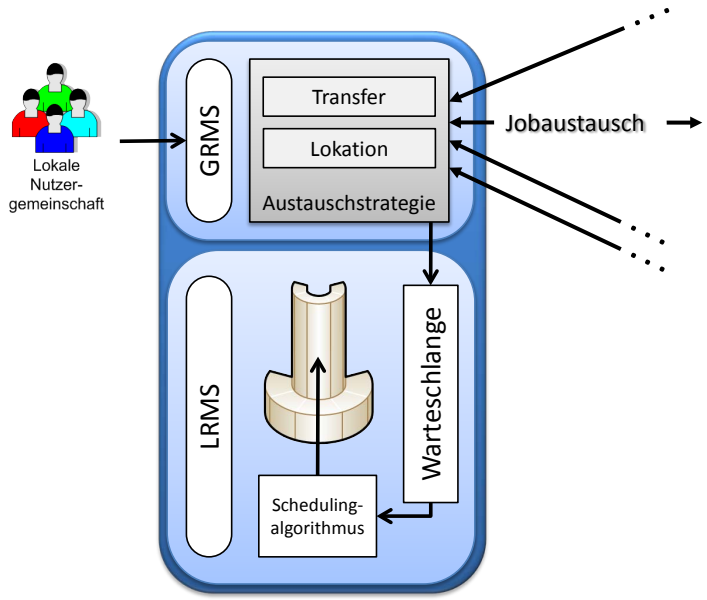


Abbildung 11.1: Dezentrale, zweistufige Grid-Schedulingarchitektur bestehend aus lokalem Ressourcen-Management-System (LRMS) und übergelagertem Grid-Ressourcen-Management-System (GRMS). Die eigentliche Austauschstrategie kann noch in Transfer- und Lokationspolitik unterteilt werden.

des lokalen als auch Teil des globalen Bereiches. Alle im globalen Kontext kommunizierten Informationen (zum Beispiel die Ressourcenanzahl m_k der teilnehmenden Sites) und zusätzlich alle jeweiligen lokalen Systeminformationen der eigenen Site sind für die Entscheidungsfindung verfügbar. Diese lokalen Daten werden aber sicher gekapselt, sodass sie für andere Sites nicht abrufbar sind. Die Autonomie der lokalen Jobverwaltung wird dadurch realisiert, dass die GRMS-Schicht keinen Einfluss auf das LRMS-Scheduling haben kann. Für die LRMS-Komponente sowie für die Nutzer geschieht die Erweiterung auf Middleware-Ebene dadurch vollkommen transparent; das heißt, aus Sicht der Nutzer werden alle Jobs stets lokal gerechnet, auch wenn sie in Wahrheit zu anderen Sites migriert werden. Bevor

nun die Austauschstrategie in allen Details beschrieben wird, sollen Grundannahmen über den Entscheidungsprozess dargestellt werden.
Zunächst einmal sind Jobs, die einmal vom GRMS an das LRMS übergeben wurden, auch nicht mehr für die weitere Delegation verfügbar. Eine Entnahme von Jobs aus dem LRMS ist aus Gründen der strikten Trennung nicht erlaubt. Weiterhin sind die Kommunikationsvorgänge zwischen den Austauschschnittstellen auf Anfragen zur Jobübernahme und die Bestätigung oder Ablehnung derselben beschränkt. Dabei sind Anfragen stets senderinitiativ, sodass keine aktive Anfrage möglich ist. Dies bedeutet aber auch, dass alle eingehenden Anfragen zur Jobübernahme stets an die Bereitschaft der anfragenden Site gekoppelt sind, den fraglichen Job auch wirklich abzugeben. Somit wird der Informationsaustausch zwischen den Sites so gering wie möglich gehalten, da jeder Job nur ein einziges Mal zwischen zwei Austauschschnittstellen ausgehandelt wird. Hat eine Site einen Job übernommen, so obliegt ihr die alleinige Verantwortung für dessen Ausführung, die auch nicht an Dritte delegiert werden kann. Ebensowenig ist es möglich, Ressourcen für einen gewissen Zeitraum, wie zum Beispiel von Fölling u. a. [5] untersucht, unter Sites auszuleihen. Dieses Verfahren verlangt nämlich eine sehr strikte Kopplung zwischen LRMS und GRMS, und eine vollkommen andere Scheduling-Architektur.

11.2 Allgemeine Austauschstrategie

Für die Behandlung lokal eingereichter und extern eingereichter Jobs werden bei der allgemeinen Austauschstrategie unterschiedliche Handlungen berücksichtigt, weshalb beide Fälle getrennt voneinander betrachtet werden. Weiterhin zerfallen Schedulingentscheidungen für lokal eingereichte Jobs in zwei Phasen: In Analogie zu den beim „Load Balancing“ eingesetzten Begriffen, siehe auch Kameda u. a. [102], werden sie als *Lokationspolitik* und *Transferpolitik* bezeichnet. Zusammen bilden beide Politiken eine Austauschstrategie, die innerhalb der Austauschschnittstelle realisiert ist, siehe auch Abbildung 11.1. Abbildung 11.2 zeigt den Entscheidungsprozess für sowohl lokale als auch externe Jobs.

11.2.1 Lokationspolitik

Die Aufgabe der Lokationspolitik ist es, unter den bekannten am Grid teilnehmenden Sites eine bestimmte Prioritätsreihenfolge für die Durchführung von Jobdelegationsanfragen zu erstellen. Die Lokationspolitik wird dann relevant, wenn sich mehr als zwei Sites an einem Grid beteiligen und somit Anfragen in einer gewissen Reihenfolge getätigt werden müssen. In Abbildung 11.2 ist dazu an Ziffer ① zunächst die Lokationspolitik dem eigentlichen Entscheidungsprozess vorgelagert. Es ist somit weiterhin möglich, neben der Priorisierung eine Vorauswahl von Delegationszielen zu integrieren und damit diejenigen Sites von vornherein von einer Anfrage auszuschließen, die für die Ausführung eines bestimmten Jobs über nicht genügend Ressourcen verfügen. Die Lokationspolitik beinhaltet dann zusätzlich noch die Aufgabe einer Vorselektion nach bestimmten Informationen oder Kriterien, wie sie besonders bei sehr großen und heterogenen Grids notwendig wird.
Eine mögliche Umsetzung der Lokationspolitik basiert beispielsweise auf der AWRT der zwischen den jeweiligen Sites migrierten Jobs. Je geringer diese ist, umso höher ist die Priorität bei Anfragen. Dieser Politik liegt die Annahme zugrunde, dass Sites, die in der Vergangenheit für eine schnelle Abarbeitung der Jobs gesorgt haben, dieses auch in Zukunft gewährleisten können. Ein weiteres denkbares Verfahren basiert auf einem Schuldenkonto der

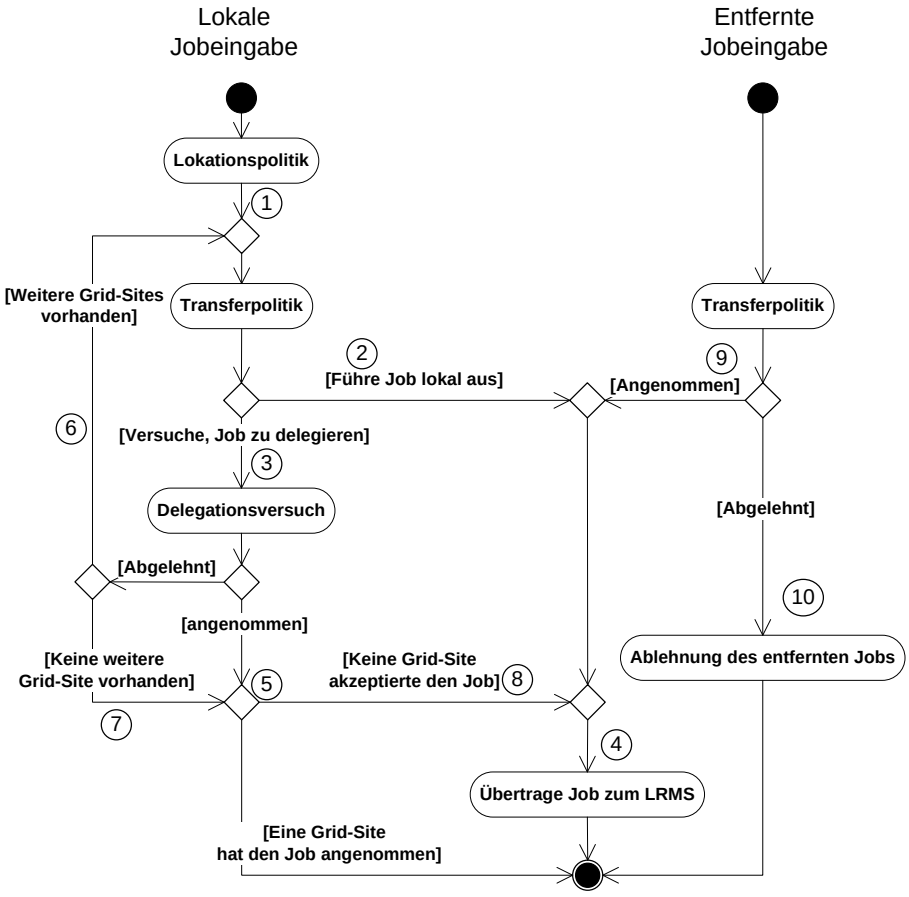


Abbildung 11.2: Entscheidungsprozess für lokal und extern eingereichte Jobs, aufgeteilt in Lokations- und Transferpolitik.

Arbeitslast, wobei diejenigen Sites bei Jobabgaben zuerst angefragt werden, die bei der eigenen Site schon viele Ressourcen extern verbraucht haben und umgekehrt. Weiterhin kann die Priorisierung rein zufällig geschehen, was auch in einigen Fällen zu recht brauchbaren Ergebnissen führt.

11.2.2 Transferpolitik

Nachdem die Lokationspolitik zur Anwendung gekommen ist, werden die weiteren Entscheidungen durch die Transferpolitik koordiniert, die über die Delegation von Jobs entscheidet. Dazu werden, wie schon in Kapitel 9.1 dargestellt, die Akzeptanzpolitik für externe Anfragen und die Distributionspolitik für die Abgabe lokaler Jobs herangezogen. Im allgemeinen Fall wird aber für beide Politiken das identische in Abbildung 11.2 dargestellte Verfahren angewendet:
Bei jeder Konsultation der Transferpolitik wird von ihr eine Entscheidung über die lokale Ausführung eines Jobs ② oder die Durchführung eines Delegationsversuches zu anderen Grid-Sites ③ gefällt. Im ersten Fall wird der Job direkt an das LRMS übergeben ④ und dort entsprechend weiterverarbeitet. Im zweiten Fall werden die bekannten Grid-Sites nach der

von der Lokationspolitik vorgegebenen Reihenfolge für die Übernahme des Jobs angefragt. Dabei kann eine solche Anfrage auf zwei unterschiedliche Arten beantwortet werden:

1. Der Job wird von der angefragten Site zur Ausführung akzeptiert ⑤. In diesem Fall wird der Job einfach zu der entsprechenden Site migriert und es werden keine weiteren Delegationsversuche durchgeführt.

2. Wenn der Job nach erfolgter Anfrage abgelehnt wurde, werden iterativ nach der vorher festgelegten Reihenfolge weitere Partner angefragt ⑥. Diese Prozedur wird so lange ausgeführt, bis entweder der Job übernommen wird (was dann identisch zu Fall 1. ist) oder alle Grid-Sites angefragt wurden ⑦. Wenn keine Site bereit ist, den Job zu übernehmen ⑧, muss die anfragende Site die Ausführung selbst übernehmen und den Job an ihr LRMS übergeben ④.

Weiterhin entscheidet die Transferpolitik auch über die von extern an eine Site herangetragenen Jobs, wobei wiederum zwischen Akzeptanz und Ablehnung gewählt werden kann. Bei Akzeptanz wird der fremde Job angenommen und ans LRMS zur Ausführung übergeben ⑨. In diesem Moment hat die annehmende Site auch die komplette Verantwortung für die Abarbeitung dieses Jobs übernommen. Alternativ kann die Site natürlich die Anfrage auch ablehnen ⑩.
Die hier dargestellte *allgemeine Austauschstrategie* dient als grundsätzliche Vorlage für alle weiteren spezifizierten und optimierten Entscheidungskonzepte. Die Unterschiede liegen dabei immer nur in den Implementierungen der Lokations- beziehungsweise Transferpolitiken; die grundlegenden Abläufe bleiben aber identisch und laufen immer nach dem in Abbildung 11.2 dargelegten Schema ab.

11.3 Eine Greedy-Austauschstrategie (AWF)

Strategien wie die aktive Anfrage, siehe Kapitel 9.1, können nur unter einer sehr offenen Informationspolitik zu wirklich deutlichen Verbesserungen führen, da die Kommunikation möglichst aller gerade verfügbaren Jobs notwendig ist. Somit geben die Partner implizit auch immer Informationen über die Menge ihrer gerade in der Warteschlange befindlichen Jobs preis oder es lassen sich durch eine rege Nachfrage auch von anderen Partnern Rückschlüsse auf die jeweilige lokale Auslastungssituation ziehen.
Deshalb wird hier nun eine Strategie vorgestellt, eine sogenannten *Greedy*-Strategie, bei der Jobs nach einem festen Regelverfahren zwischen den Partnern verhandelt werden. Diese trägt insofern den oben formulierten Anforderungen Rechnung, indem keine weiteren Informationen außer spezifischen Kenngrößen einzelner Jobs zwischen den Partnern ausgetauscht werden müssen. Dies ist deshalb von großer Bedeutung, weil für die im weiteren Teil dieser Arbeit vorgestellten Optimierungen die mit diesem Verfahren erreichten Ergebnisse als Vergleichswerte genutzt werden können.
Es werden identische Verfahren für Akzeptanz- und Distributionspolitik verwendet, sodass wiederum alle an das GRMS übergebene Jobs zunächst gleich behandelt werden. Sobald zum aktuellen Zeitpunkt t ein Job j eingereicht wird, überprüft die Strategie, ob genügend Ressourcen zur unmittelbaren Ausführung des Jobs zur Verfügung stehen, also die Bedingung 11.1, siehe auch Gleichung 7.5, erfüllt ist.

$$m_j \leq m_k \left(1 - \mathrm{U}_k(t)\right) \tag{11.1}$$

Wenn dies der Fall ist, wird der zu entscheidende Job ans LRMS übergeben und direkt in die Warteschlange eingefügt. Ist dies jedoch nicht der Fall, so wird dieser Job in einer später noch zu spezifizierenden Reihenfolge allen Partnern zur Übernahme angeboten, die wiederum über die Annahme nach dem gleichen Verfahren entscheiden. Der einzige Unterschied zwischen lokal eingereichten und entfernten Jobs liegt darin, dass bei versuchter Abgabe und keiner Akzeptanz durch einen Grid-Partner der Job letztlich an der Ursprungs-Site ausgeführt werden *muss*. Da dieses Verfahren sich nur nach der aktuellen Auslastung richtet und nur die Ressourcenanforderungen des Jobs betrachtet und somit quasi „passende Lückenfüller“ gesucht werden, wird es im Folgenden *Accept-When-Fit* (AWF) genannt.

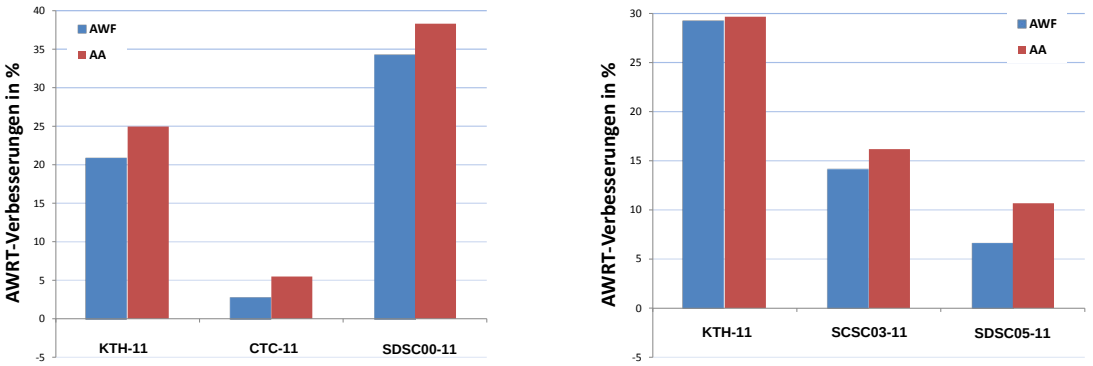
Es müssen durch diese Verfahren keine Informationen über die aktuelle Auslastung kommuniziert werden und auch der Status der Warteschlange bleibt den Partnern verborgen. Nach einer Anfrage muss ein Teilnehmer lediglich die Ressourcenanforderungen des Jobs mitteilen und als Antwort wird dann entweder eine Annahme oder eine Ablehnung ohne weitere Begründung übermittelt. Für die teilnehmenden Grid-Partner bleibt der interne Entscheidungsprozess dabei allerdings vollständig verborgen. Hier werden also zum ersten Mal die geforderten Informationsbeschränkungen genau eingehalten.

Bei AWF werden analog zur aktiven Anfrage die Jobs am Ende der Warteschlange angefügt. Deshalb ist hier auch wieder zu erwarten, dass AWF von der Anwendung von EASY im LRMS stark profitiert, während es bei FCFS schlechtere Ergebnisse erzielen wird. Um die vorgestellte Strategie fair beurteilen zu können, werden die im Folgenden vorgestellten Ergebnisse auch unter Einsatz von EASY erzielt. Die entsprechenden FCFS-Ergebnisse sind dann (allerdings für aufgeteilte Datensätze) in Kapitel 12 dieser Arbeit zu finden, wo die Simulationen konsequent ohne Laufzeitschätzungen durchgeführt werden und somit die Anwendung von EASY nicht mehr infrage kommt. Es sei hier schon vorweggenommen, dass beim Einsatz von FCFS die zu erwartenden Effizienznachteile deutlich zutage treten.

11.3.1 Evaluation des Austauschs mit AWF

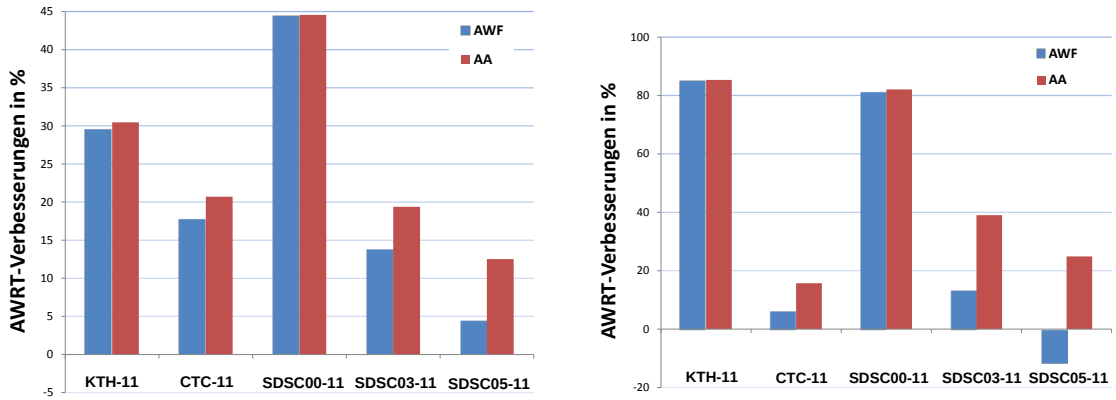
Es werden wie schon in Kapitel 9.2 die drei unterschiedlichen Szenarien mit jeweils drei beziehungsweise fünf Partnern untersucht. Da nun mehr als zwei Partner am Grid teilnehmen, muss eine Reihenfolge festgelegt sein, in der Sites für die Übernahme eines Jobs angefragt werden. Als Lokationspolitik werden die Sites nach einer festen Reihenfolge, nämlich nach jeweils aufsteigender Nummer $k = 1 \ldots K$, angefragt. Die Ergebnisse werden dabei im Vergleich zu einer rein lokaler Ausführung mit EASY angegeben. Zu Vergleichszwecken sind weiterhin die schon oben besprochenen Ergebnisse bei aktiver Anfrage (AA) mit in die entsprechenden Diagramme eingezeichnet.

In Abbildung 11.3 ist dabei zu erkennen, dass auch mit AWF deutliche Verbesserungen gegenüber einer rein lokalen Ausführung erreicht werden. Allerdings sind alle Ergebnisse schlechter als bei der aktiven Anfrage. Gerade bei größeren Sites fallen die Verbesserungen mit AWF deutlich geringer aus. Gleiches ist auch bei Szenario 9 in Abbildung 11.4(a) zu erkennen, wo für kleine Sites ungefähr gleiche Verbesserungen erzielt werden und bei großen deutlich schlechtere. Dies ist dadurch zu erklären, dass große Sites durch die generell größere Menge verfügbarer Ressourcen auch häufiger Jobs annehmen. Dabei berücksichtigt diese Strategie nur ungenügend die genaue Charakteristik der aktuellen Lastsituation an einer Site. Große Sites tendieren deshalb dazu, so viele Jobs anzunehmen, dass sie nur in geringerem Maße vom Grid profitieren können.



(a) Grid-Szenario 5 (b) Grid-Szenario 8

Abbildung 11.3: Vergleich der Verbesserungen der AWRT für Accept-When-Fit (AWF)-Strategie und die Strategie mit aktiver Anfrage (AA) bei Anwendung in den Grid-Szenarios 5 und 8. Es kam jeweils EASY-Backfilling im LRMS zum Einsatz.



(a) Grid-Szenario 9 mit EASY (b) Grid-Szenario 9 mit FCFS

Abbildung 11.4: Vergleich der Verbesserungen der AWRT für Accept-When-Fit (AWF)-Strategie und die Strategie mit aktiver Anfrage (AA) bei Anwendung im Grid-Szenario 9 und unterschiedlichen LRMS-Algorithmen.

Ein entscheidender Nachteil zeigt sich in der schon angesprochenen Abhängigkeit vom LRMS-Schedulingverfahren. In Abbildung 11.4(b) sind dazu die Austauschergebnisse für das Grid-Szenario 9 mit fünf Sites bei lokaler Anwendung von FCFS im Vergleich zu rein lokalem FCFS-Scheduling angegeben. Dabei fällt zunächst auf, dass für KTH und SDSC00 Verbesserungen von über 80 % erreicht werden. Dies liegt aber hauptsächlich daran, dass FCFS lokal sehr schlechte Ergebnisse liefert und sich durch die Umverteilung im Grid sehr schnell große Vorteile ergeben. Weiterhin zeigt die aktive Anfragestrategie (durch die Aufweichung der Informationsrestriktionen) nach wie vor gute Resultate, während wie erwartet die AWF-Strategie durch die rein temporär effizienzsteigernde Auswahl von Jobs zu deutlichen Verschlechterungen für die großen Sites führt. Dieses Verhalten ist insofern kritisch, als dass bei realen Grids oftmals keine Laufzeitschätzungen zur Verfügung stehen, siehe zum

Beispiel Anoep u. a. [28], sodass nur FCFS- oder LIST-Scheduling als LRMS-Algorithmus infrage kommt. Effiziente und flexible GRMS-Schedulingverfahren sind also notwendig, um auch bei lokalem FCFS-Scheduling Verbesserungen im Grid erzielen zu können.
An dieser Stelle sei abschließend auf die Strukturanalyse in Kapitel 10 zurückverwiesen, in dem schon bereits das Verhalten von AWF im Vergleich zur offline erzeugten Pareto-Front untersucht wurde. Im Vergleich zur Pareto-Front erzielte AWF deutlich schlechtere Ergebnisse als alle anderen dort untersuchten Strategien. Als Grund für das schlechtere Abschneiden ist die Einhaltung der restriktiven Informationspolitik zu identifizieren. Es kann also hier schon festgehalten werden, dass mit weniger zur Verfügung stehenden Informationen das Scheduling im Grid deutlich komplizierter wird und somit auch weniger gute Schedules im Hinblick auf die AWRT erreicht werden können. Die Optimierung des Grid-Scheduling unter restriktiven Informationen birgt also offensichtlich noch erhebliches Verbesserungspotenzial.

12
Realisierung mittels evolutionärer Fuzzy-Systeme

ABSTRAKT BETRACHTET sind nach dem im vorigen Kapitel vorgestellten Modell Grid-Sites mit ihren im GRMS integrierten Mechanismen unabhängige Agenten, die in einer gemeinsamen Umwelt interagieren und Entscheidungen zu treffen haben. Dabei ergibt sich die zusätzliche Schwierigkeit, dass sie nur eine eingeschränkte Sicht der Umwelt haben. Diese Einschränkung wird zur Wahrung der eigenen Privatsphäre von allen Partnern akzeptiert. So ist nur der eigene Zustand beliebig genau bekannt und global sind nur rudimentäre Beschreibungen der Jobs verfügbar.

In einem solchen Kontext wurde gezeigt, dass statische Entscheidungs- und Austauschverfahren unter bestimmten lokalen Konfigurationen gute Resultate erzielen. Werden diese Konfigurationen aber verändert und passen diese nicht mehr zu den Rahmenbedingungen, die vom GRMS gefordert sind, so ist zu erwarten, dass statische Verfahren versagen. Ein wichtiger Aspekt der LRMS-Konfiguration ist zum Beispiel die Forderung nach Laufzeitschätzung zur Anwendung von Backfilling-Methoden. Gerade in einem Grid kann üblicherweise nicht auf Laufzeitschätzungen zurückgegriffen werden, da Nutzer keinen Einfluss auf den tatsächlichen Ausführungsort ihrer Jobs haben. Deshalb ist es ihnen auch nicht möglich, die Laufzeit ihrer Jobs abzuschätzen, was den Einsatz von Backfilling-Methoden im LRMS ausschließt. Somit ist es notwendig, alternative Verfahren zu entwickeln, die es ermöglichen, auch in unbekannten oder nur vage beschreibbaren Umgebungen dynamische Schedulingentscheidungen zu treffen.

Es ist daher im Grid notwendig, flexibel auf unterschiedliche Situationen reagieren zu können und wenig auf ein rein statisches Verfahren zu vertrauen. In den letzten Jahren hat sich dazu das regelbasierte Scheduling, siehe Franke [76], als erfolgreiches Konzept etabliert. Regelbasierte Scheduler sind in der Lage, ihr algorithmisches Verhalten je nach aktuellem Systemzustand dynamisch zu verändern. Die Erstellung von Regelsystemen ist allerdings sehr komplex und erfordert die Unterstützung von effektiven Lernalgorithmen. Ein Problem ist weiterhin, dass die Partner in einem Grid in wechselseitigen Abhängigkeiten stehen, die es gleichzeitig zu berücksichtigen gilt. Dies kann durch einen einfachen evolutionären Algorithmus allein nicht geleistet werden, weshalb ein spezielles Lernkonzept mit dem Ziel zum Einsatz kommt, eine möglichst große und gleichzeitig robuste Verbesserung für alle Grid-Sites zu erzeugen.

Besonders die Repräsentation der Regelmechanismen durch Fuzzy-Systeme hat sich in neueren Entwicklungen, zum Beispiel von Franke u. a. [8, 9], in ihrer Anwendung auf Parallelrechner als sehr effizient erwiesen. Da es unter den gegebenen starken Informationseinschränkungen oftmals unmöglich ist, einen Systemzustand mit hoher Genauigkeit zu be-

schreiben, sind deshalb Verfahren erforderlich, die auch mit unscharfen Repräsentationen zu verlässlichen Entscheidungen kommen. Ein weiteres Problem stellt hier allerdings das Lernen von Entscheidungen dar, weshalb evolutionäre Fuzzy-Systeme, siehe Kapitel 4, zum Einsatz kommen werden. Die Schwierigkeit ist jedoch, dass für die Bewertung einer Entscheidung erst ein gesamtes Schedulingszenario evaluiert werden muss und Einzelentscheidungen nicht unmittelbar bewertbar sind. Deshalb wird hier das Verfahren des Kritikerlernens anhand vorgegebener Beispielszenarien eingesetzt, siehe Sutton und Barto [168], und auf die Entwicklung möglichst robuster und stabiler Verfahren Wert gelegt.

Die hier vorgestellten Strategien wenden nun diese Konzepte auf das Scheduling in Computational Grids an und führen notwendige Erweiterungen für die Interaktion zwischen unabhängigen Grid-Sites ein, siehe auch Fölling u. a. [4].

12.1 Regelbasiertes Scheduling

Die Idee, Schedulingentscheidungen nicht durch statische Algorithmen oder Heuristiken zu fällen, sondern flexibel je nach Situation anzupassen, wurde schon seit einiger Zeit hauptsächlich für MPP-Systeme, siehe auch Franke u. a. [7], aber auch im Grid-Kontext, siehe zum Beispiel Ernemann u. a. [57] oder Buyya u. a. [38], untersucht. Im Grid hat sich dieses Konzept aber, besonders durch die Arbeiten von Buyya u. a. [39] vorangetrieben, speziell in Richtung des sogenannten „Economic Grid-Scheduling“ entwickelt. Primäres Ziel ist es dort, die Dynamik eines Marktes, bestehend aus dem Wechselspiel von Angebot und Nachfrage, auf die teilnehmenden Grid-Sites zu übertragen und somit ein Schedulingkonzept zu erstellen, welches durch „Marktregeln“ repräsentiert wird. In den jetzigen Realisierungen benötigen diese Systeme aber immer zentrale Broker, ähnlich zu dem in Kapitel 8 dargestellten Konzept, die für die Vermittlung der Teilnehmer sorgen. Dort werden dann alle Informationen über das Grid zentral vorgehalten. Außerdem ist es notwendig, für ein derartiges System detailliert Kostenfunktionen für die Abarbeitung von Jobs unter gegebenen Randbedingungen zu formulieren, die in realen Grids so nicht verfügbar sind. Da solche Konzepte mit den in dieser Arbeit angenommenen Bedingungen und Voraussetzungen nicht vereinbar sind, wird hier ein viel allgemeineres, regelbasiertes Konzept umgesetzt.

Wie im vorherigen Kapitel verdeutlicht, beschränkt sich die Entwicklung eines Grid-Schedulers unter Annahme der eingeführten Architektur und der allgemeinen Austauschstrategie, siehe Kapitel 11.2, im GRMS auf Verfahren für (binäre) Entscheidungen über Annahme oder Ablehnung von Jobs. Da dies abhängig vom aktuellen Zustand der Site geschehen muss, ist es zunächst notwendig, diesen Zustand durch möglichst aussagekräftige *Zustandskenngrößen* zu beschreiben. Um die Komplexität des Regelsystems möglichst gering zu halten, ist es sinnvoll, sich auf wenige Kenngrößen zu beschränken, siehe Abschnitt 12.3.4. Ziel ist es nun, auf Basis des jeweiligen so beschriebenen Zustandes eine passende Entscheidung zu treffen.

Dazu wird ein Regelsystem eingeführt, dass sich aus einer Menge unterschiedlicher Regeln zusammensetzt. Jede einzelne Regel besteht dabei aus einer sogenannten *Bedingung* oder Regelprämisse und einer entsprechenden *Konsequenz*. Unter Einsatz einer unscharfen Beschreibung durch linguistische Terme, wie sie bei Fuzzy-Reglern möglich ist, lauten derartige Regeln:

WENN *Warteschlange sehr lang*	und *Jobparallelität sehr groß*	DANN *lehne Job ab*
⋮	⋮	⋮
WENN *Warteschlange leer*	und *Jobparallelität groß*	DANN *nimm Job an*

Dabei können, wie in diesem Beispiel, auch mehrere Bedingungen zu einer Prämisse konjunktiv verknüpft werden. Eine komplette Regelbasis besteht nun aus einer Menge von Regeln, die dabei alle möglichen Systemzustände abdecken. Eine sehr elegante Möglichkeit, dies umzusetzen und dabei gleichzeitig einen robusten Entscheider durch die Unterstützung vager Zustandsbeschreibungen zu erreichen, erlaubt der Einsatz von Fuzzy-Systemen.

12.2 Grid-Scheduler auf Basis von Fuzzy-Systemen

Für den Entwurf einer GRMS-Entscheidungspolitik und der damit verbundenen Austauschstrategie wird hier das Takagi-Sugeno-Kang-(TSK-)Modell [169] genutzt, siehe Kapitel 4.2. Das Fuzzy-System stellt den Kern des Entscheiders zur Realisierung der Transferpolitik im GRMS dar und wird immer dann konsultiert, wenn ein neuer Job lokal eingereicht wird oder von anderen Grid-Sites angeboten wird. In all diesen Fällen besteht die Möglichkeit, dass sich der lokale Systemzustand ändert, worauf dann mit angepassten Entscheidungen reagiert wird.

12.2.1 Architektur des Entscheiders

Das allgemeine TSK-Modell besteht nun aus einer Anzahl von N_r WENN-DANN-Regeln R_i in der folgenden Form:

$$\begin{aligned} R_i \quad := \quad & \text{WENN } x_1 \text{ ist } g_i^{(1)} \text{ und } x_2 \text{ ist } g_i^{(2)} \text{ und } \ldots \text{ und } x_{N_f} \text{ ist } g_i^{(N_f)} \\ & \text{DANN } y_i = b_{i0} + b_{i1}x_1 + \ldots + b_{iN_f}x_{N_f} \end{aligned} \tag{12.1}$$

In dieser Definition bezeichnen $x_1, x_2, \ldots, x_{N_f}$ die Eingangsvariablen als Elemente eines Vektors $\vec{x}$. Diese Eingangsvariablen stellen die echten aktuellen Messwerte der Zustandskenngrößen dar, die jeweils im System erhoben werden. Weiterhin stellt y_i die entsprechende lokale Ausgangsvariable oder auch Konsequenz der Regel dar.
Durch die Zugehörigkeitsfunktion $g_i^{(h)}$, die also die h-te Eingangs-Fuzzy-Menge darstellt, wird der *Zugehörigkeitsgrad* $g_i^{(h)}(x_h)$ der aktuellen Messwerte zur Kenngröße h in Bezug auf eine Regel R_i beschrieben. Durch sie muss ein funktionaler Zusammenhang zwischen jedem Punkt im Zustandsraum und einem Intervall $[0, 1]$ hergestellt werden, durch den jeder mögliche Kenngrößenwert auf diesem Intervall abgebildet wird, wobei der Wert 1 der größten Zugehörigkeit und der Wert 0 gar keiner Zugehörigkeit entspricht, siehe Kapitel 4.1. Für Regeln, die in ihrem Bedingungsteil genau N_f Kenngrößen zur Zustandsbeschreibung berücksichtigen, werden nach Definition 12.1 auch genau N_f Zustandsfunktionen benötigt.
Weiterhin wird die lokale Ausgangsvariable y_i über die reellen Parameter b_{ih} durch Linearkombination mit den Eingangsvariablen $\vec{x}$ verknüpft. Allgemein kann somit nun die Ausgabe eines Reglers (oder die Entscheidung eines Entscheiders) $y_D(\vec{x})$ durch 12.2 für eine gesamte Regelbasis über alle Regeln bestimmt werden.

$$y_D(\vec{x}) = \frac{\sum_{i=1}^{N_r} \phi_i(\vec{x}) y_i}{\sum_{i=1}^{N_r} \phi_i(\vec{x})} = \frac{\sum_{i=1}^{N_r} \phi_i(\vec{x})(b_{i0} + b_{i1}x_1 + \ldots + b_{iN_f}x_{N_f})}{\sum_{i=1}^{N_r} \phi_i(\vec{x})} \quad (12.2)$$

Dabei ist $\phi_i(\vec{x})$ der *Erfüllungsgrad* einer bestimmten Regel R_i für einen gegebenen Vektor $\vec{x}$ von Eingangsvariablen, der durch Definition 12.3 angegeben ist.

$$\phi_i(\vec{x}) = g_i^{(1)}(x_1) \wedge g_i^{(2)}(x_2) \wedge \ldots \wedge g_i^{(N_f)}(x_{N_f}) = \prod_{h=1}^{N_f} g_i^{(h)}(x_h) \quad (12.3)$$

Es wird dazu eine konjunktive Verknüpfung hergestellt, die es ermöglicht, den Systemzustand beliebig genau durch mehrere Zustandskenngrößen zu beschreiben. Der Erfüllungsgrad $\phi_i(\vec{x})$ ist demnach ein Maß für die Gültigkeit einer Regel in einer bestimmten Situation und für das Vertrauen, das der Konsequenz dieser Regel für die Bestimmung der Reglerausgabe zukommt. Für die Bestimmung der Ausgabe wird also zunächst die Konsequenz jeder einzelnen Regel mit ihrem jeweiligen Erfüllungsgrad für einen Eingangsvektor $\vec{x}$ über alle Zustandskenngrößen gewichtet. Dann wird über alle Regeln das gewichtete Mittel bestimmt und somit eine Ausgabe $y_D(\vec{x})$ errechnet, siehe Gleichung 12.2. Das speziell für ei-

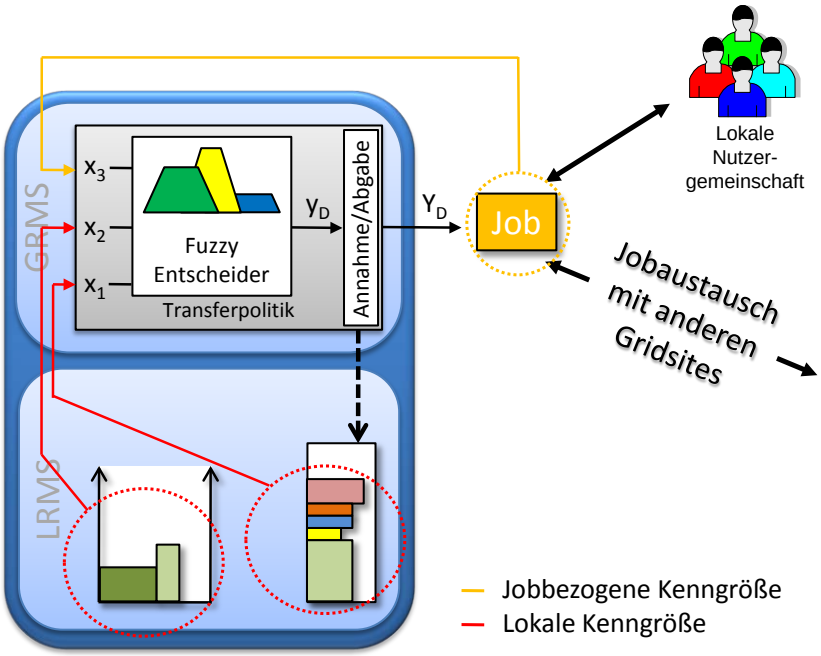


Abbildung 12.1: Aufbau eines dezentralen zweistufigen Grid-Schedulers mit einer Fuzzy-basierten Realisierung der Transferpolitik. Die Eingangsgrößen $\vec{x}$ berücksichtigen nur lokale und jobbezogene Kenngrößen zur Zustandsbeschreibung.

ne Grid-Transferpolitik angepasste Fuzzy-Regler-Konzept ist in Abbildung 12.1 dargestellt. Dort werden sowohl lokale Systemkenngrößen als auch Job-bezogene Kenngrößen durch

Messwerte in den Entscheider eingegeben. Dieser berechnet daraus dann eine Entscheidung über Abgabe oder Annahme des Jobs. Hier wird noch einmal die deutliche Trennung zwischen LRMS und GRMS deutlich, wobei wirklich nur Job-bezogene Kenngrößen im Grid kommuniziert werden. Eine Einwirkung des Fuzzy-Entscheiders auf die Handlungen des LRMS ist ebenfalls weder möglich noch notwendig.

12.3 Anpassung eines evolutionären Fuzzy-Systems

Dieses Vorgehen muss nun an die Erfordernisse der hier angegangenen Problemstellung weiter angepasst werden, was sich sowohl auf die Wahl der Zugehörigkeitsfunktion als auch auf die Auswahl von möglichst aussagekräftigen Zustandskenngrößen bezieht. Dabei stehen besonders die notwendigen Anpassungen für die spätere Optimierung der Fuzzy-Systeme durch einen evolutionären Algorithmus im Vordergrund.

12.3.1 Auswahl der Zugehörigkeitsfunktion

Das TSK-Modell erlaubt generell, wie oben dargestellt, die Beschreibung von Zugehörigkeiten durch jede gültige Zugehörigkeitsfunktion. Allerdings gibt es zwei wichtige Anforderungen, die bei der Wahl zu berücksichtigen sind: Die Funktion sollte eine große Flexibilität aufweisen, sodass es möglich ist, mit einer einzigen Funktion einen großen Teil des Zustandsraumes abzubilden. Somit bleibt die Anzahl der benötigten Regeln möglichst klein. Weiterhin sollte die Zugehörigkeitsfunktion eine möglichst „geschmeidige“ Form haben, sodass bei der Überlagerung keine schroffen Übergänge entstehen. Mit Blick auf die Menge der zu optimierenden Parameter sollte die Zugehörigkeitsfunktion möglichst einfach beschreibbar sein, da zusätzliche Objektparameter auch immer eine Vergrößerung des Suchraumes zur Folge haben. Dies kann dann wiederum zur Einschränkung der durch einen evolutionären Algorithmus erreichbaren Lösungsqualität führen.

Unter den verschiedenen Funktionen sind zwar Dreiecks- und Trapezform die am häufigsten verwendeten, allerdings sind sie schlecht dazu geeignet, einen umfangreichen Teil des Zustandsraumes abzudecken, was dann zu einer großen Zahl nötiger Regeln führt. Außerdem erzeugen derartige Funktionen kantige Übergänge bei der Überlagerung, weshalb Alternativen bevorzugt werden sollten.

Aufgrund ihrer Stetigkeit und der kurzen und kompakten Beschreibung über nur zwei Parameter (Mittelwert und Varianz) wird die Gaußfunktion gerne als Zugehörigkeitsfunktion genutzt. Diese Darstellung hat den Vorteil, dass sie keine Kanten im Verlauf aufweist und an allen Punkten nicht Null ist, wodurch schon mit einer Zugehörigkeitsfunktion[16] der gesamte Raum abgedeckt ist. Somit lässt sich vermuten, dass die Anzahl der benötigten Regeln zur ausreichenden Abdeckung des Zustandsraumes auch vergleichsweise klein ist.

[16] Unter der praktischen Einschränkung, dass die Genauigkeit der rechnerinternen Zahlendarstellung berücksichtigt werden muss, die keine beliebig kleinen Werte zulässt.

12.3.2 Repräsentation der Regeln

Jede Zustandskenngröße h aller N_f Zustandskenngrößen einer Regel R_i wird nun durch eine $(\gamma_i^{(h)}, \sigma_i^{(h)})$-Gauß'sche Zugehörigkeitsfunktion (GZF)[17] ohne Normalisierung modelliert, siehe Definition 12.4.

$$g_i^{(h)}(x) = \exp\left\{\frac{-(x-\gamma_i^{(h)})^2}{{\sigma_i^{(h)}}^2}\right\} \tag{12.4}$$

Diese Funktion ist durch die beiden Parameter $\gamma_i^{(h)}$ und $\sigma_i^{(h)}$ vollständig definiert. Der $\gamma_i^{(h)}$-Wert setzt den Mittelwert der Funktion fest, während $\sigma_i^{(h)}$ den Einflussbereich innerhalb des Zustandsraumes dieser Zugehörigkeitsfunktion beeinflusst. Mit anderen Worten wird mit größeren Werten für $\sigma_i^{(h)}$ die Form der GZF erweitert, während die Spitze der Glockenkurve konstant beim Wert 1 bleibt. Durch diese besondere Eigenschaft der GZF ist es nun möglich, den Einfluss der Regel auf eine gewisse Kenngröße durch die Variation von $\sigma_i^{(h)}$ genau zu steuern. Die Variation der Parameter beeinflusst somit die unterschiedlich starke Gewichtung der Empfehlungen einzelner Regeln und ihre Entscheidungen im gesamten Zustandsraum.
Durch die Beschreibung der Kenngrößen durch GFZ ist es bei der Nutzung von $\gamma_i^{(h)}$ und $\sigma_i^{(h)}$ nach dem Konzept von Juang u. a. [100] beziehungsweise Jin u. a. [99] möglich, den Bedingungsteil einer Regel, wie in Abbildung 12.2 dargestellt, für einen evolutionären Algorithmus zu kodieren. Da der Entscheider nur über die Annahme/Ablehnung von Jobs entscheiden muss, kann die Konsequenz einer Regel unter Berücksichtigung rein binärer Entscheidungen vereinfacht werden. Daher wird hier die Annahme durch den Wert 1 und die entsprechende Ablehnung durch den Wert -1 repräsentiert. Durch diese binäre Entscheidung werden alle Parameter außer b_{i0} in Gleichung 12.2 gleich null gesetzt, wodurch sich die TSK-Ausgabe einer Regel R_i zu $y_i = b_{i0}$ vereinfacht und die Ausgabevariable entsprechend durch

$$y_i = \begin{cases} 1, & \text{wenn Job } \textit{angenommen} \text{ werden soll} \\ -1, & \text{wenn Job } \textit{abgegeben} \text{ werden soll} \end{cases} \tag{12.5}$$

interpretiert wird.
Dieses Konzept erlaubt die Kodierung einer einzelnen Regel als eine Zeichenkette mit $(2 \cdot N_f)$ reellwertigen Parametern und *einer* ganzzahligen Variablen, siehe Abbildung 12.2. Eine gesamte Regelbasis kann dann durch einfaches Aneinanderfügen der Einzelregeln kodiert werden. So ist eine Regelbasis, bestehend aus N_r Regeln, vollständig durch eine Anzahl von

$$l = N_r(2 \cdot N_f + 1) \tag{12.6}$$

Parametern beschrieben, siehe Gleichung 12.6. Dieses Kodierungsschema ist somit ideal geeignet, um als Individuenrepräsentation innerhalb eines evolutionären Algorithmus zu dienen, wobei jedes Individuum aus genau l Objektparametern besteht. Dieses Kodierungskonzept entspricht dem in Kapitel 4.3 beschriebenen Pittsburgh-Ansatz nach Smith [162].

[17] Im Gegensatz zur sonst üblichen Notation wird in dieser Arbeit der Mittelwert mit γ bezeichnet, um Überschneidungen mit dem Buchstaben μ, der schon als Populationsgröße in Evolutionsstrategien besetzt ist, zu vermeiden.

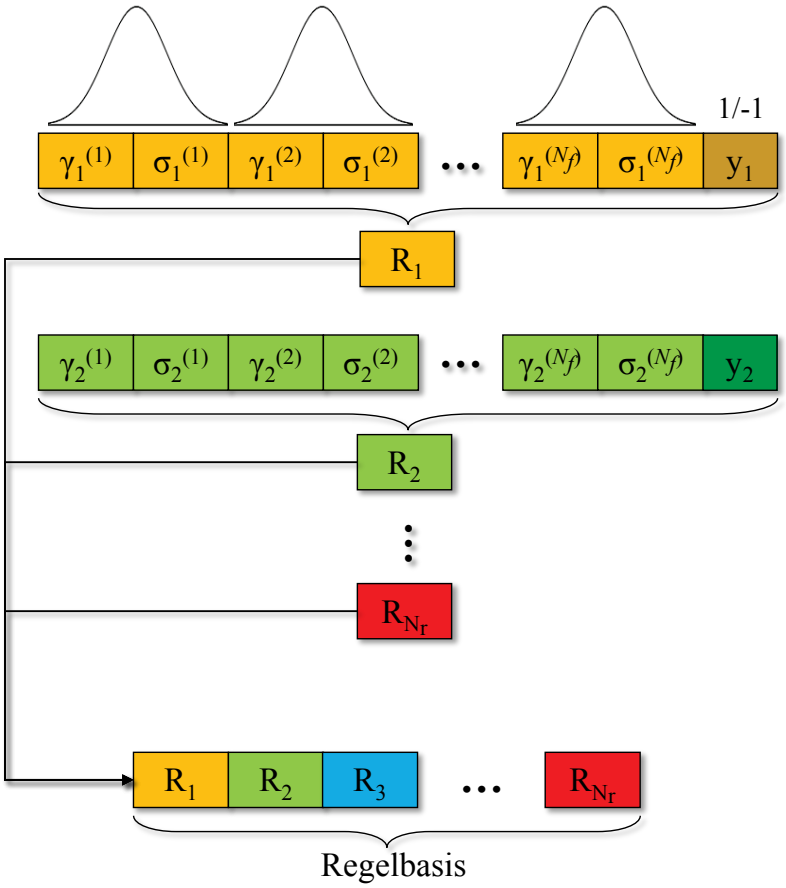


Abbildung 12.2: Kodierungsvorschrift für einzelne Regeln und die daraus resultierende Konstruktion einer gesamten Regelbasis durch Verknüpfung (Pittsburgh-Ansatz).

12.3.3 Berechnung der Entscheidung bei GZF-Kodierung

Nachdem schon das allgemeine Vorgehen zur Bestimmung der Entscheidung für einen gegebenen Eingangsvektor $\vec{x}$ beschrieben wurde, wird hier auf die angepasste Realisierung bei GFZ eingegangen. Analog zum allgemeinen Modell, siehe Gleichung 12.3, wird wiederum durch konjunktive Verknüpfung der Erfüllungsgrad $\phi_i(\vec{x})$ unter Berücksichtigung der GZF-Beschreibung für Regel R_i nach Gleichung 12.7 berechnet.

$$\phi_i(\vec{x}) = \bigwedge_{h=1}^{N_f} g_i^{(h)}(x_h) = \prod_{h=1}^{N_f} \exp\left\{-\frac{(x_h - \gamma_i^{(h)})^2}{\sigma_i^{(h)2}}\right\} = \exp\left\{-\sum_{h=1}^{N_f} \frac{(x_h - \gamma_i^{(h)})^2}{\sigma_i^{(h)2}}\right\} \qquad (12.7)$$

Dabei entspricht $g_i^{(h)}(x_h)$ dem h-ten Zugehörigkeitsgrad und x_h der h-ten Eingangskenngröße. Dann wird wie in Gleichung 12.2 die Ausgabe $y_D(\vec{x})$ über alle Regeln durch Gewichtung mit dem jeweiligen Erfüllungsgrad berechnet und schließlich die endgültige Entscheidung Y_D durch ausschließliche Betrachtung des Vorzeichens, siehe Gleichung 12.8, bestimmt.

$$Y_D = \mathrm{sgn}(y_D(\vec{x})) \qquad (12.8)$$

Es entspricht dabei $Y_D > 0$ der Annahme eines Jobs und $Y_D \leq 0$ einer Ablehnung. Rein formal wird $Y_D = 0$ ebenfalls als Ablehnung interpretiert.

12.3.4 Auswahl von Zustandskenngrößen

Die Auswahl sinnvoller Zustandskenngrößen ist neben dem Optimierungsverfahren entscheidend für die Effizienz des Grid-Schedulers. Die Anzahl der Kriterien, die für die Zustandsbeschreibung genutzt werden, erhöht dabei die Komplexität des Entscheidungsmechanismus, da sich dadurch auch die Anzahl der möglichen Systemzustände vergrößert. Gleichzeitig soll aber auch jeder Systemzustand möglichst genau abgebildet werden. Zusätzlich erfordert die restriktive Informationspolitik, dass sich die verwendeten Kenngrößen lokal berechnen lassen und kein Austausch zwischen den Grid-Schedulern notwendig wird. Die folgende Auswahl von Kenngrößen hat sich im Rahmen der Entwicklung des dezentralen Schedulingsystems als geeignet erwiesen. Als Notation werden im folgenden Jobs, die zum Zeitpunkt t in der Warteschlange ν auf Site k verweilen, als $j \in \nu_k(t)$ bezeichnet.
Für die Beschreibung des Systemzustandes werden nur $N_f = 2$ Zustandskenngrößen benötigt, die damit den Bedingungsteil einer Regel bilden. Um den lokalen Systemzustand mit nur einer Kenngröße abzubilden, wird die *normalisierte wartende Parallelität* an Site k (NWP_k), siehe Gleichung 12.9, definiert.

$$\mathrm{NWP}_k(t) = \frac{1}{m_k} \sum_{j \in \nu_k(t)} m_j \tag{12.9}$$

Diese Kenngröße dient als Maßstab für die erwartete Nutzung der vorhandenen Prozessoren durch alle bisher eingereichten und noch wartende Jobs[18] im Verhältnis zur maximal verfügbaren Prozessorenzahl m_k an Site k. Diese spiegelt zum einen die Effizienz des gerade aktiven LRMS wider und ist zum anderen aufgrund der positiven Korrelation zwischen Parallelitätsgrad und zu erwartender Joblaufzeit, siehe Kapitel 7.1, eine grobe Schätzung der in naher Zukunft zu erwartenden Arbeitsbelastung.
Die zweite Kenngröße bezieht sich auf den gerade eingereichten Job, für den eine Entscheidung zu treffen ist. Das Verhältnis zwischen der Anzahl der durch diesen Job angeforderten Prozessoren m_j und den maximal auf der aktuellen Site k verfügbaren Ressourcen m_k wird ausgedrückt durch die *normalisierte Job-Parallelität* (NJP), siehe Gleichung 12.10.

$$\mathrm{NJP}_j = \frac{m_j}{m_k} \tag{12.10}$$

Mit diesen beiden Kenngrößen wird im Folgenden jeder mögliche (zumindest für das Scheduling relevante) Systemzustand approximiert. Die Bedeutung der Überlagerung wird dazu hier noch einmal beispielhaft auf graphische Art und Weise verdeutlicht: Bei zwei Zustandskenngrößen kann der gesamte Zustandsraum, wie in Abbildung 12.3 gezeigt, als zweidimensionale Ebene dargestellt werden. Es ist zu beachten, dass NWP prinzipiell nicht im Wertebereich beschränkt ist, aber in der Praxis Werte, die größer als 10 sind, nur äußerst selten auftreten. Daher wurde die x-Achse in dieser Abbildung auf den angegebenen Wertebereich eingeschränkt.

[18] Im Gegensatz zur echten Laufzeit ist die angeforderte Prozessorenzahl m_j auch bei der hier betrachteten Online-Problematik schon zum Einreichzeitpunkt bekannt.

Jede Regel deckt nun einen bestimmten Bereich für jede Kenngröße ab, der durch die jeweilige GZF und die entsprechenden γ und σ Parameter festgelegt ist. In Abbildung 12.3 entsprechen die blauen Linien jeweils der GZF für Regel R_1, während die roten Linien Regel R_2 darstellen. Die Regeln sind dort jeweils mit ihren GZF für beide Kenngrößen dargestellt, wo-

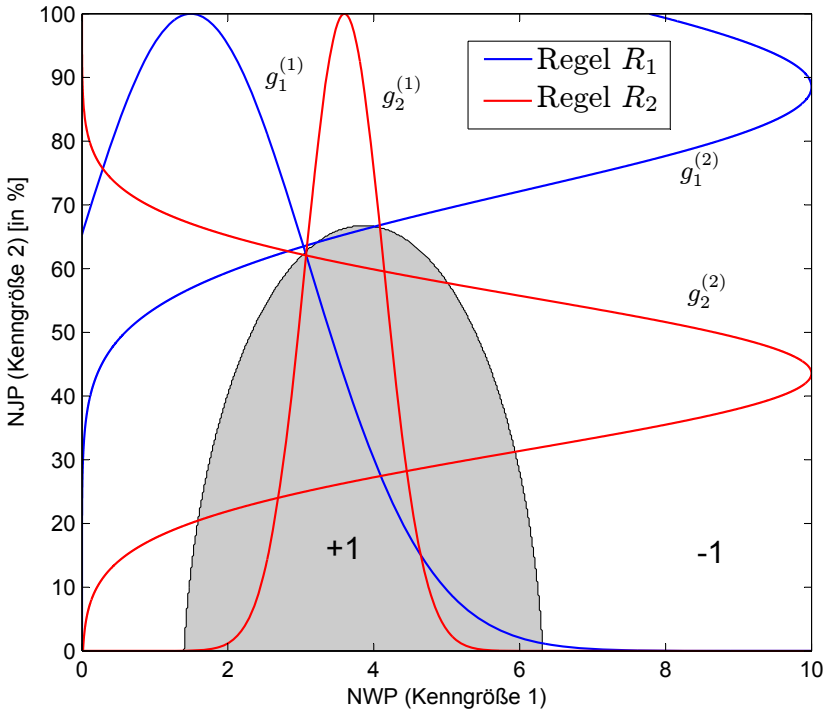


Abbildung 12.3: Zweidimensionaler Zustandsraum nach der Überlagerung von zwei Beispielregeln R_1 und R_2.

bei sich zum Beispiel $g_1^{(1)}$ auf NWP und $g_1^{(2)}$ auf NJP für die Regel R_1 bezieht. Analog sind die GZF für Regel R_2 als $g_2^{(1)}$ und $g_2^{(2)}$ dargestellt. Weiterhin ist eine Projektion des entsprechenden Kennlinienfeldes, das aus der Überlagerung der beiden Regeln entsteht, als graue Fläche im Zustandsraum eingetragen.

Es sei $y_1 = -1$, das heißt, der Ausgabewert für R_1 ist gleich -1 und $y_2 = +1$. So kann der gesamte Zustandsraum in zwei Domänen unterteilt werden: Innerhalb des grau unterlegten Bereiches entscheidet der Scheduler für beliebige Eingangsvektoren $\vec{x}$ sich für die Annahme eines Jobs (+1), während der Scheduler im restlichen Bereich Jobs ablehnt (-1). Bedingt durch die Projektion sind hier nur scharfe Kanten zu sehen, die sich natürlich aus gleitenden Übergängen zwischen -1 und +1 ergeben, siehe auch Abbildung 12.4. Da für die Entscheidung letztlich nur das Vorzeichen von $y_D(\vec{x})$ berücksichtigt wird, siehe Gleichung 12.8, ist diese diskretisierte Darstellung möglich.

12.3.5 Konfiguration des evolutionären Fuzzy-Systems

Vor der Darstellung der Simulationsergebnisse werden die Konfiguration des evolutionären Fuzzy-Systems sowie die weitere Evaluationsumgebung beschrieben. Für das evolutionäre Fuzzy-System wurde eine feste Anzahl von $N_r = 10$ Regeln verwendet. Diese Wahl ist durch die vorherigen Untersuchungen von Franke u. a. [9] gerechtfertigt, da dort bei ähnlich gelagerten Problemen basierend auf dem Pittsburgh-Ansatz bereits gute Ergebnisse mit nur fünf bis zehn Regeln erzielt wurden. Da eine gesamte Regelbasis in ein Individuum kodiert wird, ist nach Gleichung 12.6 vom evolutionären Algorithmus ein Problem mit $N_r \cdot (N_f \cdot 2 + 1) = 10 \cdot (2 \cdot 2 + 1) = 50$ Objektparametern zu lösen.
Als evolutionäres Verfahren wurde eine $(\mu + \lambda)$-Evolutionsstrategie verwendet. Während eines Laufes von 150 Generationen konnte eine kontinuierliche Verbesserung der Fitnesswerte beobachtet werden. Nach der üblicherweise verfolgten Empfehlung von Schwefel [152] ist für die Evolutionstrategien ein Verhältnis von $\mu : \lambda = 1 : 7$ herzustellen. Daher wurde hier eine Elternpopulation mit $\mu = 13$ Individuen verwendet, was in $\lambda = 91$ Nachkommen resultiert. Folglich müssen in jeder Generation 91 Individuen auf ihre Fitnesswerte hin evaluiert werden. Da zu diesem Zweck jedes Mal ein vollständiges Grid simuliert werden muss, ist diese Auswertung trotz Nutzung der effektiven Teikoku Grid-Scheduling-Framework Umgebung immer noch sehr zeitintensiv. Der Aufwand variiert je nach momentaner Parameterkonfiguration von einigen Minuten bis zu einer halben Stunde. Die Auswertung einer Population wurde zusätzlich für den Einsatz auf einem Cluster mit 200 Knoten (Pentium IV, 2,4Ghz Prozessoren) parallelisiert.
Für die Variationsoperatoren wurden weiterhin die folgenden Einstellungen verwendet: Die Mutation wird mit einer einzelnen Schrittweite für jede der zwei Kenngrößen durchgeführt. Dazu wird eine deterministische Selbstanpassung als Zweipunktoperator mit $\beta = 0,6$ genutzt. Da sich die beiden Kenngrößen in ihrem möglichen Wertebereich im Verhältnis $10 : 1$ unterscheiden, siehe Abbildung 12.3, wurde eine initiale Mutationsschrittweite von $\sigma_m = 0,01$ für NWP und $\sigma_m = 0,001$ für NJP gewählt. Diese Mutation wurde aber ausschließlich auf den Bedingungsteil der Regel angewendet, da es sich nur bei diesem um reellwertige Parameter handelt. Für den binären Konsequenzteil wurde die Mutation durch zufällige Veränderung von -1 zu $+1$ oder umgekehrt realisiert. Weiterhin wurde eine diskrete Rekombination zwischen den Individuen in jedem Reproduktionsschritt für die Objektparameter und eine intermediäre Rekombination für die Strategieparameter verwendet, siehe Kapitel 3.1.2. Die Population wurde gleich verteilt innerhalb der Werte [0, 10] für die (γ, σ)-Werte von NWP beziehungsweise [0, 1] für NJP initialisiert.

12.3.6 Zielfunktion

Da auch hier, wie schon in den vorherigen Entwicklungen, die Verbesserungen für die einzelnen Nutzer einer Site im Vordergrund stehen, wird hier ebenfalls die AWRT_k für jede Site als Zielfunktion angenommen, siehe Kapitel 7.4. Für die weiteren Bewertungen werden dann die Standardgrößen Auslastung (U_k), beziehungsweise ΔSA_k, und der Vollständigkeit halber auch $C_{max,k}$ angegeben.
Die strukturellen Analysen in Kapitel 10 haben gezeigt, dass die Erzeugung maximaler AWRT-Verbesserungen für jede Site individuell dadurch erschwert wird, dass die Sites und ihr entsprechendes Lastverhältnis stark gekoppelt sind und dieses Verhältnis im Allgemeinen sehr komplex ist. Darum ist es mit einem einkriteriellen evolutionären Algorithmus

kaum möglich, für alle Sites robuste Verbesserungen zu erzielen und zu stabilen Zuständen im Grid zu kommen. Deshalb wird hier durch ein besonderes Lernkonzept, aufbauend auf einer Ausgangsregelbasis mit anschließender partnerabhängiger Auswahl von Regeln, ein robustes Grid-Scheduling erreicht.
Vor der Beschreibung des eigentlichen Lernverfahrens zur Erzeugung von robusten Entscheidern werden noch die im weiteren Teil dieser Arbeit verwendeten und für die Untersuchung der Robustheit modifizierten Arbeitslastdaten vorgestellt.

12.4 Erlernen von Ausgangsregelbasen

Als Ausgangspunkt wird eine initiale Regelbasis für die Erstellung des Entscheiders gelernt. Die weitere Entwicklung zielt nun darauf ab, Mechanismen dafür zu schaffen, dass Ausgangsregelbasen in sich verändernden Grid-Szenarien dynamisch angepasst werden können. Dadurch wird das System in die Lage versetzt flexibel auf unterschiedliche Partner im Grid reagieren zu können.
Obwohl für die Entscheidungen sowohl Lokations- als auch Transferpolitik berücksichtigt werden muss, wird hier die Lokationspolitik zunächst vernachlässigt und ein rein partnerbezogenes Training ($K = 2$) mit alleiniger Konzentration auf die Transferpolitik durchgeführt. Dies hat den Vorteil, dass der Einfluss weiterer Partner und damit auch eine mögliche Lokationspolitik zunächst vollständig ausgeblendet wird. Weiterhin wird immer jeweils der Entscheider (das Fuzzy-System) nur *einer* Site gelernt, während die Partner-Site eine statische Greedy-Austauschstrategie verwendet. In diesem Beispiel wird dazu die bereits in Kapitel 11.3 vorgestellte AWF-Strategie verwendet. Bei der Auswahl der Paarungen ist es wichtig, möglichst alle Typen von Sites (kleine Installation, mittlere Installation, großes Rechenzentrum) zu berücksichtigen. Da hier auch alle Kombinationen der Partner gelernt werden, sind alle Sites in der Lage, mit jeweils allen anderen Sites zu interagieren. Ziel ist somit die Erstellung eines Pools von bekannten Regelbasen.
Die Nutzung einer statischen Transferpolitik für den nicht trainierten Partner ist bei diesem Verfahren deshalb notwendig, da ansonsten der Einsatz der Evolutionsstrategie erschwert würde: Bei einer wie hier verwendeten klassischen Evolutionsstrategie ist eine effektive Anpassung an den Suchraum nur dann möglich, wenn sich dieser nicht verändert. Mit anderen Worten muss eine einmal gefundene gute Regelbasis sich auch bei der nächsten Auswertung wieder als gut erweisen können. Hat sich aber die Gegenseite ebenfalls verändert und ihr Verhalten angepasst, so würde eine einmal gefundene gute Regelbasis unter diesen veränderten Umständen möglicherweise zu schlechten Resultaten führen. Dies würde den evolutionären Suchprozess nicht mehr verlässlich machen und muss deshalb vermieden werden.
Alternativ wäre es natürlich auch denkbar, das gesamte Grid gleichzeitig mit allen Sites und allen entsprechenden Regelbasen zu kodieren und dann zu optimieren. Das würde allerdings einen riesigen Suchraum zur Folge haben und außerdem nicht zu generisch verwendbaren Ausgangsregeln führen, sondern nur ein ganz bestimmtes Grid-Szenario optimieren. Ein Verfahren, das aber genau auf eine gegenseitige Adaption abzielt und dazu auch auf einem möglichst kleinen Suchraum operiert, aber eine komplett andere Lernmethode nutzt, wird im folgenden Kapitel beschrieben. Hier wird die geforderte Robustheit gegen die sich verändernde Umwelt durch zusätzliche Lernschritte erreicht und dabei zunächst eine klassische Anpassung der Regelbasen an eine sich statisch verhaltende Partner-Site versucht.

Für das eigentliche Lernen der Regelbasen mittels eines evolutionären Fuzzy-Systems werden im Folgenden die fünf Monate umfassenden Datensätze herangezogen, während die sechs Monate umfassenden Datensätze für die Anwendung verwendet werden, siehe Kapitel 7.6. Dabei sei noch bemerkt, dass natürlich all diese Datensätze nur Informationen über die Jobeinreichungen beinhalten und keine weiteren Daten über Fuzzy-Systeme (im Sinne einer Wissensbasis) daraus verfügbar sind.

12.4.1 Ergebnisse für die Trainingssequenzen

Die Ergebnisse für das paarweise Training (drei Paarungen in insgesamt sechs Szenarien) sind in Tabelle 12.1 angegeben, wobei die grau unterlegten Zeilen jeweils die trainierte Site und die weißen Zeilen die statische AWF-Site kennzeichnen. Wie erwartet, führt in allen

Szenario	Site	$AWRT_k$	U_k	$\Delta AWRT_k$	ΔU_k	ΔSA_k	$\Delta C_{max,k}$
I	KTH-5	66.100,77	35,33	86,47	-45,51	-48,56	5,60
	CTC-5	63.534,29	71,40	-9,74	12,01	12,15	-0,14
II	KTH-5	62.884,33	73,00	87,12	12,58	6,40	5,49
	CTC-5	54.745,53	62,70	5,44	-1,64	-1,60	-0,05
III	KTH-5	59.409,60	45,15	87,84	-30,36	-34,04	5,28
	SDSC05-5	58.799,15	47,34	-3,29	3,06	3,04	0,01
IV	KTH-5	70.561,62	53,39	85,55	-17,66	-21,75	4,97
	SDSC05-5	54.773,39	46,85	3,78	1,98	1,94	0,02
V	CTC-5	45.997,73	58,55	20,55	-8,14	-8,15	0,01
	SDSC05-5	57.013,44	47,27	-0,16	2,90	2,91	-0,02
VI	CTC-5	57.069,59	63,30	1,43	-0,69	-0,51	-0,20
	SDSC05-5	49.916,01	46,04	12,31	0,22	0,18	0,02

Tabelle 12.1: Ergebnisse des paarweisen Regeltrainings. Die grau unterlegten Reihen markieren jeweils die optimierte Regelbasis. Die $AWRT_k$ ist jeweils in Sekunden und alle übrigen Bewertungsfunktionen und Vergleichswerte sind in % angegeben.

untersuchten Szenarien die Anpassung der Fuzzy-Systeme durch die Evolutionsstrategie zu deutlich verkürzten AWRT-Werten. Dies resultiert dabei oftmals in auffallend größeren AWRT-Werten für die jeweils nicht adaptierte Site. Beispielsweise verbessert sich für die optimierte KTH-Site in Szenario I die AWRT um 86,47 % im Vergleich zu lokaler FCFS-Ausführung, aber für die CTC-Site geht dies mit einer Verschlechterung von nahezu 10 % einher. Es ist bemerkenswert, dass dies mit einer starken Verschiebung der Arbeitslast erreicht wird, da das Ressourcenprodukt (ΔSA) bei KTH um 48,56 % abnimmt, während es auf der CTC-Site um nahezu 12 % zunimmt. Wenn allerdings der Fokus zur Optimierung zum CTC hin verändert wird, siehe Szenario II, ist es auch für diese Site möglich, die AWRT um 5,44 % zu verkürzen. Dies geht mit einer geringen Arbeitslasterleichterung auf der CTC-Site einher. Überraschenderweise ist allerdings die AWRT für die KTH-Site immer noch deutlich verbessert, obwohl dies nicht das Ziel der Optimierung war. Dies ist zum einen in der schlechten Effizienz des FCFS-Schedulers bei rein lokaler Ausführung begründet, was die Teilnahme an einem Grid in jedem Fall begünstigt. Zum anderen zeigt es auch, dass die Kopplung der beiden Systeme durchaus komplex ist und eine Verbesserung auf einer Site nicht in jedem Fall mit einer proportionalen Verschlechterung bei der anderen einhergeht. Weiterhin zeigt dies aber auch die Grenzen der Evolutionsstrategie auf, der es offenbar nicht gelungen ist, dieses Ergebnis auch für Szenario I zu erzielen; hier wurde allerdings aufgrund

der langen Simulationszeiten immer nur ein Lauf der Optimierung durchgeführt, weshalb bei anderen Läufen bessere Lösungen nicht auszuschließen sind.

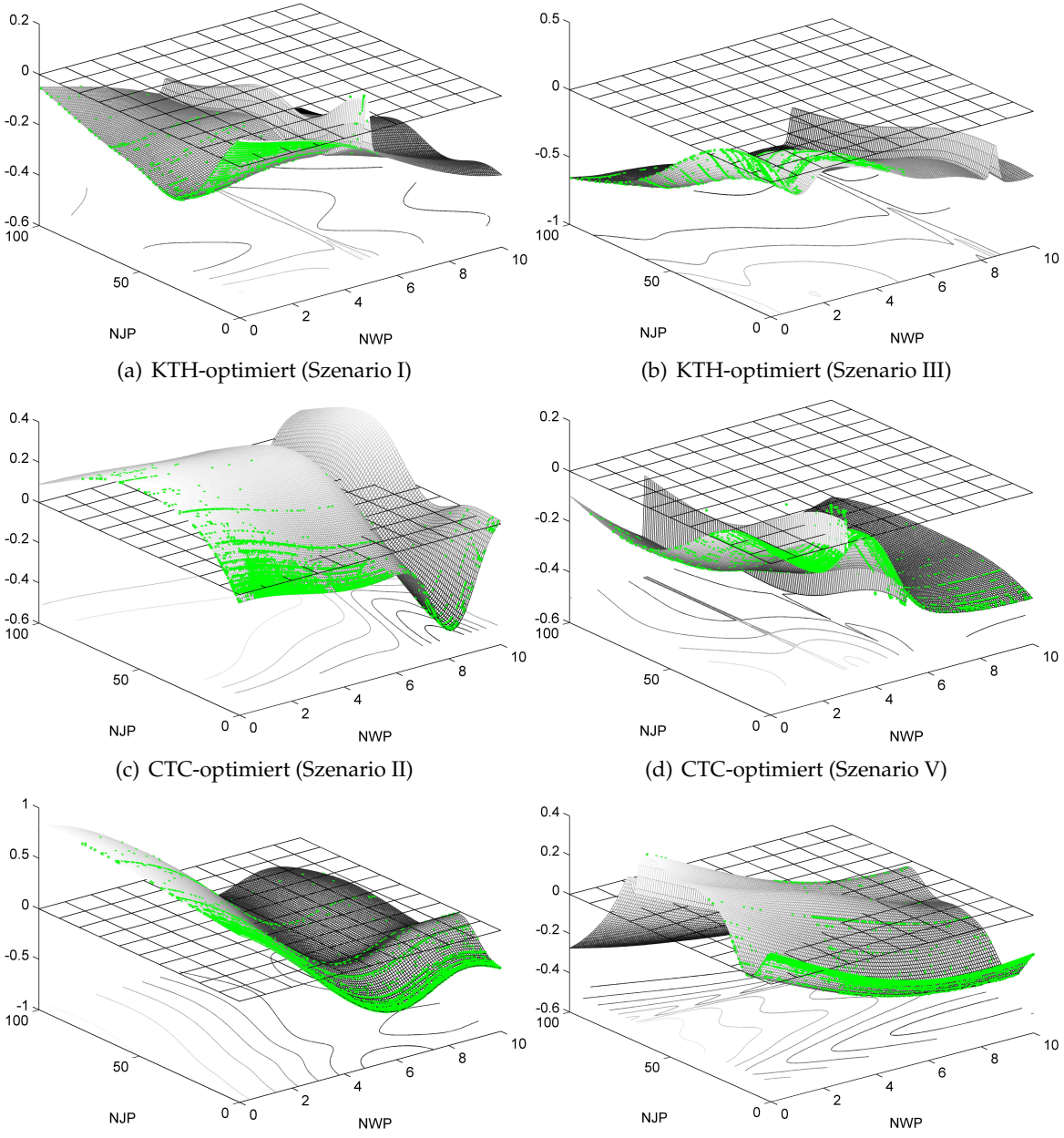
In den Szenarien III und IV interagiert jeweils die kleine KTH-Site mit der deutlich größeren SDSC05-Site und, wie erwartet, profitiert die kleine Site sehr stark von den mehr zur Verfügung stehenden Ressourcen. Es ist aber wieder bemerkenswert, dass durch die Anpassung des Entscheiders an der SDSC05-Site auch AWRT-Verbesserungen von nahezu 4 % erzielt werden. Zur gleichen Zeit erhöht sich in Szenario IV leicht die Auslastung, was beweist, dass eine Verbesserung der AWRT nicht notwendigerweise mit einer Verkleinerung der Auslastung einhergehen muss, sondern auch durch die geschickte Aushandlung von Jobs entstehen kann. Deshalb ist auch die Einbeziehung einer auf den Job bezogenen Zustandskenngröße unbedingt notwendig. Wenn, wie in Szenario V untersucht, CTC mit einem großen Rechenzentrum kooperiert, kann dort die AWRT um mehr als 20 % verbessert werden. Durch entsprechende Anpassung der Entscheider profitiert auch hier wieder die SDSC05-Site von der Kooperation.

Die vorgestellte Konzeption einer anpassbaren Entscheidungsstrategie für Grid-Scheduler ermöglicht es, auf Basis gegebener Trainingsdaten die Jobverteilung und damit auch die AWRT zwischen den Sites dynamisch und vollständig dezentral zu steuern. Dabei werden keine weiteren Informationen zwischen den Sites ausgetauscht. Um nun das eigentliche Verhalten der Entscheider nach der Optimierung besser untersuchen zu können, sind in Abbildung 12.4 die Kennlinienfelder, also die Ergebnisse aus der Überlagerung der Einzelregeln einer Regelbasis, der gelernten Entscheider angegeben. In dieser Abbildung sind die jeweiligen lokalen Zustandskenngrößen, also NJP und NWP, auf der x- beziehungsweise y-Achse aufgetragen. Das Gitter am Nullpunkt der z-Achse markiert die Grenze zwischen der Entscheidung *Annahme* (> 0) und *Ablehnung* (≤ 0) eines Jobs. Weiterhin sind die während der Anwendung aktivierten Bereiche der Fuzzy-Systeme durch grüne Punkte hervorgehoben.

Die in den Abbildungen 12.4(a) und 12.4(b) dargestellten Kennlinien müssen so interpretiert werden, dass dort ein sehr restriktives Austauschverhalten für die KTH-Site umgesetzt wird, da nur sehr kleine Jobs überhaupt angenommen und fast alle entfernten Jobs abgelehnt werden. Weil aber lokale und entfernte Jobs auf GRMS-Ebene gleich behandelt werden, siehe Kapitel 11.2, führt ein derartiges Verhalten dazu, dass zunächst alle Jobs dem anderen Partner angeboten werden. Dabei werden dann nur die kleinen Jobs lokal akzeptiert. Dies wird auch bei Betrachtung der wirklich aktivierten Bereiche deutlich. Wie Abbildung 12.4(b) zeigt, werden bei Beteiligung der großen SDSC05-Site auch die großen Jobs dem KTH zur Abgabe angeboten, da beinahe der gesamte Bereich der NJP-Kenngröße aktiviert ist. Allerdings werden diese nicht vom KTH akzeptiert und nur die kleinen Jobs zugelassen. Diese gelernte Beschränkung auf den Austausch von primär kleinen Jobs verhindert eine Überlastung der KTH-Site und ermöglicht gleichzeitig die Delegation einer großen Menge von lokal auf der KTH-Site eingereichten Jobs.

Die Situation stellt sich hingegen anders für die CTC-Site dar, siehe Abbildung 12.4(c), da diese nicht nur die eigenen großen Jobs annimmt, sondern auch externe Jobs mit großen Ressourcenanforderungen. Im Gegensatz dazu werden aber kleine Jobs zunächst lokal abgewiesen und der entfernten Site angeboten.

Schließlich zeigt sich für ein großes Rechenzentrum in den Abbildungen 12.4(e) und 12.4(f), dass der Entscheider jeden Job unabhängig von seinem Parallelitätsgrad annimmt, solange die lokale Warteschlange nicht zu voll ist. Wenn der NWP-Wert über vier steigt, schaltet die Transferpolitik von „Annahme“ auf „Ablehnung“ um. Dies erscheint sinnvoll, da so auch



(a) KTH-optimiert (Szenario I)

(b) KTH-optimiert (Szenario III)

(c) CTC-optimiert (Szenario II)

(d) CTC-optimiert (Szenario V)

(e) SDSC05-optimiert (Szenario IV)

(f) SDSC05-optimiert (Szenario VI)

Abbildung 12.4: Kennlinienfelder der Fuzzy-Systeme nach Optimierung der unterschiedlichen Paarungen. Die Systemzustandsbeschreibung findet sich auf der x- und y-Achse als NJP beziehungsweise NWP. Das Gitter am Nullpunkt der z-Achse $(y_D(\vec{x}))$ markiert die Grenze zwischen der Entscheidung *Annahme* (> 0) oder *Ablehnung* (≤ 0) eines Jobs. Die während des Betriebs aktivierten Bereiche sind durch grüne Punkte gekennzeichnet.

selbst ein großes Rechenzentrum vor der Überlastung mit externen Jobs geschützt wird und damit die Stabilität innerhalb des Grid erhalten bleibt.

12.4.2 Robustheit der erlernten Ausgangsregelbasen

In diesem Abschnitt werden nun die Ergebnisse aus zwei eigentlich unabhängigen Schritten zusammengefasst besprochen. Zunächst werden die jeweils gegen einen statischen Partner gelernten Regelbasen in Paarungen gegeneinander verwendet. Es kommt also keine AWF-Strategie zum Einsatz, sondern nur die jeweils optimierten Entscheider. Weiterhin wird direkt die Robustheit der egoistisch gelernten Regelbasen überprüft, indem diese auf die verbleibenden sechs Monate der ursprünglichen Arbeitslastaufzeichnungen angewendet werden. In Abbildung 12.5 sind dazu die Veränderungen der AWRT und der SA aufgezeichnet,

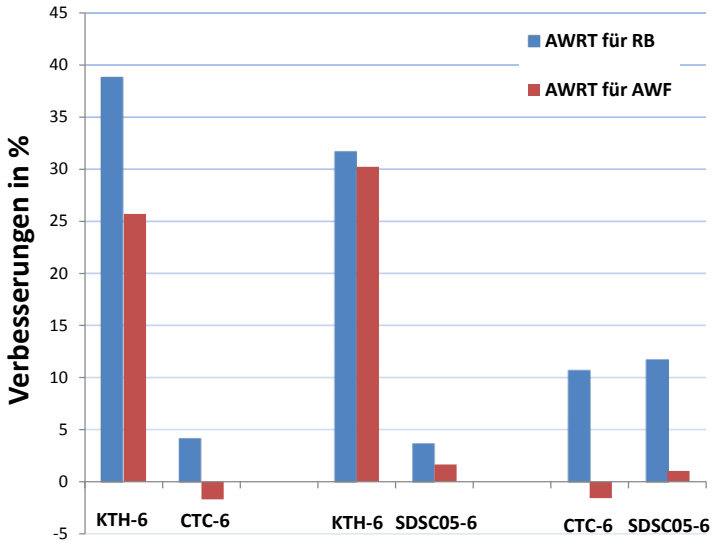


Abbildung 12.5: Verbesserungen in Antwortzeit (AWRT) und Auslastung (SA) der optimierten Regeln bei Anwendung auf nicht im Training enthaltene Jobeinreichungen.

wobei jeder Partner seine jeweils gegen den anderen Partner gelernte Regelbasis einsetzt; die Werte sind wie immer im Vergleich zu rein lokaler Ausführung mit FCFS angegeben. Offenbar werden durch die evolutionären Fuzzy-Systeme auch bei unbekannten Einreichungen deutliche Verbesserungen erzielt, was die große Robustheit der Verfahren unterstreicht. Es wurden also nicht nur Speziallösungen gelernt, sondern allgemein auf die jeweilige Sitegröße angepasste sinnvolle Entscheidungsregeln umgesetzt.
Weiterhin werden in Abbildung 12.6 die AWRT-Verbesserungen im Vergleich zur reinen Anwendung der AWF-Strategie gezeigt. Dies ist notwendig, um sie nicht nur mit lokaler Ausführung, sondern auch mit alternativen Austauschstrategien zu vergleichen und den Mehraufwand durch die Optimierung zu rechtfertigen. Da AWF die einzige entwickelte statische Strategie ist, die unter den geforderten starken Informationsbeschränkungen einsetzbar ist, wird sie als Greedy-Vergleichsstrategie genutzt, siehe auch Kapitel 11.3. Obwohl AWF gute Ergebnisse für KTH und leichte Verbesserungen für SDSC05 liefert, versagt sie bei CTC vollständig. Dies allein gibt eine Motivation für die Weiterentwicklung der Entscheidungs-

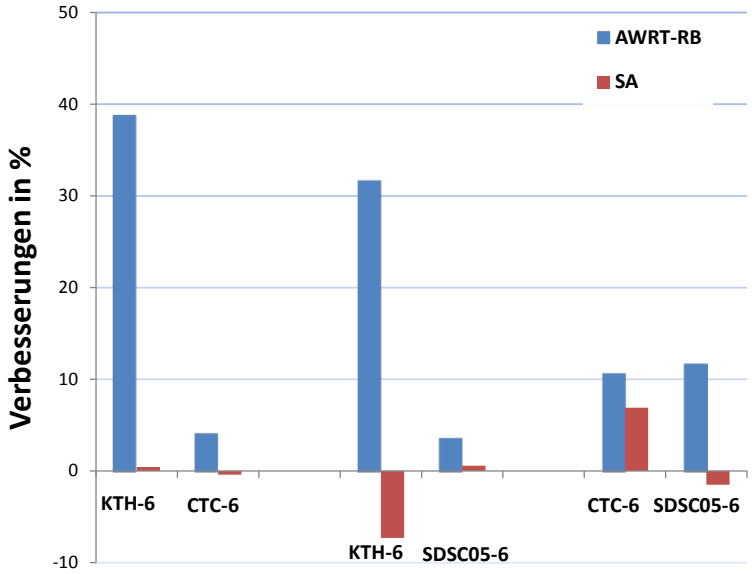


Abbildung 12.6: Verbesserungen in Antwortzeit (AWRT) und Auslastung (SA) der regelbasierten Strategie (RB) und AWF als Vergleichsstrategie in Anwendung auf nicht im Training enthaltene Jobeinreichungen.

strategien. Diese regelbasierten Strategien liefern in allen Fällen deutlich bessere Ergebnisse als AWF und erzeugen sogar deutlich kürzere AWRT-Werte für die Paarung von CTC und SDSC05. Dies ist umso motivierender, weil auch hier kein zusätzliches Training und keine Anpassungen der Fuzzy-Systeme vorgenommen wurden und somit ebenso wenig Vorwissen wie für die Anwendung von AWF benötigt wird.

12.5 Verhalten in größeren Grids

Nachdem eine Menge von möglichst allgemein verwendbaren Fuzzy-Systemen für unterschiedliche Sites evolutionär optimiert wurde, konnte schon die Robustheit der Regeln gegenüber neuen Jobeinreichungen nachgewiesen werden. Im Folgenden werden nun die erzeugten Fuzzy-Systeme auch auf ihre Einsetzbarkeit in einem größeren Grid untersucht. Dazu werden wiederum die unbekannten Jobeinreichungen (sechs Monate) der KTH-, CTC- und SDSC05-Sites verwendet und zu einem gemeinsamen Grid verbunden. Diesmal ist allerdings die Einführung einer Lokationspolitik notwendig $(K > 2)$, um eine Priorisierung der Partner bei Jobabgabe zu erstellen.

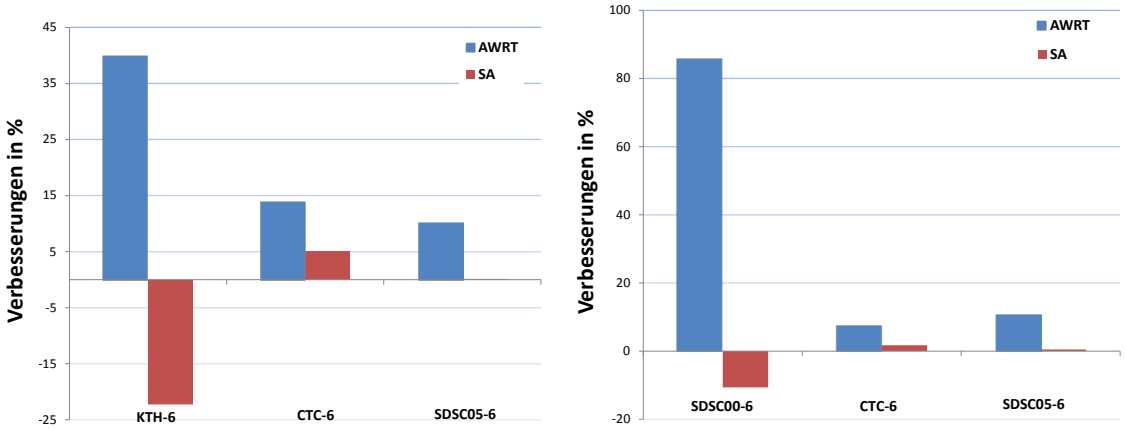
12.5.1 Eine AWRT-basierte Lokationspolitik

Um diese gewünschte Priorisierung möglicher Delegationsziele zu erstellen, wird eine zweischrittige Strategie verfolgt. Im ersten Schritt wird eine Teilmenge der Sites erzeugt, die überhaupt genug Ressourcen für die Ausführung der zu entscheidenden Jobs zur Verfügung haben. Es werden also alle Sites mit $m_k < m_j \; \forall \; k \in K$ aussortiert.
Im zweiten Schritt wird dann die so erzeugte Teilmenge entsprechend ihrer früher abgelieferten Leistung in Bezug auf den anfragenden Partner sortiert. Die Qualität der Leistung

gegenüber einer anfragenden Site wird durch die jeweiligen AWRT-Werte über alle von der zu bewertenden Site angenommenen Jobs ausgedrückt. Es werden also von jeder Site für alle anderen Grid-Teilnehmer jeweils die AWRT-Werte der abgegebenen Jobs bestimmt. Je kürzer diese Werte sind, umso schneller wurden die abgegebenen Jobs in der Vergangenheit von der entsprechenden Site bearbeitet. Es wird davon ausgegangen, dass diese Site dann auch in der Zukunft die Jobs schnell bearbeiten kann, wodurch sie dann für eine Jobdelegation hoch priorisiert wird.

12.5.2 Ergebnisse für größere Grids

In Abbildung 12.7(a) werden die entsprechenden Simulationsergebnisse unter Einsatz der vorgestellten Lokationspolitik und der Anwendung auf unbekannte Jobeinreichungen als AWRT-Verbesserungen und SA-Veränderungen gezeigt. Es werden in allen Fällen deutlich verbesserte AWRT-Werte erreicht, während die Auslastung auf den kleinen Sites teilweise stark abnimmt. Obwohl CTC und SDSC05 stärker ausgelastet sind, wird ebenfalls ihre jeweilige AWRT verkürzt, was in der Jobselektion durch den Entscheider begründet liegt.



(a) Mehrere Partner (b) Unbekannter Partner

Abbildung 12.7: Verbesserungen in Antwortzeit (AWRT) und Auslastung (SA) für Grids mit drei Sites und Anwendung auf nicht im Training enthaltene Jobeinreichungen.

12.6 Verhalten gegenüber veränderter LRMS-Strategie

Um einen weiteren Aspekt der Robustheit und Anwendbarkeit in unbekannten Szenarien darzustellen, wird die Abhängigkeit von der im LRMS eingesetzten Strategie untersucht. Bisher wurde das gesamte Verfahren immer mit FCFS als LRMS-Strategie angewendet, weil im Grid keine Laufzeitschätzungen angenommen werden. Da die verwendeten Arbeitslastaufzeichnungen aber diese Schätzungen enthalten,[19] ist hier auch ein Vergleich mit EASY als LRMS-Strategie möglich. Dazu wurden die vorher optimierten Regelbasen auf die nicht

[19]Die Aufzeichnungen stammen von Parallelrechnern und nicht aus Grid-Umgebungen, weshalb hier trotzdem auf Laufzeitschätzungen zurückgegriffen werden kann.

im Training enthaltenen Daten angewendet; dabei wurde EASY als LRMS-Strategie verwendet. Es ist wichtig zu bemerken, dass während des Trainings dabei weder diese Jobs, noch die LRMS-Strategie verwendet wurden. In Abbildung 12.8 sind die Ergebnisse für AWRT und SA im Vergleich zu rein lokaler Ausführung mit EASY dargestellt. Die Ergebnisse zei-

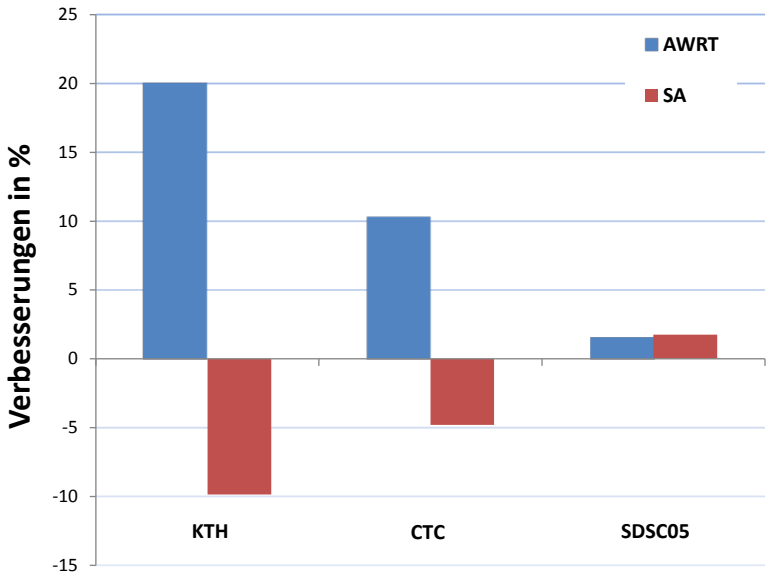


Abbildung 12.8: Verbesserung von AWRT und SA in Prozent bei Anwendung der trainierten Regelbasen und Einsatz von EASY als LRMS-Strategie.

gen, dass dieses Fuzzy-basierte Austauschverfahren auch gegenüber der LRMS-Strategie eine Robustheit aufweist. EASY-Backfilling wurde als effizienteste lokale Schedulingheuristik für die hier verwendeten Daten identifiziert, siehe Kapitel 7.5, weshalb sich die AWRT hier wie erwartet nicht verschlechtert. Eine gewisse Kopplung zwischen GRMS und LRMS ist dabei aber nie vollständig aufhebbar. Deshalb sind diese Ergebnisse eine Rechtfertigung nicht nur für den Einsatz des in dieser Arbeit entwickelten Grid-Schedulingverfahrens, sondern auch für Grid-Computing im Allgemeinen. Offensichtlich ist es möglich, mit diesen Verfahren deutlich bessere Antwortzeiten zu erzeugen, als wenn rein lokal gerechnet wird. Dabei sind die hier im LRMS benötigten Laufzeitschätzungen auf GRMS-Ebene gar nicht genutzt worden. Es ist also immer noch möglich (unter der Annahme generell verfügbarer Laufzeitschätzungen), das hier entwickelte Entscheidermodell noch weiter zu verfeinern, indem eventuell vorhandene Laufzeitschätzungen auch auf GRMS-Ebene mit einbezogen werden. Dies würde aber eine erneute Optimierung der Entscheider verlangen und ist deshalb im Rahmen dieser Arbeit nicht weiter umgesetzt worden.

12.7 Verhalten bei unbekannten Grid-Sites

Bis hierher ist der beschriebene Lernprozess dazu geeignet, eine Menge von Regelbasen zum Umgang mit bereits bekannten Partnern zu erzeugen. Dies setzt aber genaues Vorwissen

über eine gewisse Menge der auf diesen Sites eingereichten Jobs voraus, auf die dann die Transferpolitik angepasst wird. Obwohl eine Robustheit der so erzeugten Regelbasen gegenüber unbekannte aber ähnlichen Jobeinreichungen nachgewiesen wurde, ist dieser Ansatz noch nicht dazu geeignet, mit vorher vollkommen unbekannten Grid-Partnern zu interagieren. Dafür wird ein Verfahren für die automatische Anpassung an neue Grid-Szenarien benötigt, die vorher während des Trainings nicht berücksichtigt wurden. Es wird sich hier also bewusst auf die Anpassung des Szenarios während des Betriebes (Online-Fall) bezogen, bei dem noch keine Daten über die möglichen Jobeinreichungen auf der neuen Site verfügbar sind. Es ist allerdings möglich, nach einer gewissen Zeit auch eine speziell angepasste Regelbasis für einen neuen Partner mittels evolutionärer Algorithmen in den Pool von Regelbasen aufzunehmen. Dies ist aber nur dann sinnvoll, wenn sich dieser Partner auch längerfristig am Grid beteiligt.
Ein unmittelbar einsetzbares Verfahren, das auf der geschickten Auswahl von bereits vorhandenen Regelbasen für den Einsatz von unbekannten Partnern basiert, wird nun hier im Folgenden untersucht.

12.7.1 Auswahl von Regelbasen für unbekannte Grid-Sites

Wie schon in Kapitel 11.2 beschrieben, wird die regelbasierte Transferpolitik einzeln für jeden Partner angewendet. Wenn nun ein neuer Partner in den Grid-Verbund eintritt und als potenzielles Delegationsziel zur Verfügung steht, muss aus dem Pool bereits optimierter Regelbasen eine Auswahl für die Interaktion mit der neuen Site getätigt werden. Um die am besten passende Regelbasis zu finden, wird eine Beziehung zwischen den auf einer Ziel-Site maximal verfügbaren Ressourcen (m_k) und ihrem Austauschverhalten ausgenutzt. Es wurde bereits in Kapitel 10 auf die Abhängigkeit des Verbesserungspotenzials von der Sitegröße hingewiesen. Auch die durch einen evolutionären Algorithmus angepassten Regelbasen, siehe Abbildung 12.4, modellieren je nach Sitegröße ein deutlich unterschiedliches Verhalten.
Daher werden die verschiedenen angepassten Regelbasen nach ihrer entsprechenden Sitegröße (m_k) unterschieden und die Transferpolitik nutzt nun jeweils diejenige Regelbasis, die nach der Sitegröße die beste Übereinstimmung liefert. Diese wird dann statisch für alle Entscheidungen der Transferpolitik gegenüber dem bisher unbekannten Partner verwendet.

12.7.2 Ergebnisse für unbekannte Grid-Sites

Abschließend werden für das vorgestellte Regelauswahlverfahren Evolutionsergebnisse präsentiert. Dazu wird die SDSC00-6-Site mit insgesamt $m_k = 128$ Prozessoren dem Grid-Verbund hinzugefügt. Nach Tabelle 12.1 ist diese Site niemals Teil des Regeltrainings gewesen und deshalb dem System vollkommen unbekannt. Durch das vorgestellte Auswahlverfahren wird an jeder Site für die Interaktion mit der SDSC00-Site die Regelbasis verwendet, die in Verbindung mit der KTH-Site gelernt wurde, da sie mit $m_k = 100$ ungefähr die gleiche Größe wie SDSC00 besitzt. Die SDSC00-Site verwendet dabei AWF für die Interaktion mit den anderen Grid-Teilnehmern. In Abbildung 12.7(b) sind Ergebnisse für ein Grid mit drei Sites, bestehend aus CTC-, SDSC05- und der unbekannten SDSC00-Site dargestellt, bei denen auch wieder die Anwendung auf die nicht im Training enthaltenen Jobeinreichungen untersucht wurde. Deutliche AWRT-Verbesserungen sind für alle Sites zu verzeichnen, was auch teilweise wieder, ähnlich wie beim KTH, auf die schlechte Effizienz des rein lokalen

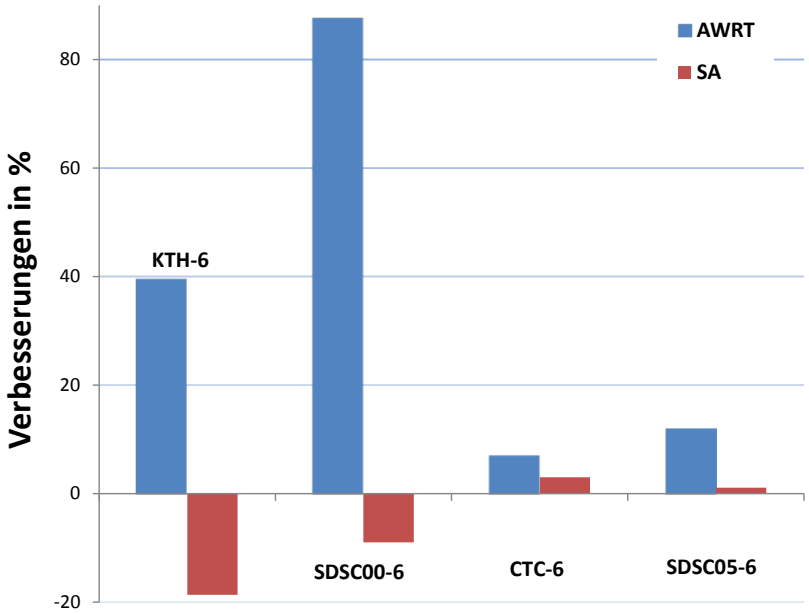


Abbildung 12.9: Verbesserungen in Antwortzeit (AWRT) und Auslastung (SA) für Grids mit drei Sites, SDSC00 als unbekannter Site und Anwendung auf nicht im Training enthaltene Jobeinreichungen.

Scheduling auf der SDSC00-Site zurückzuführen ist. Deshalb ist auch hier für die SDSC00-Site ein großes Verbesserungspotenzial gegeben. Es ist aber beachtlich, dass sich nicht nur SDSC00, sondern alle Partner um mindestens 8 % verbessern, was die große Robustheit des hier eingesetzten Verfahrens auch in unbekannten Umgebungen verdeutlicht.

Das Gleiche gilt für ein Grid aus vier Sites mit einem unbekannten Partner, wie in Abbildung 12.9 dargestellt. Es werden wiederum für alle Sites Verbesserungen um mindestens 10 % bei der AWRT erzielt, während sich die Auslastung wie gewohnt auf den kleineren Sites verringert und an den größeren erhöht. Generell ist davon auszugehen, dass sich bei steigender Zahl der teilnehmenden Sites das Problem eher vereinfacht, da zwar mehr Jobs im System zu entscheiden sind, aber auch deutlich mehr Ressourcen und damit mehr Wahlmöglichkeiten zur Verfügung stehen. In größeren Grids bestehen einfach mehr Möglichkeiten, Jobs dynamisch bei anderen Partnern unterzubringen, weshalb dort von noch größeren AWRT-Verbesserungen auszugehen ist.

Es ist also zusammenfassend festzustellen, dass durch Fuzzy-Systeme eine sehr geeignete Methode für die Umsetzung eines effizienten und robusten Jobaustauschs in dezentralen Grid-Umgebungen gefunden wurde. Dabei ist es möglich, mit einem Minimum an kommunizierten und öffentlich bereitgestellten Informationen in teilweise vollkommen unbekannten Umgebungen zu agieren. Für alle untersuchten Szenarien wurden deutlich verkürzte Antwortzeiten erreicht, wobei für große Sites Verbesserungen von 10–20 % und für kleinere Sites Verbesserungen von 40–80 % erreicht werden.

Einen essenziellen Nachteil stellt allerdings das evolutionäre Lernverfahren dar, das in diesem Fall aus mehreren Einzeloptimierungen besteht, da alle Partnerkombinationen separat

gegen statische Strategien gelernt werden müssen. Es ist mit diesem Ansatz auch nicht gelungen, *eine* Regel zu erstellen, die dann universell für alle Sites einsetzbar ist. Vielmehr ist eine deutliche Abhängigkeit der optimierten Regelbasen von der Sitegröße aufgezeigt worden, was die Übertragbarkeit von Regeln deutlich einschränkt. Weiterhin wird durch den Einsatz einer klassischen Evolutionsstrategie eine wesentliche Eigenschaft, nämlich die kooperativ-kompetitive Dynamik eines Grid-Verbunds, nicht beachtet. Diese stellt aber den Schlüssel für die Erzeugung eines möglichst gut ausbalancierten Grid-Schedulings dar. Deshalb wird im folgenden Kapitel ein mehr auf die wechselseitige Dynamik abgestimmtes co-evolutionäres Lernverfahren vorgestellt.

13

Optimierung der Entscheidungsstrategie mit co-evolutionären Algorithmen

Die Tatsache, dass in einem Grid unabhängige Partner miteinander interagieren, führt dazu, dass prinzipiell nahezu beliebige Lastverteilungen innerhalb des Grids möglich sind. Dabei ist, wie die vorherigen Untersuchungen bereits gezeigt haben, das Verbesserungspotenzial für kleine Sites sehr viel höher als für die größeren Teilnehmer-Sites. Deshalb stellt es eine besondere Herausforderung dar, einen stabilen Zustand im Hinblick auf teilnehmerübergreifende AWRT-Verbesserungen zu erzielen. Derartige Verbesserungen wurde in dem in Kapitel 12 vorgestellten Ansatz durch wiederholtes Training gegen unterschiedliche Partner und anschließende Auswahl der entsprechenden Regelbasen erreicht. Der Hauptnachteil besteht allerdings darin, dass dieses Lernen sehr aufwendig ist, weil nur jeweils eine feste Paarung betrachtet werden kann. Außerdem sind diese Verfahren nicht nativ an die Problemstellung angepasst, da sie die Interaktion der unterschiedlichen Partner nicht berücksichtigen. Vielmehr wird das Gesamtproblem durch eine externe Lernmethode nach und nach gelöst. Dies liegt in erster Linie in der traditionellen Evolutionsstrategie begründet, die für eine derartige Umgebung nur bedingt einsetzbar ist. In dem hier vorgestellten alternativen Ansatz wird das ganze Problem mit *einem* Algorithmus gelöst, ohne dass paarweise Optimierungen notwendig sind. Zu diesem Zweck wird hier ein co-evolutionärer Algorithmus eingesetzt.

Die Ursache für die oben beschriebenen Schwierigkeiten liegt darin, dass es sich bei einem Grid um ein kompetitives Marktszenario handelt. Jede Site hat ihr ganz eigenes egoistisches Interesse, die besten Ergebnisse für die jeweils von ihr betreute Nutzergemeinschaft zu erreichen. Dabei sind ihr die Ergebnisse, die an anderen Sites erzielt werden, zunächst vollkommen gleichgültig. Vorteile ergeben sich aus Sicht einer Site primär dadurch, dass viele Jobs vor allem an die großen Teilnehmer abgegeben werden, da dort eine schnelle Bearbeitung realisierbar ist und die eigenen Ressourcen geschont beziehungsweise effizienter für lokal eingereichte Jobs benutzt werden können. Dabei ist es den einzelnen Partnern nicht möglich, eine Art strategischer Wettbewerbsanalyse im Sinne einer Competitive Intelligence [122] durchzuführen, weil die dazu benötigten Informationen, siehe Kapitel 6.3.3, nicht verfügbar sind.

In einem Grid bestehen, wie mehrfach in dieser Arbeit nachgewiesen, starke wechselseitige Abhängigkeiten zwischen den Partnern. Dies führt dazu, dass eine rein egoistische Distributionspolitik, das heißt häufiges Delegieren von Jobs und permanente Ablehnung externer Jobs, Überlastungen an anderen Sites erzeugt. Auf lange Sicht verschlechtert ein derartiges Verhalten die Möglichkeit, andere Sites für die Ausführung der eigenen Jobs zu nutzen, weil dort hohe Überlast herrscht und dadurch die externe Ausführung sogar langsamer ist als

eine rein lokale Berechnung. Bei starker Ausprägung führt die rein egoistische Sicht zu einer schlechten Ausnutzung *aller* Ressourcen, weshalb insgesamt auch nur schlechtere Antwortzeiten für *alle* im Wettbewerb stehenden Sites erreicht werden. Es ist daher für alle Sites gewinnbringender, wenn diese auch externe Jobs zur Ausführung annehmen. Dies steigert nicht nur die Effizienz des gesamten Grids, sondern führt auch vor allem für die Sites selbst zu deutlichen Verbesserungen gegenüber reinem Egoismus. Obwohl sie rein egoistisch auf die maximale Verbesserung ihrer eigenen Antwortzeiten fixiert sind, müssen alle Sites lernen, von dieser starren Sicht Abstand zu nehmen. Nur so können sie nämlich die größtmögliche Verbesserung für sich selbst erreichen.

In der Literatur ist ein solches Phänomen durch das vom Novell-Gründer Raymond Noorda geprägte und von Brandenburger und Nalebuff [35] ausführlich dargestellte Kunstwort *Koopetition* beschrieben, das sich aus den beiden englischen Begriffen *competition* (Wettbewerb) und *cooperation* (Kooperation) zusammensetzt. Es bezeichnet ein Marktphänomen, bei dem eine Dualität aus Kooperation und Wettbewerb besteht und das Handeln der Marktteilnehmer beeinflusst, ohne dass diese explizit kooperieren. Das Konzept ist dabei aus der Spieltheorie entlehnt, wo es einen Gegenpart zu den im Wirtschaftsleben sonst üblichen „Nullsummenspielen“ darstellt und auf das Erreichen von Win-win Situationen abzielt.

Ein Beispiel für Koopetition ist die Open Source Strategie von Microsoft. Der Marktführer im Bereich Betriebssysteme ist der Open-Source-Business-Foundation (OSBF), einem Netzwerk europäischer Unternehmen im Bereich „Open Source“, beigetreten und gibt damit teilweise die Quelltexte eigener Produkte zur weiteren Verwendung frei. Dies erlaubt es dem Unternehmen, sich mit den Konkurrenten aus Open-Source-Initiativen besser abzustimmen, und vermeidet dadurch, in eine Außenseiterposition bei bestimmten Marktsegmenten zu geraten. Ein weiteres Beispiel ergibt sich im Bereich der Fluggesellschaften, wo konkurrierende Unternehmen wie „Continental Airlines“ und „Lufthansa“ in Teilbereichen kooperieren. So stehen diese in Konkurrenz um Kunden, Zeitfenster zum Landen an Flughäfen oder Abfluggates. Sie kooperieren allerdings durch Abstimmung eines gemeinsamen Vielfliegerprogramms, das jeweils beiden Anbietern eine größere Kundenbindung garantieren soll. Dadurch sind beide in der Lage, ihre jeweiligen Preise zu erhöhen, da die Gefahr einer Abwanderung zum Konkurrenten verringert wird. Weiterhin besteht zwischen beiden Unternehmen eine Abstimmung ihrer Zusammenarbeit mit den Flugzeugherstellern, wodurch sich die jeweiligen Entwicklungskosten für beide Unternehmen reduzieren lassen.

Viele Konzepte der Koopetition, deren genaue Ausführungen hier zu weit führen würden, sind auch auf die Beschreibung evolutionärer Prozesse übertragbar, siehe Brandenburger und Nalebuff [35]. Insbesondere aber das Problem der in Konkurrenz befindlichen Sites eines Grids weist zahlreiche Analogien zu konkurrierenden Spezies innerhalb eines gemeinsamen Ökosystems auf. Auch bei der Entwicklung unterschiedlicher Spezies zeigt sich, dass erfolgreiche und besonders schnelle Anpassung oftmals nur durch eine Kooperation zwischen Spezies möglich ist. Daher sind derartige Problemstellungen und die hier betrachtete Grid-Problematik ideal für die Lösung durch einen co-evolutionären Algorithmus. Bei einem solchen Verfahren werden genau diese gegenseitigen Abhängigkeiten ausgenutzt, um für alle Teilnehmer verbesserte Lösungen zu erzielen, siehe Kapitel 3.2. Nach dem biologischen Vorbild werden also die unterschiedlichen Sites mit ihren jeweiligen Austauschstrategien als Spezies modelliert. Das Grid stellt dann das gemeinsame Ökosystem dar, in dem nun das Spiel aus Wettbewerb und Kooperation zu verbesserten Zielfunktionswerten führen wird [2, 6].

13.1 Kompetitiv co-evolutionäres Lernverfahren

In diesem Abschnitt wird nun die Architektur eines kompetitiv co-evolutionären Algorithmus (KCA), der hier zur Lösung des bekannten Problems genutzt wird, dargestellt. Im Allgemeinen liegt die Schwierigkeit bei der Anwendung von co-evolutionären Algorithmen in der sinnvollen Aufteilung des Gesamtproblems in einzelne möglichst gleich große Teilprobleme. In dem hier geschilderten Fall ist dies allerdings idealerweise schon in der Problemstellung an sich angelegt, sodass die Anpassung des KCA sehr einfach ist. Da das angenommene dezentrale Grid-Modell von Natur aus schon aus verteilten und autarken Komponenten besteht, die in natürlicher Konkurrenz zueinander stehen, wird jede einzelne Instanz einer Grid-Site als eigene Spezies innerhalb des KCA modelliert. Dort befinden sie sich dann in Konkurrenz zu anderen Spezies und werden sich durch den evolutionären Prozess weiterentwickeln. Jede Spezies gehört dabei genau zu einer abgeschlossenen Population, sodass Reproduktionen zwischen den Spezies nicht möglich sind. Für die Realisierung der Entscheider wird genau das gleiche Verfahren in Bezug auf Fuzzy-Regelung, Zustandskenngrößen, Entscheidungsfindung, Zielfunktionen und Kodierung, wie im vorherigen Kapitel 12 beschrieben, verwendet. Es wird deshalb hier nicht weiter auf die Details zur Entscheiderarchitektur und Kodierung von Individuen eingegangen und stattdessen auf die entsprechenden Abschnitte im vorherigen Kapitel verwiesen. Die Fitness eines jeden Individuums kann aber nur durch die Interaktion mit den Individuen der anderen Spezies geschehen und findet innerhalb des Grids als gemeinsame Umwelt statt. Dazu müssen dann für jede Bewertung einzelne Grid-Paarungen zur Evaluation gebildet werden, die im Folgenden genauer beschrieben sind.

13.1.1 Bestimmung der Fitness

Die Fitnessbestimmung und der dazu benötigte Auswahlprozess[20] ist der entscheidende Teil des KCA. Um die Einzelindividuen bewerten zu können, müssen jeweils sogenannte Grid-Paarungen erzeugt werden, die jeweils einen Repräsentanten aus jeder Population enthalten. Dadurch werden die entsprechenden Individuen in Konkurrenz zueinander gestellt und ihre Fitness durch die Interaktion mit den anderen Partnern beurteilt. Nur Individuen, die bei Interaktion mit anderen Spezies gute Fitnesswerte erzielen, werden bei der lokalen Selektion in jeder Einzelpopulation weiter überleben und Nachkommen erzeugen.
Um einen hohen Wettbewerbsdruck zwischen den Individuen zu erzeugen, der dann auch zu schnellen Verbesserungen führen soll, werden deterministisch Paarungen aus den jeweils besten Individuen jeder Spezies gebildet. Dabei werden zunächst die besten Individuen aus jeder Spezies zu einer Grid-Paarung kombiniert, während die jeweils zweitbesten zu einer weiteren Grid-Paarung zusammengesetzt werden. Dies wird dann für alle $(\mu + \lambda)$ Individuen durchgeführt, siehe Abbildung 13.1. Dieses Verfahren kombiniert also jeweils die besten jeder Spezies, was nicht nur zu einem starken Wettbewerbsdruck führt, sondern auch zu möglichst fairen Wettkämpfen zwischen denselben. Nach Erreichen des Terminierungskriteriums wird dann diejenige Paarung als finale Lösung verwendet, die in der Summe der Einzelzielfunktionen das Minimum ergibt. Diese Gesamtbetrachtung wird aber nur ganz

[20] Hier sei noch einmal deutlich darauf hingewiesen, dass die *Selektion* innerhalb einer Population nach den klassischen Verfahren erfolgt und der *Auswahlprozess* zwischen den Populationen stattfindet und nur für die Bewertung benötigt wird.

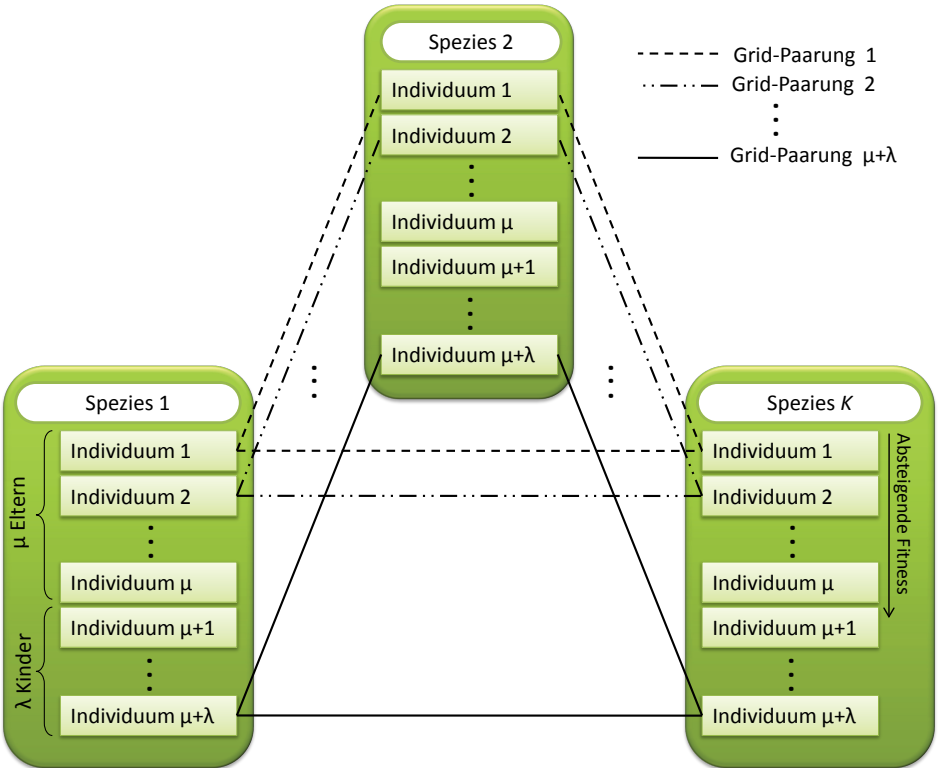


Abbildung 13.1: Erstellung von Grid-Paarungen zur Fitnessbestimmung zwischen verschiedenen konkurrierenden Spezies.

am Ende durchgeführt, sodass keine gewichtete Summe während der eigentlichen evolutionären Optimierung betrachtet wird.
Eine weitere Motivation für die so getroffene Auswahl von Grid-Paarungen im KCA liegt in der Behandlung des Egoismus: Individuen mit großer Fitness können prinzipiell ihre guten Resultate auf Kosten von anderen erzielen. Ihre Parameter entsprechen dann Regelbasen, die häufig Jobs abgeben, aber kaum Jobs von anderen Sites zur Ausführung annehmen. Wenn sich in mehreren Spezies derart eingestellte Individuen bilden, werden sie aufgrund ihrer ähnlich hohen Fitness beim nächsten Evaluationsprozess alle miteinander in den Wettbewerb treten. Bei Beharren auf egoistischer Einstellung werden diese Paarungen zu drastisch schlechteren Fitnesswerten führen. Als Konsequenz aus diesem Verhalten werden sich auf lange Sicht nur kooperative Individuen bei jeder Spezies durchsetzen. Der Schritt von der reinen Konkurrenz zu einer weitsichtigeren, auf die Gesamtproblemstellung angepassten, „koopetitiven“ Einstellung wird durch eine solche Auswahl von Grid-Paarungen explizit forciert.

13.2 Evaluation

Für die Überprüfung des oben geschilderten Konzeptes werden hier, wie schon in den vorherigen Kapiteln, drei unterschiedliche Szenarien untersucht. Dabei sind die Regelbasen zunächst wieder auf Basis von 5 Monate umfassenden Aufzeichnungen mit dem KCA trai-

niert und anschließend zur Robustheitsanalyse auf neue, insgesamt 6 Monate umfassende Einreichungen angewendet worden. Die Parametereinstellungen und die Konfiguration des Fuzzy-Systems sind identisch mit den in Kapitel 12 beschriebenen und auch der KCA wurde über 150 Generationen ausgewertet. Für die Lokation wird ebenfalls die AWRT-basierte Politik verwendet. Für die hier durchgeführten Untersuchungen wurde vor allem die bei den bisherigen Betrachtungen eher vernachlässigte SDSC03-Site mit berücksichtigt, da sie auch eine weitere große Site mit insgesamt 1.152 Ressourcen repräsentiert. Dies ist deshalb wichtig, weil es besonders schwierig ist, für große Sites deutliche Verbesserungen zu erzielen.

13.2.1 Grids mit drei teilnehmenden Sites

Zunächst wird ein kleines aus den Sites CTC mit 430, SDSC03 mit 1.152 und SDSC05 mit 1.664 Prozessoren bestehendes Grid-Szenario untersucht. Die sich daraus ergebenden AWRT-Verbesserungen sind in Abbildung 13.2 dargestellt, wobei die blauen Balken jeweils die Trainingsergebnisse und die roten Balken die Anwendungsergebnisse angeben. Es ist of-

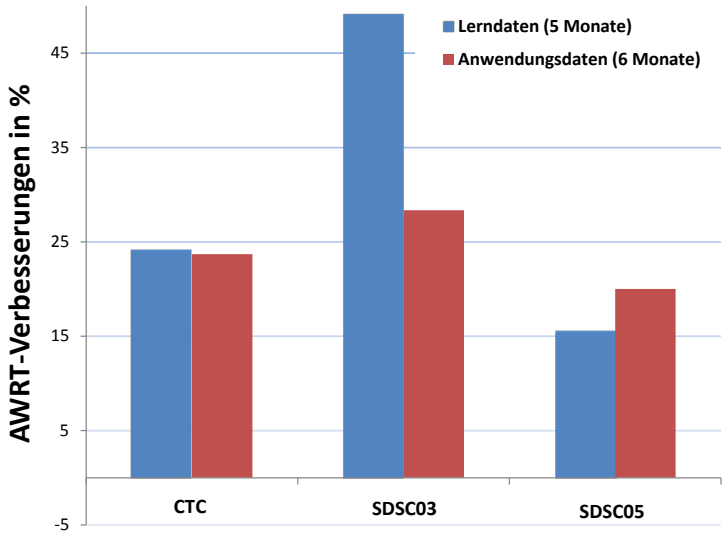


Abbildung 13.2: Trainings- und Anwendungsergebnisse der AWRT-Verbesserungen gegenüber rein lokaler Ausführung und FCFS als LRMS-Algorithmus für ein aus drei Sites bestehendes Grid-Szenario.

fensichtlich, dass auch hier wieder die AWRT für alle Sites um mindestens 10 % verkürzt wird, wobei natürlich wiederum kleineren Sites von dem Ressourcenmehrangebot profitieren. Allerdings wird durch das entsprechend angepasste Fuzzy-System vermieden, dass die großen Sites zu sehr ausgenutzt werden. Die Anwendungsergebnisse fallen erwartungsgemäß etwas schlechter aus, aber insgesamt zeigen sie eine große Robustheit gegenüber neuen Jobeinreichungen.
Wie erwartet ist in der Tat eine Zurücknahme des rein egoistischen Verhaltens zu beobachten: In Tabelle 13.1 ist dazu die Anzahl der zwischen den Sites migrierten Jobs dargestellt (siehe auch Matrix M_n in Gleichung 7.10). Die migrierte Arbeitslast (siehe Matrix M_{SA} in Gleichung 7.11) wird zusätzlich bezogen auf die jeweils lokal eingereichte Arbeitslast angegeben. Es wird also eine große Zahl von Jobs während des Grid-Betriebs ausgetauscht, so-

	M_n			M_{SA}		
Site	CTC	SDSC03	SDSC05	CTC	SDSC03	SDSC05
CTC	–	3.904	162	–	9,3	1,5
SDSC03	3.798	–	1.313	1,0	–	25,9
SDSC05	13.371	6.098	–	11,2	16,8	–

Tabelle 13.1: Anzahl der migrierten Jobs (M_n) und der migrierten Arbeitslasten (M_{SA} in %) in dem aus drei Sites bestehenden Grid-Szenario (Trainingsdaten). Die Zeilen geben die Einreichungs-Site an, während Spalten die jeweilige Ausführungs-Site spezifizieren.

dass es offensichtlich zu Kooperationen zwischen all den teilnehmenden Partnern kommt. Dabei ist es erstaunlich, dass auch die relativ kleine CTC-Site eine große Anzahl von Jobs akzeptiert und nur wenige Jobs an größere Sites abgibt. Es zeigt sich jedoch, dass die Auswahl der migrierten Jobs durch das Fuzzy-System zu sehr viel besseren Schedules und dadurch zu individuellen Gewinnen für jeden einzelnen führt.

Noch deutlicher wird dies bei Betrachtung der migrierten Arbeitslast (M_{SA} in Tabelle 13.1). Es wird offenbar eine auffallend große Menge von Jobs zwischen den beiden großen Sites ausgetauscht, sodass SDSC03 sogar nahezu ein Viertel der Gesamtarbeitslast an SDSC05 abgibt. Auf der anderen Seite akzeptiert diese Site aber auch eine Menge Arbeit von sowohl CTC als auch SDSC05. Auffallend ist weiterhin, dass sogar die kleine CTC-Site eine Menge Arbeit von der großen SDSC05-Site übernimmt.

Die konkurrierenden Sites haben sich auf eine derart sinnvolle Kooperation im Bezug auf den Austausch von Jobs einigen können, dass sich deutliche Vorteile für alle ergeben. Das Konzept der Koopetition hat im Grid eine vorteilhaft einsetzbare Anwendung gefunden.

13.2.2 Grids mit vier teilnehmenden Sites

In Abbildung 13.3 sind nun analog Ergebnisse für ein größeres Grid mit SDSC00 als zusätzlichem kleinen Partner zu sehen. Auch hier zeigen sich, wie erwartet, große Verbesserungen für alle Sites, wobei sie in der Anwendung leicht verlagert sind. Insgesamt kann aber auch hier von einem robust einsetzbaren Verfahren gesprochen werden.

Interessant ist aber hier die Abbildung 13.4, bei der wie schon im vorherigen Kapitel die Kennlinien der KCA-optimierten Fuzzy-Regler mit den entsprechend grün markierten Aktivierungsbereichen dargestellt sind. Dabei wurden alle Fuzzy-Systeme in *einem* Algorithmus gelernt, weshalb keine Angaben zu Paarungen mehr notwendig sind. Die Regelbasis in Abbildung 13.4(a) zum Beispiel wird also immer gegen alle anderen hier betrachteten Partner verwendet.

Bei genauer Betrachtung ist bei allen Regelbasen erkennbar, dass es relevante Bereiche gibt (also Bereiche, die durch grüne Punkte markiert sind und deshalb wirklichen Einfluss haben), in denen $y_D > 0$ ist. Dies bedeutet also, dass hier deutlich der Egoismus zurückgenommen wird, da genau in diesen Bereichen Jobs von anderen Sites zur Ausführung angenommen werden. In Abbildung 13.4(b) ist interessanterweise zu erkennen, dass die relativ kleine SDSC00-Site auch bei starker Auslastung (also großem NWP-Wert) sehr große Jobs annimmt, während alle anderen Sites Jobs nur bei niedriger Auslastung annehmen.

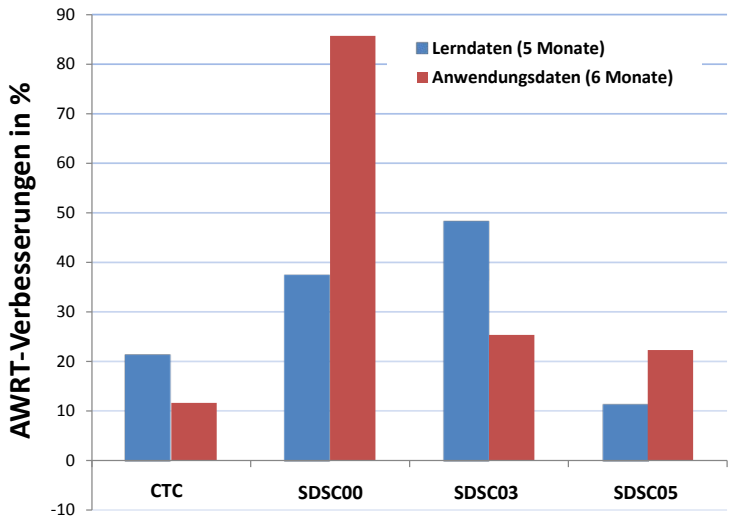


Abbildung 13.3: Trainings- und Anwendungsergebnisse der AWRT-Verbesserungen gegenüber rein lokaler Ausführung und FCFS als LRMS-Algorithmus für ein aus vier Sites bestehendes Grid-Szenario.

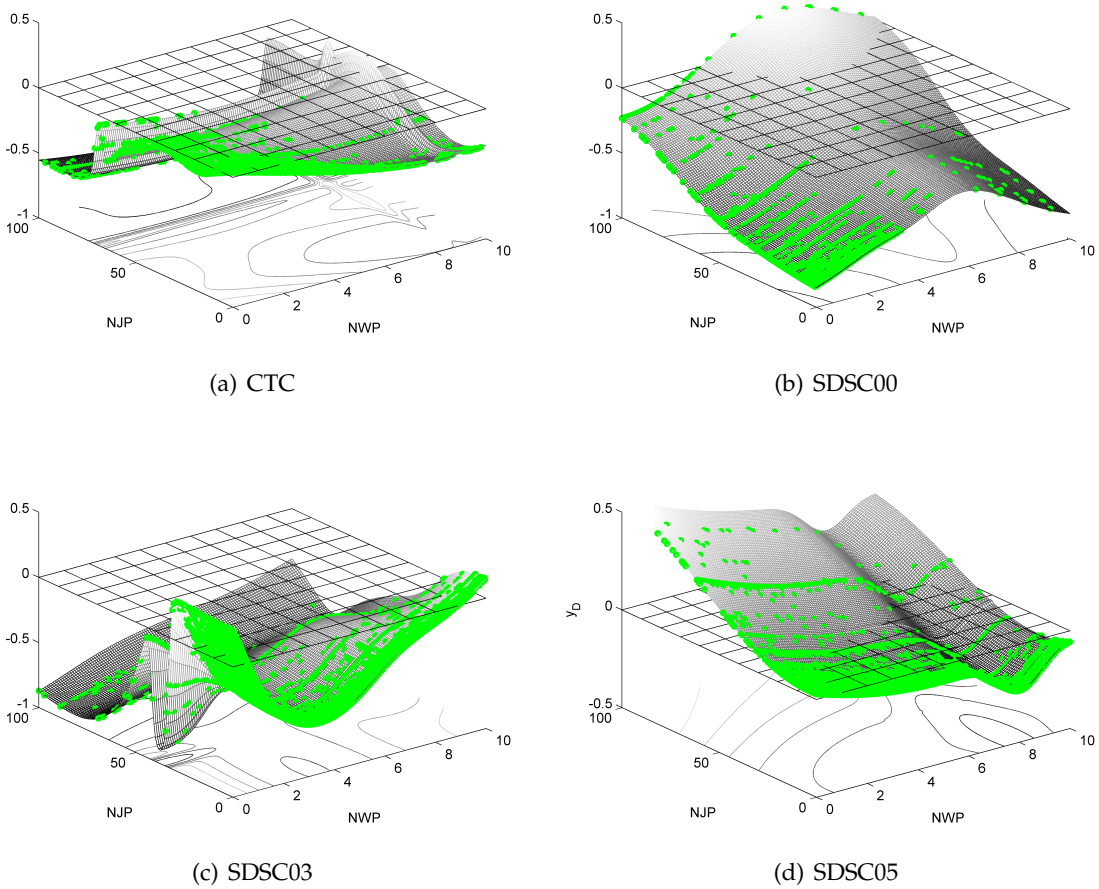
Um diese Verhalten der SDSC00-Site erklären zu können, ist wiederum die Betrachtung der Anzahl der migrierten Jobs notwendig. Es soll hier auf eine vollständige Darstellung verzichtet werden. Es stellt sich jedoch heraus, dass SDSC00 nur neun Jobs von anderen Sites akzeptiert, dafür aber über 10.000 Jobs zu anderen Sites migriert. Es ist also eine deutliche Spezialisierung zu beobachten. Die SDSC00-Site ist in der Lage, durch Kooperation mit anderen Partnern viele ihrer kleinen Jobs abzugeben und dafür einige Jobs, die ihrer maximalen Kapazität ($m_k = 128$) entsprechen, möglichst schnell auszuführen. Durch diese wechselseitige Kooperation und die Spezialisierung auf bestimmte Jobtypen sind alle Partner in der Lage, ihre Jobs deutlich schneller zu starten.

13.2.3 Grids mit fünf teilnehmenden Sites

Da der co-evolutionäre Ansatz weniger Optimierungsschritte benötigt, ist er besonders gut für die Anwendung in der Optimierung großer Grid-Szenarien geeignet. Deshalb wird hier noch ein Grid bestehend aus fünf Sites optimiert, siehe Abbildung 13.5.
Dazu wurden erstmals alle (mit Ausnahme des LANL) in dieser Arbeit verwendeten Datensätze zu einem Grid formiert und im co-evolutionären Wettstreit optimiert. Es ist wiederum ersichtlich, dass nach dem Lernen klare AWRT-Verbesserungen von mindestens 15 % und bis zu 90 % gegenüber rein lokaler Ausführung erreicht werden. Das gleiche gilt auch hier für die Übertragbarkeit der gelernten Regelbasen. Offensichtlich kann auch dieses Verfahren bei der Anwendung auf nicht im Training enthaltene Daten sehr gute Ergebnisse liefern. Weitere Ergebnisse zur Robustheit sind in Abschnitt 13.3 zu finden.

13.2.4 Verhalten gegenüber veränderter LRMS-Strategie

Genau wie im vorherigen Kapitel 12.6 wird auch für den co-evolutionären Ansatz das Verhalten bei veränderter LRMS-Strategie evaluiert. Dazu wurden wiederum die optimierten



(a) CTC

(b) SDSC00

(c) SDSC03

(d) SDSC05

Abbildung 13.4: Kennlinienfelder der Fuzzy-Systeme nach co-evolutionärer Optimierung. Die Systemzustandsbeschreibung findet sich auf der x- und y-Achse als NJP beziehungsweise NWP. Das Gitter am Nullpunkt der z-Achse $(y_D(\vec{x}))$ markiert die Grenze zwischen der Entscheidung *Annahme* (> 0) oder *Ablehnung* (≤ 0) eines Jobs. Die während des Betriebs aktivierten Bereiche sind durch grüne Punkte gekennzeichnet.

Regelbasen auf den Anwendungsdaten von sechs Monaten simuliert. Dabei wurde EASY-Backfilling statt der beim Training eingesetzten FCFS-Strategie als LRMS-Strategie verwendet. Die Verbesserungen sind in Abbildung 13.6 für AWRT und SA im Vergleich zu rein lokaler Ausführung mit EASY dargestellt.

Es zeigt sich auch hier, allerdings nicht in so eindrucksvoller Form, dass die Regelbasen robust gegenüber der LRMS-Strategie sind. Für die große Site SDSC05 ergeben sich keine AWRT-Vorteile, wohingegen die beiden kleineren Sites um bis zu 15 % profitieren. Dieses Ergebnis mag darin begründet sein, dass für dieses Szenario bei dem durchgeführten Lauf mit dem co-evolutionären Algorithmus keine guten Lösungen gefunden wurden, siehe auch Abbildung 13.7(b). Es kann aber auch hier mit einiger Sicherheit vermutet werden, dass

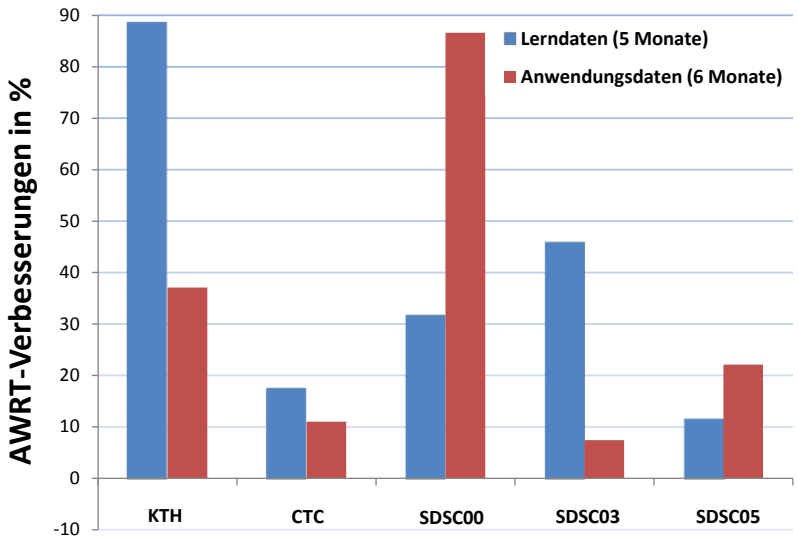


Abbildung 13.5: Trainings- und Anwendungsergebnisse der AWRT-Verbesserungen gegenüber rein lokaler Ausführung und FCFS als LRMS-Algorithmus für ein aus fünf Sites bestehendes Grid-Szenario.

sich für diesen Ansatz bei Nutzung einer effizienteren LRMS-Strategie auch Verbesserungen gegenüber rein lokaler Ausführung ergeben.
Es ist bei beiden Ansätzen festzuhalten, dass diese bereits unter sehr schwierigen Bedingungen (keine Laufzeitschätzungen und sehr ineffizientes FCFS-Scheduling) sehr gute Ergebnisse erzeugen. Sobald mehr Informationen nutzbar sind, ist davon auszugehen, dass sich zwar die lokalen Schedulingergebnisse verbessern lassen, aber Grid-Computing in der hier durchgeführten dezentralen Form dennoch deutliche Vorteile für die lokalen Nutzergruppen liefert.

13.2.5 Verhalten bei unbekannten Grid-Sites

Analog zu den Analysen in Kapitel 12.7 ist noch die Anwendbarkeit der hier optimierten Regelbasen in sich verändernden Grid-Umgebungen zu untersuchen. Dazu ist in dem optimierten Szenario bestehend aus KTH, CTC und SDSC05, das erst im nächsten Abschnitt genauer besprochen wird, die KTH-Site durch die vorher unbekannte SDSC00-Site ersetzt worden. Es wird wieder die AWRT-basierte Lokationspolitik eingesetzt und die unbekannte Site verwendet AWF als Austauschstrategie. Die Ergebnisse sind in Tabelle 13.2 zu finden, wobei zu Vergleichszwecken noch einmal die Ergebnisse aus Abbildung 12.7(b) des rein evolutionären Ansatzes aufgelistet sind.
Hier ist klar ersichtlich, dass beide Verfahren mit dem unbekannten SDSC00-6-Partner sehr gut umgehen können und das Grid trotzdem stabil bleibt. Die starken Verbesserungen beim SDSC00 lassen sich zum einen mit dem sehr ineffizienten lokalen FCFS-Scheduling ohne Grid und zum anderen mit der hohen Effizienz der dort eingesetzten AWF-Strategie erklären. Es finden zwar Verschiebungen der Verbesserungen zwischen den Sites je nach Verfahren statt, aber insgesamt sind beide auch mit unbekannten Partnern funktionsfähig.

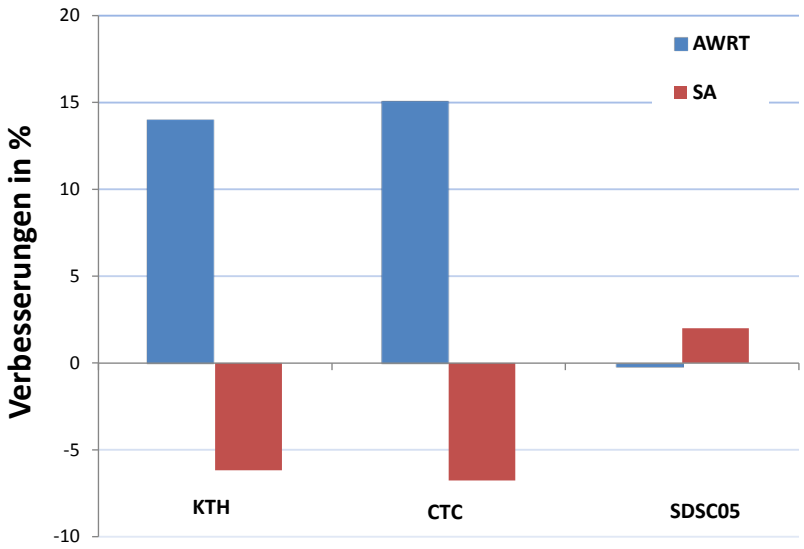


Abbildung 13.6: Verbesserung von AWRT und SA in % bei Anwendung der trainierten Regelbasen und Einsatz von EASY als LRMS-Strategie.

EA	SDSC00-6	CTC-6	SDSC05-6
	78,03 %	10,02 %	17,95 %
COEA	SDSC00-6	CTC-6	SDSC05-6
	85,54 %	7,59 %	10,82 %

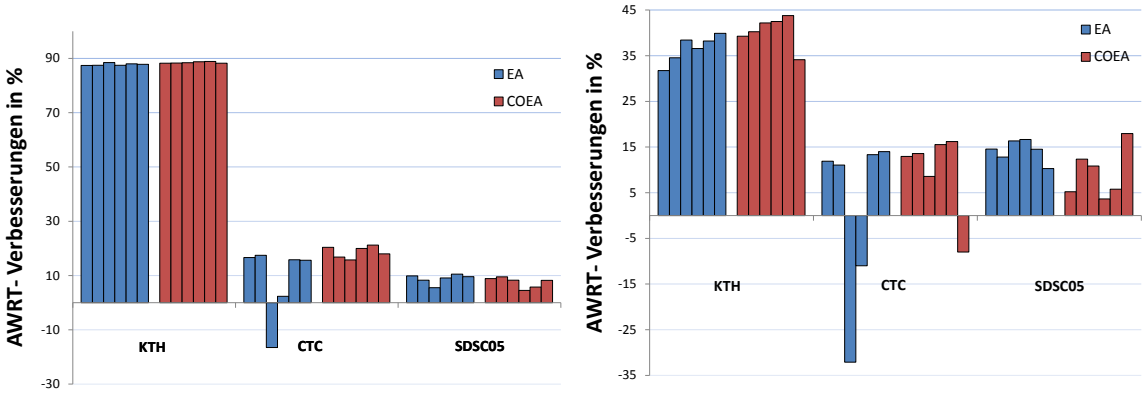
Tabelle 13.2: Verbesserung der AWRT bei einem Grid mit drei Sites und einem unbekannten Partner in Anwendung auf nicht im Training enthaltene Jobeinreichungen.

13.3 Vergleich der beiden evolutionären Optimierungsverfahren

Abschließend wird der hier vorgestellte KCA mit dem rein evolutionären Ansatz aus dem vorherigen Kapitel verglichen. Dabei ist es schwierig, einen tatsächlich haltbaren Vergleich zu liefern, da Einschränkungen zu berücksichtigen sind: So wurden alle durchgeführten Untersuchungen aufgrund der zeitlich aufwendigen Optimierung nicht, wie sonst bei randomisierten Suchverfahren üblich, über eine Vielzahl von Läufen gemittelt. Es war unter den genannten Umständen lediglich möglich, die gleichen Simulationen jeweils sechsmal bei unterschiedlichen Startpopulationen auszuführen.

Dennoch kann auf Basis der vorliegenden Daten eine Verfahrensprüfung *(proof of concept)* durchgeführt und damit die Anwendbarkeit beider Verfahren dokumentiert werden. Die Tatsache, dass in den meisten untersuchten Grids durch die evolutionäre Optimierung deutliche Verbesserungen der jeweiligen Zielfunktionen erreicht werden, lässt den Schluss zu, dass diese Verfahren nicht nur leistungsfähig, sondern auch verlässlich sind. Weiterhin sind die untersuchten Szenarien nur Testbeispiele und reale Grid-Szenarien können in der Praxis so unterschiedlich sein, dass jeder quantitative Vergleich der jeweiligen Optimierungsverfahren nur approximativen Charakter haben kann. Es sei hier dennoch erlaubt, auf Basis der

vorhandenen Simulationen einige wesentliche Gemeinsamkeiten und Unterschiede herauszuarbeiten.
Als Vergleichsbeispiel dient hier nochmals ein Szenario mit drei Sites bestehend aus KTH, CTC und SDSC05. Die Qualität der nach der Lernphase erreichten Lösungen ist, wie in Abbildung 13.7(a) für die sechs durchgeführten Läufe dargestellt, bei beiden Verfahren annähernd gleich groß. Es werden jeweils für die beiden Verfahren die sechs Realisierungen in einem Diagramm dargestellt.



(a) Lernen auf 5 Monate

(b) Anwendung auf 6 Monate

Abbildung 13.7: Unterschiede der erreichten Verbesserungen der Antwortzeit (AWRT) bei Optimierung mit klassischen evolutionären Verfahren (EA) und co-evolutionärem Verfahren (COEA). Es sind jeweils sechs Läufe für jedes Verfahren dargestellt.

Es zeigt sich hier allerdings schon, dass in zwei Läufen der rein evolutionär basierte Ansatz (EA) zu keinen Verbesserungen beziehungsweise sogar zu Verschlechterungen führt. Offenbar ist dort die evolutionäre Suche nicht erfolgreich gewesen, was aber bei anderen Initialisierungen nicht generell bestätigt werden kann. Es gilt weiterhin zu bedenken, dass bei dem EA-Verfahren jeweils nur paarweise gelernt wurde und die Anwendung der Regelbasen in einem Szenario mit drei Partnern bereits eine gewisse Übertragbarkeit der Ergebnisse voraussetzt. Deshalb ist hier auch eine größere Empfindlichkeit hinsichtlich der Ergebnisse zu erwarten.
Bei der Übertragbarkeit der gefundenen Regelbasen auf neue Arbeitslastdaten erzielt der co-evolutionäre Ansatz (COEA) deutlich bessere Ergebnisse, siehe Abbildung 13.7(b). Die durch den co-evolutionären Ansatz erzeugten Regeln liefern für den CTC bei Anwendung zwar auch einmal schlechtere AWRT-Werte als bei lokaler Ausführung, aber insgesamt präsentieren die Ergebnisse eine höhere Robustheit als bei einem reinen EA. Das intensive Wechselspiel der einzelnen Partner hat zu einer angepassten „koopetitiven“ Lösung für ein gegebenes Szenario unabhängig von den Jobeinreichungen geführt. Bei reinen EA ist insgesamt zwar eine gute Übertragbarkeit festzustellen, aber die im Training versagten Läufe zeigen auch in der Übertragbarkeit konsequenterweise ein schlechtes Verhalten. Diese Resultate lassen den Schluss zu, dass durch den Fuzzy-basierten Ansatz gut angepasste Verfahren auch eine gute Robustheit aufweisen.
Ein weiterer Vorteil des co-evolutionären Verfahrens liegt in seiner Kompaktheit, durch die der Simulationsaufwand für die Optimierung deutlich geringer ausfällt. Zwar steigt die

Komplexität des Algorithmus durch Einführung weiterer Spezies mit K, aber die Anzahl der benötigten Auswertungen bleibt konstant. Dies ist also ein deutlicher Vorteil des co-evolutionären Ansatzes, weshalb er für die schnelle Erzeugung von einsetzbaren Verfahren klar zu bevorzugen ist. Beim rein evolutionären Ansatz ist zur Optimierung von K Sites die Bildung von $(K^2 - K)$ Paarungen notwendig, was einen erheblichen Simulationsaufwand bedeutet, der nur durch eine hochgradig parallele Ausführung der Einzelsimulationen handhabbar ist.
Abschließend sei noch bemerkt, dass die Anwendung beider Algorithmen während des Grid-Betriebs keinen zusätzlichen Rechenaufwand mit sich bringt, da jeweils nur die Berechnung der Fuzzy-Entscheidung durchgeführt werden muss. Dies kann in der Regel in weniger als einer Sekunde geschehen, weshalb durch den reinen Schedulingvorgang keine weiteren Verzögerungen auftreten.

14
Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurden Strategien für den koordinierten Jobaustausch in Computational Grids mit verschiedenen Methoden der Computational Intelligence konzipiert, optimiert und auf Basis von realen Arbeitslastaufzeichnungen evaluiert. Dazu wurde zunächst ein Modell für ein dezentrales Grid-Szenario entwickelt, das aus aktuellen Realisierungen im Rahmen von nationalen und internationalen Grid-Projekten motiviert ist. Dort wurden bereits Einzellösungen für unterschiedliche, in virtuellen Organisationen zusammengefasste Communities entwickelt. Ziel war und ist es aber, im Rahmen eines Computational Grid die Zusammenarbeit zwischen Communities auf globaler Ebene durch die Unterstützung eines beiderseitigen Jobaustauschs zu ermöglichen. Communities wurden dazu in Form einzelner Parallelrechners mit einer lokalen Nutzergemeinschaft als Sites abgebildet, die dann als abgeschlossene Einheiten in einem Grid agieren. Ziel bei dem Entwurf der Schedulingstrategien war es, für die jeweilige Nutzergemeinschaft einer Site eine möglichst große Verbesserung der Antwortzeiten im Vergleich zur rein lokalen Ausführung ihrer Jobs zu erreicht.

Dazu wurden zunächst zwei einfache Strategien, einmal unter Nutzung einer zentralen Austauschplattform und einmal mittels einer aktiven Anfrage von Jobs, untersucht. Hierdurch sind erste Erkenntnisse über das Verhalten eines derartigen dezentralen Verbunds gewonnen worden, die ein hohes Potenzial für mögliche Verbesserungen der Antwortzeiten offenbaren. Allerdings sind derart einfache Austauschverfahren nur dann vorteilhaft nutzbar, wenn zwischen den Partnern intensive Informationen sowohl über die gerade lokal wartenden Jobs als auch über lokale Auslastungen kommuniziert werden.

Als weiterer Schritt wurde die genaue Dynamik eines dezentralen Grids untersucht und dazu mittels mehrkriterieller evolutionärer Optimierung die Pareto-Front der maximal erreichbaren Antwortzeiten für beispielhafte Grid-Paarungen erzeugt. Die Sites agieren in einem Grid natürlich nicht unabhängig voneinander, sondern können durch eine Verlagerung der Arbeitslast unterschiedlich gute Antwortzeiten für ihre jeweilige Nutzergruppe erzielen. Dabei ergeben sich für eine Site zu einem gewissen Grad Vorteile auf Kosten anderer Sites. Durch die offline durchgeführte Erzeugung der Pareto-Fronten wurde das Verbesserungspotenzial zwischen den Sites sichtbar gemacht. Dazu wurden a priori Partitionen der Gesamtarbeitslast erzeugt und für ihre Modifikation wurden spezielle genetische Variationsoperatoren entworfen. Diese wurden dann in NSGA-II integriert und auf Grid-Szenarien mit zwei Sites angewendet. Es ergaben sich deutliche Abhängigkeiten der erreichbaren Verbesserungen von der Größe einer Site, wobei kleine Sites mit wenig verfügbaren Ressourcen deutlich mehr von der Teilnahme am Grid profitierten. Weiterhin konnte gezeigt werden, dass die vorher untersuchten statischen Strategien zu einem Lastausgleich im Grid tendieren, was allerdings selten mit der maximal erreichbaren Verbesserung für alle teilnehmenden

Sites gegenüber lokaler Ausführung einherging. Dies gilt insbesondere, wenn kleine Sites mit großen Sites interagieren, weshalb die Verfeinerung der Austauschstrategien notwendig erschien.

Die Forderung nach einer restriktiven Informationspolitik, die nicht nur die Sicherheit sondern auch die lokale Autonomie der am Grid teilnehmenden Sites garantiert, motivierte den Entwurf einer eigenen Grid-Schedulingarchitektur. Dieser wurde in zwei Stufen umgesetzt, wobei das lokale Ressourcen-Management (LRMS) von der Grid-Schicht (GRMS) vollkommen entkoppelt ist. Das GRMS lässt sich in vorhandene Middleware-Lösungen integrieren und realisiert den Austausch von Jobs mit Sites oder reicht akzeptierte Jobs an das LRMS weiter. Im GRMS ist dabei die eigentliche Austauschstrategie umgesetzt, die zur Entscheidungsfindung lediglich auf Informationen der Jobbeschreibung und der jeweiligen lokalen Systemgrößen zurückgreifen kann. Zu Vergleichszwecken wurde eine statische Greedy-Strategie (AWF) untersucht, die aber auf Grund der starken Informationsbeschränkungen nur teilweise verlässliche Ergebnisse liefert.

Um in derart vagen Umgebungen dennoch sinnvolle Entscheidungen treffen zu können, wurde ein Fuzzy-System entworfen und mittels evolutionärer Algorithmen optimiert. Im ersten Schritt wurden dazu Regelbasen für jeweils einen festen Partner entwickelt und dann in erweiterten Grid-Umgebungen eingesetzt. Zur Überprüfung der Robustheit wurden die Lerndaten von den späteren Anwendungsdaten getrennt und einzeln evaluiert. Die erlernten Fuzzy-Systeme erreichten durchweg große Verbesserungen der Antwortzeiten an allen Sites. Als nächster Schritt wurde ein Verfahren zum Umgang mit unbekannten und vorher nicht am Lernprozess beteiligen Sites entwickelt, bei dem bereits gelernte Regelbasen für die Interaktion mit unbekannten Partnern ausgewählt wurden. Als sinnvolles Auswahlkriterium wurde dabei die Größe der jeweiligen Site identifiziert. Mit diesem Ansatz konnte gezeigt werden, dass es auch möglich ist, die erlernten Fuzzy-Systeme in unbekannten Grid-Umgebungen effektiv einzusetzen. Die Notwendigkeit zum Erlernen einer Grundmenge von verwendbaren Regelbasen stellte einen großen Berechnungsaufwand dar und lies sich für größere Grids nur schwer in vertretbarer Zeit durchzuführen. Weiterhin wurden durch diesen Ansatz die jeweils egoistischen Ziele der Sites für die Teilnahme aber nur teilweise berücksichtigt.

Deshalb wurde als letzte Untersuchung das Fuzzy-System mit einem kompetitiven co-evolutionären Algorithmus erlernt. Hierbei wurden Sites jeweils als Spezies modelliert, die in einer gemeinsamen Umwelt in Konkurrenz stehen. Die Fitness eines Individuums wurde dabei durch die für die jeweils eigene Nutzergruppe erzielte Antwortzeit bestimmt, wobei Spezies jeweils die eigenen Fitnesswerte auf Kosten der anderen verbessern konnten. Vorherige Untersuchungen hatten jedoch gezeigt, dass sich für die eigenen Nutzergruppen maximale Vorteile nur dann ergeben, wenn Sites auch bereit sind, Arbeitslast von anderen zu übernehmen, weil nur so ein sinnvoller Jobaustausch initiiert werden kann. Dabei wurde durch die Wahl der Paarungen zur Fitnessbestimmung bewusst ein „koopetitives“ Verhalten forciert, bei dem die Konkurrenten untereinander lernen zu kooperieren, um den eigenen Vorteil zu erhöhen. Die abschließenden Evaluationen zeigten, dass auch in diesem System mit geringerem Rechenaufwand robuste und effiziente Grid-Schedulingentscheidungen getroffen werden konnten, ohne dass interne Informationen global kommuniziert werden mussten. Es wurde so durch den Einsatz von Methoden der Computational Intelligence eine Win-win Situation für alle Teilnehmer erreicht.

Für die weitere Entwicklung ist zunächst die Verfeinerung des hier angenommenen Grid-Modells von entscheidender Bedeutung. Obwohl die vorgestellten Verfahren nicht auf die Anwendung in diesem Modell beschränkt bleiben, sondern auch durchaus in komplexeren Umgebungen denkbar sind, müssen doch einige zusätzliche Aspekte noch genauer berücksichtigt werden. Die Verzögerungen, die durch Bandbreitenbeschränkungen zwischen den Sites entstehen, müssen in einem adäquaten Modell integriert werden, sodass Schedulingentscheidungen auch von unterschiedlich schnellen Netzwerkverbindungen beeinflusst werden. Dies betrifft zwar nicht die Verhandlung, aber durchaus die Migration von Jobs. Von besonderer Wichtigkeit ist dies, wenn zusätzlich noch Datenabhängigkeiten der Jobs und die lokale Bereitstellung von Daten (staging) garantiert werden müssen. In diesem Zusammenhang ist natürlich auch die Berücksichtigung von Jobabhängigkeiten wünschenswert, wobei verschiedene bereits vorhandene Konzepte des Workflowschedulings in dezentrale Entscheider integriert werden können.
Obwohl, wie ausführlich bei der Wahl der Arbeitslastdaten diskutiert, Laufzeitschätzungen generell nicht im Grid verfügbar sind, könnte das Fuzzy-System natürlich auch unter Annahme von Laufzeitschätzungen untersucht werden. Dies würde nicht nur die Anwendung von EASY im LRMS erlauben, sondern auch die feinere Entscheidungsfindung auf GRMS-Ebene ermöglichen. Dazu müsste die Laufzeitschätzung aber als zusätzliche Zustandskenngröße im Regler berücksichtigt werden, was dann wieder zu einer deutlichen Vergrößerung des Suchraumes führen würde. Es bleibt zu evaluieren, inwieweit sich durch diese zusätzlichen Informationen Vorteile für die Verteilung der Jobs ergeben können.
Neben diesen modellbezogenen Erweiterungen ist auf lange Sicht aber besonders die Entwicklung eines komplett selbstlernenden Systems wichtig. Die hier vorgestellten Strategien wurden alle mittels eines offline durchgeführten Trainings angepasst. Durch zusätzliche Mechanismen war es möglich, die so erstellten Entscheidungsmechanismen zudem robust gegenüber sich verändernden Umgebungen zu machen. Da aber die zukünftigen Grid-Umgebungen als stark veränderlich angesehen werden müssen, ist davon auszugehen, dass derartige Robustheitsmechanismen nicht ausreichen werden. Außerdem verlangt dieser Ansatz immer die Verfügbarkeit gewisser Trainigsdatensätze der in der Vergangenheit in einer Community oder virtuellen Organisation eingereichten Jobs. Es ist aber eher wünschenswert, ein komplett selbstlernendes System zu entwickeln, bei dem zunächst ein naiver und heuristischer Austausch unter den schon angegebenen Informationsbeschränkungen durchgeführt wird. Durch die Bewertung von Entscheidungen sollten sich anschließend in jedem Schritt die Schedulingstrategien der jeweiligen Sites selbst in ihrem Verhalten adaptieren. Dies entspricht im Wesentlichen einem maschinellen Online-Lernverfahren, bei dem sich allerdings die große Schwierigkeit ergibt, dass einzelne Entscheidungen beim Online-Scheduling sehr schwer in ihrer Konsequenz zu beurteilen sind. Prinzipiell wären dazu Verfahren wie Q-Learning oder TD-Learning geeignet, aber die Formulierung einer geeigneten Belohnungsfunktion stellt eine große Herausforderung dar. Online adaptive Grid-Schedulingverfahren könnten dann in bestehende Middleware-Schnittstellen integriert werden und wären vom Benutzer ohne weitere Konfiguration oder Lernphase direkt nutzbar. Dies würde die algorithmische Grundlage für die Umsetzung sich selbst organisierender dezentraler Grids bilden, bei denen durch geschicktes Ressourcen-Management alle Communities deutlich bessere Serviceleistungen für ihre Mitglieder erzielen können.

Eigene Veröffentlichungen

[1] ERNEMANN, Carsten; KROGMANN, Martin; LEPPING, Joachim; YAHYAPOUR, Ramin: **Scheduling on the Top 50 Machines**. In: FEITELSON, D. G. (Hrsg.); RUDOLPH, L. (Hrsg.); SCHWIEGELSHOHN, U. (Hrsg.): *Proceedings of the 10th Workshop on Job Scheduling Strategies for Parallel Processing* Bd. 3277, Springer, 2004 (Lecture Notes in Computer Science), S. 17–46

[2] FÖLLING, Alexander; GRIMME, Christian; LEPPING, Joachim; PAPASPYROU, Alexander: **Co-evolving Fuzzy Rule Sets for Job Exchange in Computational Grids**. In: *Proceedings of the IEEE International Conference on Fuzzy Systems*, IEEE Computer Society Press, 2009, S. 1683–1688

[3] FÖLLING, Alexander; GRIMME, Christian; LEPPING, Joachim; PAPASPYROU, Alexander: **Decentralized Grid Scheduling with Evolutionary Fuzzy Systems**. In: FRACHTENBERG, E. (Hrsg.); SCHWIEGELSHOHN, U. (Hrsg.): *Proceedings of the 14th Workshop on Job Scheduling Strategies for Parallel Processing* Bd. 5798, Springer, 2009 (Lecture Notes in Computer Science), S. 16–36

[4] FÖLLING, Alexander; GRIMME, Christian; LEPPING, Joachim; PAPASPYROU, Alexander: **Robust Load Delegation in Service Grid Environments**. In: *IEEE Transactions on Parallel and Distributed Systems* 21 (2010), Nr. 9, S. 1304–1316

[5] FÖLLING, Alexander; GRIMME, Christian; LEPPING, Joachim; PAPASPYROU, Alexander: **The Gain of Resource Delegation in Distributed Computing Environments**. In: SCHWIEGELSHOHN, U. (Hrsg.); FRACHTENBERG, E. (Hrsg.): *Proceedings of the 15th Workshop on Job Scheduling Strategies for Parallel Processing* Bd. 6253, Springer, 2010 (Lecture Notes in Computer Science), S. 77–92

[6] FÖLLING, Alexander; GRIMME, Christian; LEPPING, Joachim; PAPASPYROU, Alexander; SCHWIEGELSHOHN, Uwe: **Competitive Co-evolutionary Learning of Fuzzy Systems for Job Exchange in Computational Grids**. In: *Evolutionary Computation* 17 (2009), Nr. 4, S. 545–560

[7] FRANKE, Carsten; HOFFMANN, Frank; LEPPING, Joachim; SCHWIEGELSHOHN, Uwe: **Development of Scheduling Strategies with Genetic Fuzzy Systems**. In: *Applied Soft Computing* 8 (2008), Nr. 1, S. 706–721

[8] FRANKE, Carsten; LEPPING, Joachim; SCHWIEGELSHOHN, Uwe: **On Advantages of Scheduling using Genetic Fuzzy Systems**. In: *Proceedings of the 12th Workshop on Job Scheduling Strategies for Parallel Processing* Bd. 4376, Springer, 2006 (Lecture Notes in Computer Science), S. 68–93

[9] FRANKE, Carsten; LEPPING, Joachim; SCHWIEGELSHOHN, Uwe: **Genetic Fuzzy Systems applied to Online Job Scheduling**. In: *Proceedings of the IEEE International Conference on Fuzzy Systems*, IEEE Computer Society Press, 2007, S. 1573–1578

[10] FRANKE, Carsten; LEPPING, Joachim; SCHWIEGELSHOHN, Uwe: **Greedy Scheduling with Complex Objectives**. In: *Proceedings of the IEEE Symposium on Computational Intelligence in Scheduling*, IEEE Computer Society Press, 2007, S. 113–120. – CD-ROM

[11] FRANKE, Carsten; LEPPING, Joachim; SCHWIEGELSHOHN, Uwe: **Greedy Scheduling with Custom-made Objectives**. In: *Annals of Operations Research* 180 (2010), November, Nr. 1, S. 145–164

[12] GRIMME, Christian; LEPPING, Joachim: **Designing Multi-Objective Variation Operators Using a Predator-Prey Approach**. In: *Proceedings of the 4th International Conference on Evolutionary Multi-Criterion Optimization* Bd. 4403, Springer, 2007 (Lecture Notes in Computer Science), S. 21–35

[13] GRIMME, Christian; LEPPING, Joachim; PAPASPYROU, Alexander: **Exploring the Behavior of Building Blocks for Multi-Objective Variation Operator Design using Predator-Prey Dynamics**. In: THIERENS, D. (Hrsg.): *Proceedings of the Genetic and Evolutionary Computation Conference*, ACM Press, 2007, S. 805–812

[14] GRIMME, Christian; LEPPING, Joachim; PAPASPYROU, Alexander: **Identifying Job Migration Characteristics in Decentralized Grid Scheduling Scenarios**. In: ZHENG, S. Q. (Hrsg.): *Proceedings of the 19th International Conference on Parallel and Distributed Computing and Systems*, ACTA Press, 2007, S. 124–129

[15] GRIMME, Christian; LEPPING, Joachim; PAPASPYROU, Alexander: **Prospects of Collaboration between Compute Providers by means of Job Interchange**. In: FRACHTENBERG, E. (Hrsg.); SCHWIEGELSHOHN, U. (Hrsg.): *Proceedings of the 13th Workshop on Job Scheduling Strategies for Parallel Processing* Bd. 4942, Springer, 2007 (Lecture Notes in Computer Science), S. 132–151

[16] GRIMME, Christian; LEPPING, Joachim; PAPASPYROU, Alexander: **Benefits of Job Exchange Between Autonomous Sites in Decentralized Computational Grids**. In: PRIOL, T. (Hrsg.); LEFEVRE, L. (Hrsg.); BUYYA, R. (Hrsg.): *Proccedings of the 8th IEEE/ACM International Symposium on Cluster Computing and the Grid*, IEEE Computer Society Press, 2008, S. 25–32

[17] GRIMME, Christian; LEPPING, Joachim; PAPASPYROU, Alexander: **Discovering Performance Bounds for Grid Scheduling by using Evolutionary Multiobjective Optimization**. In: KEIJZER, M. (Hrsg.): *Proceedings of the Genetic and Evolutionary Computation Conference*, ACM Press, 2008, S. 1491–1498

[18] GRIMME, Christian; LEPPING, Joachim; PAPASPYROU, Alexander: **The Parallel Predator-Prey Model: A Step Towards Practical Application**. In: RUDOLPH, G. (Hrsg.): *Proceedings of the 10th International Conference on Parallel Problem Solving From Nature* Bd. 5199, Springer, 2008 (Lecture Notes in Computer Science), S. 681–690

[19] GRIMME, Christian; LEPPING, Joachim; PAPASPYROU, Alexander: **Adapting to the Habitat: On the Integration of Local Search into the Predator-Prey Model**. In: EHRGOTT, M. (Hrsg.); FONSECA, C. M. (Hrsg.); GANDIBLEUX, X. (Hrsg.); HAO, J.-K. (Hrsg.); SEVAUX, M. (Hrsg.): *Proceedings of the 5th International Conference on Evolutionary Multi-Criterion Optimization*, Springer, 2009 (Lecture Notes in Computer Science 5467), S. 510–524

[20] GRIMME, Christian; LEPPING, Joachim; PAPASPYROU, Alexander; WIEDER, Philipp; YAHYAPOUR, Ramin; OLEKSIAK, Ariel; WÄLDRICH, Oliver; ZIEGLER, Wolfgang: **To-**

wards a standards-based Grid Scheduling Architecture / Institute on Resource Management and Scheduling. 2007 (TR-0123). – CoreGRID Technical Report

[21] GRIMME, Christian; LEPPING, Joachim; PICON, Jonathan M.; PAPASPYROU, Alexander: **Applying P2P Strategies to Scheduling in Decentralized Grid Computing Infrastructures**. In: *Proceedings of the 39th International Conference on Parallel Processing Workshops*, IEEE Computer Society, 2010, S. 295–302

[22] KRAMPE, Anne; LEPPING, Joachim; SIEBEN, Wiebke: **A Hybrid Markov Chain Modeling Architecture for Workload on Parallel Computers** / Collaborative Research Center 531, Dortmund University of Technology. 2008 (246/08). – CI-Series Technical Report

[23] KRAMPE, Anne; LEPPING, Joachim; SIEBEN, Wiebke: **A Hybrid Markov Chain Model for Workload on Parallel Computers**. In: EPEMA, D. H. J. (Hrsg.): *Proceedings of the Workshop on Large-Scale System and Application Performance*, ACM Press, 2010, S. 589–596

Weitere Literaturverweise

[24] ALBERS, Susanne: **Online Algorithms: A Survey**. In: *Mathematical Programming* 97 (2003), S. 3–26

[25] ALDABBAS, Omar; ALFAWAIR, Mai; ZEDAN, Hussein; CAU, Antonio: **Temporal dimension for job submission description language**. In: *Proceedings of the WSEAS International Conference on Software Engineering, Parallel and Distributed Systems*, World Scientific, 2008, S. 80–88

[26] ALTMANN, Jörn; ION, Mihaela; MOHAMMED, Ashraf Adel B.: **Taxonomy of Grid Business Models**. In: *Grid Economics and Business Models* Bd. 4685, Springer, 2007 (Lecture Notes in Computer Science), S. 29–43

[27] ANDREETTO, Paolo; BORGIA, Stefano; DORIGO, Alvise u. a.: **Practical Approaches to Grid Workload and Resource Management in the EGEE Project**. In: *Proceedings of the Conference on Computing in High Energy and Nuclear Physics*, Organisation Européenne pour la Recherche Nucléaire (online), 2004, S. 899–902

[28] ANOEP, Shanny; DUMITRESCU, Catalin; EPEMA, Dick; IOSUP, Alexandru; JAN, Mathieu; LI, Hui; WOLTERS, Lex: *Grid Workload Archive*. http://gwa.ewi.tudelft.nl, November 2010

[29] AXELROD, Robert: *The Evolution of Cooperation*. 1. Auflage. Basic Books, 1984

[30] BERMAN, Fran; FOX, Geoffrey; HEY, Anthony J.: *Grid Computing: Making The Global Infrastructure a Reality*. 1. Auflage. John Wiley & Sons, 2003

[31] BEUME, Nicola; NAUJOKS, Boris; EMMERICH, Michael: **SMS-EMOA: Multiobjective Selection based on Dominated Hypervolume**. In: *European Journal of Operational Research* 181 (2007), S. 1653–1669

[32] BEYER, Hans G.: *The Theory of Evolution Strategies*. 1. Auflage. Springer, 2001

[33] BLUMOFE, Robert D.; LEISERSON, Charles E.: **Scheduling Multithreaded Computations by Work Stealing**. In: *Proceedings of the Annual Symposium on Foundations of Computer Science*, 1994, S. 356–368

[34] BONARINI, Andrea: **Evolutionary Learning of Fuzzy Rules: Competition and Cooperation**. In: PEDRYCZ, W. (Hrsg.): *Fuzzy Modelling: Paradigms and Practice*. Kluwer Academic Press, 1996, S. 265–284

[35] BRANDENBURGER, Adam M.; NALEBUFF, Barry J.: *Co-Opetition*. 1. Auflage. Broadway Business, 1997

[36] BREMERMANN, Hans J.: **Optimization through Evolution and Recombination**. In: YOVITS, M. (Hrsg.); JACOBI, G. T. (Hrsg.); GOLDSTEIN, G. D. (Hrsg.): *Self-Organizing Systems-1962*, Spartan Books, 1962, S. 93–106

[37] BRUCKER, Peter: *Scheduling Algorithms*. 4. Auflage. Springer, 2004

[38] BUYYA, Rajkumar; ABRAMSON, David; GIDDY, Jonathan; STOCKINGER, Heinz: **Economic Models for Resource Management and Scheduling in Grid Computing**. In: *Concurrency and Computation: Practice and Experience* 14 (2002), S. 1507–1542

[39] BUYYA, Rajkumar; ABRAMSON, David; VENUGOPAL, Srikumar: **The Grid Economy**. In: *Special Issue of the Proceedings of the IEEE on Grid Computing* Bd. 93, IEEE Computer Society Press, 2005, S. 698 – 714

[40] BUYYA, Rajkumar; MURSHED, Manzur: **GridSim: a Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing**. In: *Concurrency and Computation: Practice and Experience* 14 (2002), S. 1175–1220

[41] CASANOVA, Henri: **SimGrid: A Toolkit for the Simulation of Application Scheduling**. In: *Proceedings of the IEEE/ACM International Symposium on Cluster Computing and the Grid*, IEEE Computer Society Press, 2001, S. 430–437

[42] CHAPIN, Steve J.; CIRNE, Walfredo; FEITELSON, Dror G.; JONES, J. P.; LEUTENEGGER, Scott T.; SCHWIEGELSHOHN, Uwe; SMITH, Warren; TALBY, David: **Benchmarks and Standards for the Evaluation of Parallel Job Schedulers**. In: *Proceedings of the 5th Workshop on Job Scheduling Strategies for Parallel Processing* Bd. 1659, Springer, 1999 (Lecture Notes in Computer Science), S. 67–90

[43] CIRNE, Walfredo; BERMAN, Francine: **A Model for Moldable Supercomputer Jobs**. In: *Proceedings of the International Parallel and Distributed Processing Symposium*, IEEE Computer Society Press, 2001, S. 8–16

[44] COELLO COELLO, Carlos A.: **Twenty Years of Evolutionary Multi-Objective Optimization: A Historical View of the Field**. In: *IEEE Computational Intelligence Magazine* 1 (2006), Nr. 1, S. 28–36

[45] COELLO COELLO, Carlos A.; VAN VELDHUIZEN, David. A.; LAMONT, Gary B.: *Evolutionary Algorithms for Solving Multi-Objective Problems*. 2. Auflage. Springer, 2007

[46] CORDÓN, Oscar; HERRERA, Francisco; HOFFMANN, Frank; MAGDALENA, Luis: *Advances in Fuzzy Systems – Applications and Theory*. Bd. 19: *Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*. 1. Auflage. World Scientific, 2001

[47] CRISTIANO, Kevin; GRUBER, Ralf; KELLER, Vincent; PIERRE, Kuonen; MAFFIOLETTI, Sergio; NELLARI, Nello; SAWLEY, Marie-Christine; SPADA, Michela; TRAN, Trach-Minh; WÄLDRICH, Oliver; WIEDER, Philipp; ZIEGLER, Wolfgang: **Integration of ISS into the VIOLA Meta-scheduling Environment**. In: GORLATCH, S. (Hrsg.); DANELUTTO, M. (Hrsg.): *Proceedings of the Integrated Research in Grid Computing Workshop*, Universita di Pisa, 2005, S. 357–366

[48] DEB, Kalyanmoy: *Multi-Objective Optimization using Evolutionary Algorithms*. 1. Auflage. John Wiley & Sons, 2001

[49] DEB, Kalyanmoy; AGRAWAL, Ram B.: **Simulated Binary Crossover for Continuous Search Space**. In: *Complex Systems* 9 (1995), S. 115–148

[50] DEB, Kalyanmoy; AGRAWAL, Subrajyoti; PRATAB, Amrit; MEYARIVAN, T.: **A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II**. In: SCHOENAUER, M. (Hrsg.): *Proceedings of the International Conference on Parallel Problem Solving from Nature* Bd. 1917, Springer, 2000 (Lecture Notes in Computer Science), S. 849–858

[51] DOWNEY, Allen B.; FEITELSON, Dror G.: **The Elusive Goal of Workload Characterization**. In: *Performance Evaluation Review* 26 (1999), Nr. 4, S. 14–29

[52] DROZDOWSKI, Maciej: **Scheduling Multiprocessor Tasks – An Overview**. In: *European Journal of Operational Research* 94 (1996), S. 215–230

[53] ENGELBRECHT, Andries P.: *Computational Intelligence – An Introduction*. 2. Auflage. John Wiley & Sons, 2007

[54] ERNEMANN, C.; SCHWIEGELSHOHN, U.; EMMERICH, M.; SCHÖNEMANN, L.; BEUME, N.: **Scheduling Algorithm Development based on Complex Owner Defined Objectives** / SFB 531, CI-Report. 44221 Dortmund, Germany : TU Dortmund University, 2005 (191/05). – Technical Report

[55] ERNEMANN, Carsten; HAMSCHER, Volker; SCHWIEGELSHOHN, Uwe; STREIT, Achim; YAHYAPOUR, Ramin: **Enhanced Algorithms for Multi-Site Scheduling**. In: PARASHAR, M. (Hrsg.): *Proceedings of the IEEE/ACM International Workshop on Grid Computing* Bd. 2536, Springer, 2002 (Lecture Notes in Computer Science), S. 219–231

[56] ERNEMANN, Carsten; HAMSCHER, Volker; SCHWIEGELSHOHN, Uwe; STREIT, Achim; YAHYAPOUR, Ramin: **On Advantages of Grid Computing for Parallel Job Scheduling**. In: *Proccedings of the IEEE/ACM International Symposium on Cluster Computing and the Grid*, IEEE Computer Society Press, 2002, S. 39–46

[57] ERNEMANN, Carsten; HAMSCHER, Volker; YAHYAPOUR, Ramin: **Economic Scheduling in Grid Computing**. In: *Proceedings of the 8th Workshop on Job Scheduling Strategies for Parallel Processing* Bd. 2537, Springer, 2002 (Lecture Notes in Computer Science), S. 128–152

[58] ERNEMANN, Carsten; HAMSCHER, Volker; YAHYAPOUR, Ramin: **Benefits of Global Grid Computing for Job Scheduling**. In: *Proceedings of the IEEE/ACM International Workshop on Grid Computing*, IEEE Computer Society Press, 2004, S. 374–379

[59] ERWIN, Dietmar W.; SNELLING, David F.: **UNICORE: A Grid Computing Environment**. In: GOOS, G. (Hrsg.); HARTMANIS, J. (Hrsg.); LEEUWEN, J. van (Hrsg.): *Proceedings of the 7th International Euro-Par Conference* Bd. 2150, Springer, 2001 (Lecture Notes in Computer Science), S. 825–834

[60] ETSION, Yoav; TSAFRIR, Dan: **A Short Survey of Commercial Cluster Batch Schedulers** / School of Computer Science and Engineering, The Hebrew University of Jerusalem. 44221 Dortmund, Germany, 2005 (2005-13). – Technical Report

[61] FAYAD, Carole; GARIBALDI, Jonathan M.; OUELHADJ, Djamila: **Fuzzy Grid Scheduling Using Tabu Search**. In: *Proceedings of the IEEE International Conference on Fuzzy Systems*, IEEE Computer Society Press, 2007, S. 1–6

[62] FEITELSON, Dror G.: **Packing Schemes for Gang Scheduling**. In: FEITELSON, D. G. (Hrsg.); RUDOLPH, L. (Hrsg.): *Proceedings of the 2nd Workshop on Job Scheduling Strategies for Parallel Processing* Bd. 1162, Springer, 1996 (Lecture Notes in Computer Science), S. 89–110

[63] FEITELSON, Dror G.: **Metrics for Parallel Job Scheduling and their Convergence**. In: FEITELSON, D. G. (Hrsg.); RUDOLPH, L. (Hrsg.): *Proceedings of the 7th Workshop on Job Scheduling Strategies for Parallel Processing* Bd. 2221, Springer, 2001, S. 188–206

[64] FEITELSON, Dror G.: **Workload Modeling for Performance Evaluation**. In: CALZAROSSA, M. C. (Hrsg.); TUCCI, S. (Hrsg.): *Performance Evaluation of Complex Systems: Techniques and Tools* Bd. 2459. Springer, 2002, S. 114–141

[65] FEITELSON, Dror G.: *Parallel Workload Archive*. http://www.cs.huji.ac.il/labs/parallel/workload/, November 2010

[66] FEITELSON, Dror G.; FRACHTENBERG, Eitan: **Pitfalls in Parallel Job Scheduling Evaluation**. In: *Proceedings of the 11th Workshop on Job Scheduling Strategies for Parallel Processing* Bd. 3834, Springer, 2005 (Lecture Notes in Computer Science), S. 257–282

[67] FEITELSON, Dror G.; RUDOLPH, Larry: **Towards Convergence in Job Schedulers for Parallel Supercomputers**. In: *Proceedings of the 2nd Workshop on Job Scheduling Strategies for Parallel Processing*, Springer, 1996, S. 1–26

[68] FEITELSON, Dror G.; RUDOLPH, Larry; SCHWIEGELSHOHN, Uwe: **Parallel Job Scheduling – A Status Report**. In: FEITELSON, D. G. (Hrsg.); RUDOLPH, L. (Hrsg.): *Proceedings of the 10th Workshop on Job Scheduling Strategies for Parallel Processing* Bd. 3277, Springer, 2004 (Lecture Notes in Computer Science), S. 1–16

[69] FEITELSON, Dror G.; TSAFRIR, Dan: **Workload Sanitation for Performance Evaluation**. In: *IEEE International Symposium on Performance Analysis of Systems & Software*, IEEE Computer Society Press, 2006, S. 221–230

[70] FEITELSON, Dror G.; WEIL, Ahuva M.: **Utilization and Predictability in Scheduling the IBM SP2 with Backfilling**. In: *Proceedings of the 12th International Parallel Processing Symposium and the 9th Symposium on Parallel and Distributed Processing*, IEEE Computer Society Press, 1998, S. 542–547

[71] FOGEL, Lawrence J.: **Autonomous Automata**. In: *Industrial Research* 4 (1962), S. 14–19

[72] FONSECA, Carlos M.; FLEMING, Peter J.: **Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization**. In: *Proceedings of the 5th International Conference on Genetic Algorithms*, Morgan Kaufmann, 1993, S. 416–423

[73] FOSTER, Ian: **Globus Toolkit Version 4: Software for Service-Oriented Systems.** In: JIN, H. (Hrsg.); REED, D. A. (Hrsg.); JIANG, W. (Hrsg.): *IFIP International Conference on Network and Parallel Computing* Bd. 3779, Springer, 2005 (Lecture Notes in Computer Science), S. 2–13

[74] FOSTER, Ian; KESSELMAN, Carl: *The Grid: Blueprint for a New Computing Infrastructure*. 2. Auflage. Morgan Kaufmann, 2003

[75] FOSTER, Ian T.; ZHAO, Yong; RAICU, Ioan; LU, Shiyong: **Cloud Computing and Grid Computing 360-Degree Compared**. In: *Grid Computing Environments Workshop*, IEEE Computer Society Press, 2009, S. 1–10

[76] FRANKE, Carsten: *Design and Evaluation of Multi-Objective Online Scheduling Strategies for Parallel Machines using Computational Intelligence*. University of Dortmund, 44221 Dortmund, Germany, University of Dortmund, Diss., 2006

[77] FRANKE, Carsten; SCHWIEGELSHOHN, Uwe; YAHYAPOUR, Ramin: **Job Scheduling for Computational Grids** / Dortmund University, Department of Electrical Engineering and Information Technology. 44221 Dortmund, Germany : TU Dortmund University, 2006 (0206). – Technical Report

[78] FRASER, Alex S.: **Simulation of Genetic Systems by Automatic Digital Computers**. In: *Australian Journal of Biological Science* 10 (1957), S. 484–499

[79] FREITAG, Stefan; GRIMME, Christian; PAPASPYROU, Alexander; SCHLEY, Lars: **On the Applicability of OGSA-BES to D-Grid Community Scheduling Systems**. In: *German E-Science Conference*, Max-Planck Society (online), 2007, S. 1–9

[80] GAREY, Michael R.; GRAHAM, Ronald L.: **Bounds for Multiprocessor Scheduling with Resource Constraints**. In: *SIAM Journal on Computing* 4 (1975), Nr. 2, S. 187–200

[81] GOLDBERG, David E.: *Genetic Algorithms in Search, Optimization and Machine Learning*. 1. Auflage. Addison-Wesley, 1989

[82] GRAHAM, Ronald L.: **Bounds on Multiprocessing Timing Anomalies**. In: *SIAM Journal of Applied Mathematics* 17 (1969), Nr. 2, S. 416–429

[83] GRAHAM, Ronald L.; LAWLER, Eugene L.; LENSTRA, Jan. K.; RINNOOY-KAN, A. H. G.: **Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey**. In: *Annals of Discrete Mathematics* 15 (1979), S. 287–326

[84] GRIMME, Christian; PAPASPYROU, Alexander: **Cooperative Negotiation and Scheduling of Scientific Workflows in the Collaborative Climate Community Data and Processing Grid**. In: *Future Generation Computer Systems* 25 (2009), S. 301–307

[85] GRIMME, Christian; SCHMITT, Karlheinz: **Inside a Predator-Prey Model for Multi-Objective Optimization: A Second Study**. In: H.-G.BEYER (Hrsg.): *Proceedings of the Genetic and Evolutionary Computation Conference*, ACM Press, 2006, S. 707–714

[86] GRUBER, Ralf; KELLER, Vincent; MANNEBACK, Pierre; THIEMARD, Michela; WÄLDRICH, Oliver; WIEDER, Philipp; ZIEGLER, Wolfgang: **Integration of Grid Cost Model into ISS/VIOLA Meta-Scheduler Environment**. In: *Proceedings of the 12th International Euro-Par Conference*, Springer, 2007 (Lecture Notes in Computer Science), S. 215–224

[87] HENDERSON, Robert L.: **Job Scheduling Under the Portable Batch System**. In: *Proceedings of the 1st Workshop on Job Scheduling Strategies for Parallel Processing*, Springer, 1995, S. 279–294

[88] HILLIS, W. D.: **Co-evolving Parasites Improve Simulated Evolution as an Optimization Procedure**. In: *Physics D* 42 (1990), Nr. 1-3, S. 228–234

[89] HOLLAND, John H.: **Outline for a Logical Theory of Adaptive Systems**. In: *Journal of the ACM* 9 (1962), S. 297–314

[90] HOLMBLAD, L. P.; ØSTERGAARD, J.-J.: **The FLS Application of Fuzzy Logic**. In: *Fuzzy Sets Systems* 70 (1995), Nr. 2-3, S. 135–146

[91] HOPFIELD, John J.: **Neural Networks and Physical Systems with Emergent Collective Computational Abilities**. In: *Proceedings of the National Academy of Science of the USA* 79 (1988), Nr. 8, S. 457–464

[92] HOTOVY, Steven: **Workload Evolution on the Cornell Theory Center IBM SP2**. In: *Proceedings of the 2nd Workshop on Job Scheduling Strategies for Parallel Processing* Bd. 1162, Springer, 1996 (Lecture Notes in Computer Science), S. 27–40

[93] HUANG, Jin; JIN, Hai; XIE, Xia; ZHANG, Qin: **An Approach to Grid Scheduling Optimization Based on Fuzzy Association Rule Mining**. In: *Proceedings of the 1st International Conference on e-Science and Grid Computing*, IEEE Computer Society Press, 2005, S. 189–195

[94] HUEDO, Eduardo; MONTERO, Rubén S.; LLORENTE, Ignacio M.: **A Recursive Architecture for Hierarchical Grid Resource Management**. In: *Future Generation Computing System* 25 (2009), Nr. 4, S. 401–405

[95] ITO, Koichi; AKAGI, Shinsuke; NISHIKAWA, Masaaki: **A Multiobjective Optimization Approach to a Design Problem of Heat Insulation for Thermal Distribution Piping Network Systems**. In: *Journal of Mechanisms, Transmissions and Automation in Design (Transactions of the ASME)* 105 (1983), S. 206–213

[96] JAGERMAN, David L.; MELAMED, Benjamin; WILLINGER, Walter: **Stochastic Modeling of Traffic Processes**. In: DSHALALOW, J. H. (Hrsg.): *Frontiers in Queuing*. CRC Press, 1997 (Probability and Stochastics), S. 271–320

[97] JANN, Joefon; PATTNAIK, Pratap; FRANKE, Hubertus; WANG, Fang; SKOVIRA, Joseph; RIODAN, Joseph: **Modeling of Workload in MPPs**. In: FEITELSON, D. G. (Hrsg.); RUDOLPH, L. (Hrsg.): *Proceedings of the 3rd Workshop on Job Scheduling Strategies for Parallel Processing* Bd. 1291, Springer, 1997 (Lecture Notes in Computer Science), S. 95–116

[98] JANSEN, Thomas; WIEGAND, R. P.: **Exploring the Explorative Advantage of the Cooperative Coevolutionary $(1+1)$ EA**. In: CANTÚ-PAZ, E. (Hrsg.): *Proceedings of the Genetic and Evolutionary Computation Conference* Bd. 2723, Springer, 2003 (Lecture Notes in Computer Science), S. 310–321

[99] JIN, Yaochu; SEELEN, Werner von; SENDHOFF, Bernhard: **On Generating FC^3 Fuzzy Rule Systems from Data Using Evolution Strategies**. In: *IEEE Transactions on System, Man and Cybernetics* 29 (1999), Nr. 6, S. 829–845

[100] JUANG, Chia-Feng; LIN, Jiann-Yow; LIN, Chin-Teng: **Genetic Reinforcement Learning through Symbiotic Evolution for Fuzzy Controller Design**. In: *IEEE Transactions on System, Man and Cybernetics* 30 (2000), Nr. 2, S. 290–302

[101] KALÉ, Laxmikant V.; KUMARAND, Sameer; DESOUZA, Jayant: **A Malleable-Job System for Timeshared Parallel Machines**. In: *Proccedings of the IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2002, S. 230–238

[102] KAMEDA, Hisao; LI, Jie; KIM, Chonggun; ZHANG, Yongbing: *Optimal Load Balancing in Distributed Computer Systems*. 1. Auflage. Springer, 1997

[103] KANDEL, Abraham; LANGHOLZ, Gideon: *Fuzzy Control Systems*. 1. Auflage. CRC Press, 1993

[104] KAWAGUCHI, Tsuyoshi; KYAN, Seiki: **Worst Case Bound of an LRF Schedule for the Mean Weighted Flow-time Problem**. In: *SIAM Journal on Computing* 15 (1986), Nr. 4, S. 1119–1129

[105] KNOWLES, Joshua; CORNE, David: **Quantifying the Effects of Objective Space Dimension in Evolutionary Multiobjective Optimization**. In: *Proceedings of the 4th International Conference on Evolutionary Multi-Criterion Optimization*, 2007 (Lecture Notes in Computer Science 4403), S. 757–771

[106] KOST, Bernd: *Optimierung mit Evolutionsstrategien*. 1. Auflage. Verlag Harri Deutsch, 2003

[107] KOTZ, Samuel; READ, Campbell B.; BALAKRISHNAN, Narayanaswamy; VIDAKOVIC, Brani: *Encyclopedia of Statistical Science*. Bd. 4. 2. Auflage. Wiley Press, 2006

[108] KRALLMANN, Jochen; SCHWIEGELSHOHN, Uwe; YAHYAPOUR, Ramin: **On the Design and Evaluation of Job Scheduling Systems**. In: FEITELSON, D. G. (Hrsg.); RUDOLPH, L. (Hrsg.): *Proceedings of the 5th Workshop on Job Scheduling Strategies for Parallel Processing* Bd. 1659, Springer, 1999 (Lecture Notes in Computer Science), S. 17–42

[109] KURSAWE, Frank: **A Variant of Evolution Strategies for Vector Optimization**. In: SCHWEFEL, H.-P. (Hrsg.); MÄNNER, R. (Hrsg.): *Proceedings of the International Conference on Parallel Problem Solving from Nature*, Springer, 1991, S. 193–197

[110] LAUMANNS, Marco; RUDOLPH, Günter; SCHWEFEL, Hans-Paul: **A Spatial Predator-Prey Approach to Multi-Objective Optimization: A Preliminary Study**. In: BÄCK, T. (Hrsg.); EIBEN, A. E. (Hrsg.); SCHOENAUER, M. (Hrsg.); SCHWEFEL, H.-P. (Hrsg.): *Proceedings of the International Conference on Parallel Problem Solving from Nature*. Springer, 1998, S. 241–249

[111] LEE, Cynthia B.; SCHWARTZMAN, Yael; HARDY, Jennifer; SNAVELY, Allan: **Are User Runtime Estimates Inherently Inaccurate?** In: *Proceedings of the 10th Workshop on Job Scheduling Strategies for Parallel Processing* Bd. 3277, Springer, 2004 (Lecture Notes in Computer Science), S. 253–263

[112] LI, Hui: **Workload Dynamics on Clusters and Grids**. In: *The Journal of Supercomputing* 47 (2009), Nr. 1, S. 1–20

[113] LI, Hui; MUSKULUS, Michael: **Analysis and Modeling of Job Arrivals in a Production Grid**. In: *ACM Sigmetrics Performance Evaluation Review* 34 (2007), Nr. 4, S. 59–70

[114] LI, Hui; MUSKULUS, Michael; WOLTERS, Lex: **Modeling Correlated Workloads by Combining Model Based Clustering and A Localized Sampling Algorithm**. In: *Proceedings of the 21st ACM International Conference on Supercomputing*, ACM Press, 2007, S. 16–20

[115] LIFKA, David A.: **The ANL/IBM SP Scheduling System**. In: *Proceedings of the 1st Workshop on Job Scheduling Strategies for Parallel Processing* Bd. 949, Springer, 1995 (Lecture Notes in Computer Science), S. 295–303

[116] LO, Virginia; MACHE, Jens; WINDISCH, Kurt: **A Comparative Study of Real Workload Traces and Synthetic Workload Models for Parallel Job Scheduling**. In: FEITELSON, D. G. (Hrsg.); RUDOLPH, L. (Hrsg.): *Proceedings of the 4th Workshop on Job Scheduling Strategies for Parallel Processing* Bd. 1459, Springer, 1998 (Lecture Notes in Computer Science), S. 25–46

[117] LU, Kai; SUBRATA, Riky; ZOMAYA, Albert Y.: **Towards Decentralized Load Balancing in a Computational Grid Environment**. In: *International Conference an Advances in Grid and Pervasive Computing* Bd. 3947, Springer, 2006 (Lecture Notes in Computer Science), S. 466–477

[118] LUBLIN, Uri; FEITELSON, Dror G.: **The Workload on Parallel Supercomputers: Modeling the Characteristics of Rigid Jobs.** In: *Journal of Parallel and Distributed Computing* 63 (2003), Nr. 11, S. 1105–1122

[119] MAMDANI, Ebrahim H.; ASSILIAN, Sedrak: **Application of Fuzzy Algorithms for Control of Simple Dynamic Plant**. In: *Proceedings of the IEEE* 121 (1974), Nr. 12, S. 1585–1588

[120] MCNAUGHTON, R.: **Scheduling with Deadlines and Loss Functions**. In: *Management Science* 6 (1959), Nr. 1, S. 1–12

[121] MEDERNACH, Emmanuel: **Workload Analysis of a Cluster in a Grid Environment**. In: FEITELSON, D. G. (Hrsg.); FRACHTENBERG, E. (Hrsg.); RUDOLPH, L. (Hrsg.); SCHWIEGELSHOHN, U. (Hrsg.): *Proceedings of the 11th Workshop on Job Scheduling Strategies for Parallel Processing* Bd. 3834, Springer, 2005 (Lecture Notes in Computer Science), S. 36–61

[122] MICHAELI, Rainer: *Competitive Intelligence – Strategische Wettbewerbsvorteile erzielen durch systematische Konkurrenz-, Markt- und Technologieanalysen*. 1. Auflage. Springer, 2005

[123] MICHALEWICZ, Zbigniew: *Genetic Algorithms + Data Structures = Evolution Programs*. 2. Auflage. Springer, 1998

[124] MICHELS, Kai; KLAWONN, Frank; KRUSE, Rudolf; NÜRNBERGER, Andreas: *Fuzzy-Regelung*. 1. Auflage. Springer, 2002

[125] MIETTINEN, Kaisa M.: *Nonlinear Multiobjective Optimization*. 1. Auflage. Kluwer Academic Press, 1999

[126] MINSKY, Marvin; PAPERT, Seymour: *Perceptrons: An Introduction to Computational Geometry*. 1. Auflage. MIT Press, 1969

[127] MOTWANI, Rajeev; PHILLIPS, Steven; TORNG, Eric: **Nonclairvoyant Scheduling**. In: *Theoretical Computer Science* 130 (1994), Nr. 1, S. 17–47

[128] MU'ALEM, Ahuva W.; FEITELSON, Dror G.: **Bicriteria Scheduling for Parallel Jobs**. In: *Multidisciplinary International Conference on Scheduling: Theory & Applications* Bd. 2, Springer, 2003, S. 606–619

[129] MURATA, Tadahiko; ISHIBUCHI, Hisao: **Multi-objective Genetic Algorithms**. In: *Proceedings of the 2th IEEE Conference on Evolutionary Computation*, IEEE Computer Society Press, 1995, S. 289–294

[130] NAROSKA, Edwin; SCHWIEGELSHOHN, Uwe: **On an On-line Scheduling Problem with Parallel Jobs**. In: *Information Processing Letters* 81 (2002), S. 297–5304

[131] NISSIMOV, Avi; FEITELSON, Dror G.: **Probabilistic Backfilling**. In: *Proceedings of the 13th Workshop on Job Scheduling Strategies for Parallel Processing* Bd. 4942, Springer, 2008, S. 102–115

[132] OLSSON, Björn: **Co-evolutionary Search in Asymmetric Spaces**. In: *Information Sciences – Informatics and Computer Science: An International Journal* 133 (2001), Nr. 3-4, S. 103–125

[133] OSYCZKA, Andrzej: *Evolutionary Algorithms for Single and Multicriteria Design Optimization*. 1. Auflage. Springer, 2002 (Studies in Fuzziness and Soft Computing)

[134] PAREDIS, Jan: **Coevolutionary Computation**. In: *Artificial Life* 2 (1995), Nr. 4, S. 355–375

[135] PAREDIS, Jan: **Coevolutionary Algorithms**. In: BÄCK, T. (Hrsg.); FOGEL, D. B. (Hrsg.); MICHALEWICZ, Z. (Hrsg.): *Evolutionary Computation 2: Advanced Algorithms and Operators*. Institute of Physics Publishing, Bristol, 2000, S. 224–238

[136] PICHOT, Antoine; WIEDER, Philipp; WÄLDRICH, Oliver; ZIEGLER, Wolfgang: **Dynamic SLA Negotiation based on WS-Agreement** / Institute on Resource Management and Scheduling. 2008 (TR-0082). – CoreGRID Technical Report

[137] PINEDO, Mike: *Scheduling: Theory, Algorithms, and Systems*. 2. Auflage. Prentice-Hall, 2002

[138] POTTER, Mitchell A.; DE JONG, Kenneth A.: **A Cooperative Coevolutionary Approach to Function Optimization**. In: DAVIDOR, Y. (Hrsg.); SCHWEFEL, H.-P. (Hrsg.); MÄNNER, R. (Hrsg.): *Proceedings of the International Conference on Parallel Problem Solving from Nature* Bd. 866, Springer, 1994 (Lecture Notes in Computer Science), S. 249–257

[139] POTTER, Mitchell A.; DE JONG, Kenneth A.: **Cooperative coevolution: An Architecture for Evolving Coadapted Subcomponents**. In: *Evolutionary Computation* 8 (2000), Nr. 1, S. 1–29

[140] RECHENBERG, Ingo: *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. 1. Auflage. Frommann-Holzboog, 1973

[141] RECHENBERG, Ingo: *Evolutionsstrategie '94*. 1. Auflage. Frommann-Holzboog, 1994

[142] REED, Jon; TOOMBS, Robert; BARRICELLI, Nils A.: **Simulation of Biological Evolution and Machine Learning: I. Selection of Self-reproducing Numeric Patterns by Data Processing Machines, Effects of Hereditary Control, Mutation Type and Crossing**. In: *Journal of Theoretical Biology* 17 (1967), S. 319–342

[143] ROSENBLATT, Frank: *Principles of Neurodynamics; Perceptrons and the Theory of Brain Mechanisms*. 1. Auflage. Spartan Books, 1962

[144] ROSIN, Christopher D.; BELEW, Richard K.: **New Methods for Competitive Coevolution**. In: *Evolutionary Computation* 5 (1997), Nr. 1, S. 1–29

[145] ROTHLAUF, Franz: *Representations for Genetic and Evolutionary Algorithms*. 2. Auflage. Springer, 2006

[146] RUMELHART, David E.; MCCLELLAND, James: *Parallel Distributed Processing*. 1. Auflage. MIT press, 1986

[147] SASTRY, Kumara: **Single and Multiobjective Genetic Algorithm Toolbox for Matlab in C++** / Illinois Genetic Algorithms Laboratory. Urbana-Champaign, 117 Transportation Building, 104 S. Mathews Avenue Urbana, IL 61801 : University of Illinois, 2007 (2007017). – Technical Report

[148] SCHAFFER, J. D.: *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*, Vanderbilt University, Diss., 1984

[149] SCHEEL, Armin: *Beitrag zur Theorie der Evolutionsstrategie*, TU Berlin, Diss., 1985

[150] SCHLEY, Lars: **Prospects of Co-Allocation Strategies for a Lightweight Middleware in Grid Computing**. In: GONZALEZ, T. F. (Hrsg.): *Proceedings of the 20th International Conference on Parallel and Distributed Computing and Systems*, ACTA Press, 2008, S. 198–205

[151] SCHWEFEL, Hans-Paul: *Evolutionsstrategie und Numerische Optimierung*, TU Berlin, Diss., 1975

[152] SCHWEFEL, Hans-Paul: *Evolution and Optimum Seeking*. 1. Auflage. John Wiley & Sons, 1995

[153] SCHWEFEL, Hans-Paul; RUDOLPH, Günther: **Contemporary Evolution Strategies**. In: MORÁN, F. (Hrsg.); MORENO, A. (Hrsg.); MERELO, J. J. (Hrsg.); CHACÓN, P. (Hrsg.): *Proceedings of the 3rd European Conference on Artificial Life*, Springer, 1995, S. 893–907

[154] SCHWIEGELSHOHN, Uwe: **An Owner-centric Metric for the Evaluation of Online Job Schedules**. In: BLAZEWICZ, J. (Hrsg.); DROZDOWSKI, M. (Hrsg.); KENDALL, G. (Hrsg.); MCCOLLUM, B. (Hrsg.): *Proceedings of the 4th Multidisciplinary International Scheduling Conference: Theory and Applications*, 2009, S. 557–569

[155] SCHWIEGELSHOHN, Uwe; TCHERNYKH, Andrei; YAHYAPOUR, Ramin: **Online Scheduling in Grids**. In: *Proceedings of the 22nd International Parallel and Distributed Processing Symposium*, IEEE Computer Society Press, 2008, S. 1–10. – CD-ROM

[156] SCHWIEGELSHOHN, Uwe; YAHYAPOUR, Ramin: **Fairness in Parallel Job Scheduling**. In: *Journal of Scheduling* 3 (2000), Nr. 5, S. 297–320

[157] SEJNOWSKI, Terrence J.; ROSENBERG, Charles R.: **NETtalk: A Parallel Network that Learns to Read Aloud** / Johns Hopkins University. Cambridge, MA, USA : MIT Press, 1986 (JHU/EEC-86/01). – Technical Report

[158] SHMOYS, David B.; WEIN, Joel; WILLIAMSON, David P.: **Scheduling Parallel Machines On-line**. In: *SIAM Journal on Computing* 24 (1995), Nr. 6, S. 1313–1331

[159] SKOVIRA, Joseph; CHAN, Waiman; ZHOU, Honbo; LIFKA, David A.: **The EASY - LoadLeveler API Project**. In: *Proceedings of the 2nd Workshop on Job Scheduling Strategies for Parallel Processing* Bd. 1162, Springer, 1996 (Lecture Notes in Computer Science), S. 41–47

[160] SLAWINSKI, Timo; KRONE, Angelika; KRAUSE, Peter; KIENDL, Harro: **The Fuzzy-ROSA Method: A Statistically Motivated Fuzzy Approach for Data-Based Generation of Small Interpretable Rule Bases in High-dimensional Search Spaces**. In: LAST, M. (Hrsg.); KANDL, A. (Hrsg.); BUNKE, H. (Hrsg.): *Data Mining and Computational Intelligence*, Physica-Verlag, 2001, S. 141–166

[161] SMARR, Larry; CATLETT, Charles E.: **Metacomputing**. In: *Communications fo the ACM* 35 (1992), Nr. 6, S. 44–52

[162] SMITH, Stephen F.: *A Learning System Based on Genetic Adaptive Algorithms*, Department of Computer Science, University of Pittsburgh, Diss., 1980

[163] SONG, Baiyi; ERNEMANN, Carsten; YAHYAPOUR, Ramin: **Modeling of Parameters in Supercomputer Workloads**. In: *Workshop on Parallel Systems and Algorithms in Conjunction with the International Conference on Architecture of Computing Systems: Organic and Pervasive Computing* Bd. P-41, Gesellschaft für Informatik, 2004 (Lecture Notes in Informatics), S. 400–409

[164] SONG, Baiyi; ERNEMANN, Carsten; YAHYAPOUR, Ramin: **Parallel Computer Workload Modeling with Markov Chains**. In: FEITELSON, D. G. (Hrsg.); RUDOLPH, L. (Hrsg.); SCHWIEGELSHOHN, U. (Hrsg.): *Proceedings of the 10th Workshop on Job Scheduling Strategies for Parallel Processing* Bd. 3277, Springer, 2004 (Lecture Notes in Computer Science), S. 47–62

[165] SONG, Baiyi; ERNEMANN, Carsten; YAHYAPOUR, Ramin: **User Group-based Workload Analysis and Modeling**. In: *Proccedings of the IEEE/ACM International Symposium on Cluster Computing and the Grid* Bd. 2, IEEE Computer Society Press, 2005, S. 953–961. – CD-ROM

[166] SONG, Hyo J.; LIU, Xin; JAKOBSEN, Dennis; BHAGWAN, Ranjita; ZHANG, Xianan; TAURA, Kenjiro; CHIEN, Andrew: **The MicroGrid: a Scientific Tool for Modeling Computational Grids**. In: *Scientific Programming* 8 (2000), S. 127–141

[167] SRINIVAS, N.; DEB, Kalyanmoy: **Multiobjective Optimization Using Nondominated Sorting Genetic Algorithms**. In: *Evolutionary Computation* 2 (1994), Nr. 3, S. 221–248

[168] SUTTON, Richard S.; BARTO, Andrew G.: *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. 1. Auflage. MIT Press, 1998

[169] TAKAGI, Tomohiro; SUGENO, Michio: **Fuzzy Identification of Systems and Its Applications to Modeling and Control**. In: *IEEE Transactions on System, Man, and Cybernetics* SMC-15 (1985), Nr. 1, S. 116–132

[170] T'KINDT, Vincent; BILLAUT, Jean-Charles: *Multicriteria Scheduling: Theory, Models and Algorithms*. 2. Auflage. Springer, 2002

[171] TONELLOTTO, Nicola; WIEDER, Philipp; YAHYAPOUR, Ramin: **A Proposal for a Generic Grid Scheduling Architecture**. In: GORLATCH, S. (Hrsg.); DANELUTTO, M. (Hrsg.): *Proceedings of the Integrated Research in Grid Computing Workshop*, Universita di Pisa, 2005, S. 337–346

[172] TROGER, Peter; RAJIC, Hrabri; HAAS, Andreas; DOMAGALSKI, Piotr: **Standardization of an API for Distributed Resource Management Systems**. In: *Proceedings of the IEEE/ACM International Symposium on Cluster Computing and the Grid*, IEEE Computer Society Press, 2007, S. 619–626

[173] TSAFRIR, Dan; ETSION, Yoav; FEITELSON, Dror G.: **Modeling User Runtime Estimates**. In: FEITELSON, D. G. (Hrsg.); FRACHTENBERG, E. (Hrsg.); RUDOLPH, L. (Hrsg.); SCHWIEGELSHOHN, U. (Hrsg.): *Proceedings of the 11th Workshop on Job Scheduling Strategies for Parallel Processing* Bd. 3834, Springer, 2005 (Lecture Notes in Computer Science), S. 1–35

[174] TSAFRIR, Dan; FEITELSON, Dror G.: **Instability in Parallel Job Scheduling Simulation: The Role of Workload Flurries**. In: *Proceedings of the International Parallel and Distributed Processing Symposium*, IEEE Computer Society Press, 2006 (Lecture Notes in Computer Science), S. 1–10

[175] TUREK, John; LUDWIG, Wolfgang; WOLF, Joel L.; FLEISCHER, Lisa; TIWARI, Prasoon; GLASGOW, Jason; SCHWIEGELSHOHN, Uwe; YU, Philip S.: **Scheduling Parallelizable Tasks to Minimize Average Response Times**. In: *Proceedings of the 6th annual ACM Symposium on Parallel Algorithms and Architectures*, ACM Press, 1994, S. 200–209

[176] WÄLDRICH, Oliver; ZIEGLER, Wolfgang; PAPASPYROU, Alexander; WIEDER, Philipp; YAHYAPOUR, Ramin: **Novel Approaches for Scheduling in D-Grid – Towards an interoperable Scheduling Framework** / Institute on Resource Management and Scheduling. 2007 (TR-0122). – CoreGRID Technical Report

[177] WASSENHOVE, Luc N.; GELDERS, Ludo F.: **Solving a Bicriterion Scheduling Problem**. In: *European Journal of Operational Research* 2 (1980), Nr. 4, S. 281–290

[178] ZADEH, Lotfi A.: **Fuzzy Sets**. In: *Information and Control* 8 (1965), Nr. 3, S. 338–353

[179] ZITZLER, E.; LAUMANNS, M.; THIELE, L.: **SPEA2: Improving the Strength Pareto Evolutionary Algorithm** / Computer Engineering and Communication Networks Lab (TIK). 2001 (TIK-Report 103). – Technical Report

Lebenslauf

Persönliche Daten

Name:	Joachim Lepping
Anschrift:	Kirchhörder Straße 236 44229 Dortmund
Geburtstag:	12. Dezember 1979
Geburtsort:	Münster (Westf.)
Nationalität:	deutsch

Ausbildung

seit 03/2006	Wissenschaftlicher Mitarbeiter am Institut für Roboterforschung, Abteilung Informationstechnik, TU Dortmund
10/2004–04/2005	Praktikum bei der Daimler AG, Stuttgart
10/2000–03/2006	Studium der Elektrotechnik, TU Dortmund (Abschluss Dipl.-Ing.)
07/1999–09/2000	Zivildienst im St.-Johannes-Hospital, Altenberge
08/1990–06/1999	Gymnasium Paulinum, Münster (Abschluss Abitur)

Dortmund, 29. März 2011