Der Titel der Arbeit

Bachlorarbeit

Fachbereich Informatik
NORDAKADEMIE

Vorgelegt von: Nachname, Vorname
Geboren am: Irgendwann in Irgendwo
Matr.-Nr.: Matrikelnummer

Gutachter:

- Dr.-Ing. Jan Himmelspach
- Zweitgutachter

Betrieblicher Betreuer:

- Betrieblicher Betreuer

Abgabedatum: 31. Januar 2011

Inhaltsverzeichnis

I Einleitung

1 Algorithm Selection Problem

Das *Algorithm Selection Problem* (ASP), bereits 1976 von John Rice formuliert [Rice, 1976], beschäftigt sich mit der Frage, welchen Algorithmus man zur Berechnung der Lösung eines Problems am ehesten benutzen sollte. Obwohl das Problem einfach beschrieben werden kann ergeben sich daraus viele schwierige Fragestellungen. Dieser Abschnitt soll einen kurzen Überblick über einige dieser Fragen geben.

Rice betrachtet unterschiedliche Mengen um das ASP zu charakterisieren (siehe Abbildung I.1). Eine davon ist die Problemmenge $\mathcal{P}$. Im Bereich der Simulation könnten unter anderem verschiedene Modelle innerhalb dieser Menge sein. Denkbar wäre zum Beispiel das Problem der Berechnung einer Zustandsüberführung eines zellulären Automaten.

Anhand dieser Probleme unterscheidet Rice nun drei Arten das ASP zu lösen. Man sucht einen Algorithmus, der entweder für alle Probleme oder für eine bestimmte Problemklasse oder nur für ein einzelnes Problem der beste Algorithmus ist. Universell beste Algorithmen zu finden ist dabei eine nicht realistisch zu lösende Aufgabe [Smith-Miles, 2009]. Ideale Algorithmen für Problemklassen zu finden ist die interessante Variante des ASP.

Schwierigkeiten bei diesem Ansatz bereitet die Definition der Problemklassen. Es müssen Eigenschaften, dargestellt in der Eigenschaftsmenge $\mathcal{E}$, gefunden werden, welche Probleme zu Klassen zusammenfassen und gleichzeitig folgendes Kriterium erfüllen: Erzielt ein Algorithmus gute Ergebnisse für ein Problem einer Problemklasse, dann muss er auch gute Ergebnisse für alle anderen Probleme dieser Klasse liefern.

Das Finden von repräsentativen Eigenschaften für Problemklassen wird von Rice als eine der am wichtigsten zu lösenden Aufgaben des ASP bezeichnet und ist heute als eigenständiges Problem unter dem Namen *Feature Selection Problem* bekannt [Kira und Rendell, 1992, Guyon und Elisseeff, 2003].

Alleine die Tatsache, dass ein Algorithmus abhängig von den Parametern eines einzelnes Problems gute oder schlechte Ergebnisse liefern kann [Roberts, 2006], unterstreicht die Schwierigkeit einer geeigneten Klassifizierung. Zudem gibt es dynamische Eigenschaften,

welche erst zur Laufzeit berechnet werden können. Wie sollen solche Eigenschaften vor der Ausführung der Algorithmen zur Konstruktion der Problemklassen benutzt werden?

Bei Simulationsmodellen scheint eine mögliche Einteilung auf den ersten Blick einfach: eine Modellart entspricht einer Problemklasse. Jedoch kann auch ein Modelltyp unterschiedliche Varianten aufweisen. Je nachdem ob zum Beispiel zelluläre Automaten deterministisch sind oder stochastische Elemente enthalten müssen Eigenschaften anders bewertet werden. Des Weiteren muss auch die Hardware, auf der das Modell ausgeführt werden soll, bei der Bewertung der Eigenschaften mit berücksichtigt werden.

Neben der Problem- und Eigenschaftsmenge muss die Algorithmenmenge $\mathcal{A}$ betrachtet werden. Im Idealfall sollte diese Menge möglichst viele Lösungsalgorithmen für eine Problemklasse enthalten, jedoch ist die Analyse dieser dann sehr großen Menge von Algorithmen zu aufwändig, sodass die Algorithmenmenge eingeschränkt werden muss. Das führt zwangsweise dazu, dass eine Art Vorauswahl getroffen werden muss. Fehler bei dieser Vorauswahl lassen sich nicht vermeiden, sodass gute Algorithmen möglicherweise von vornherein ausgeschlossen werden.

Auch bei Simulationsproblemen kann die Algorithmenmenge sehr groß sein. Ein Beispiel für einen solchen Fall ist das Modellierungs- und Simulationsframework JAMES II, welches die Möglichkeit bietet Algorithmen zu kombinieren. Durch die möglichen Kombinationen steigt die Anzahl der Algorithmen in der Algorithmenmenge stark an [Ewald et al., 2010].

Abschließend führt Rice die Notwendigkeit einer Metrik zur Bewertung der Algorithmen an. Um sie zu definieren müssen geeignete Eigenschaften und ein zweckmäßiges Verhältnis für die Bewertung zwischen diesen Eigenschaften gefunden werden. Beispiele für solche Eigenschaften sind die Laufzeit- und Platzkomplexität, die Optimalität eines Ergebnisses oder die Einfachheit eines Algorithmus. Die erhaltenen Ergebnisse der Leistungsmessung können dann im Performanceraum $\mathcal{Y}$ abgebildet werden, welcher die Leistung eines Algorithmus für ein bestimmtes Problem bestimmt.

2 Algorithmenportfolios

Der Ansatz der Algorithmenportfolios stammt ursprünglich aus der Ökonomie im Zusammenhang mit der Streuung von Investitionen. Denn obwohl die Investition in eine einzige viel versprechende Anlage hohe Gewinne erwarten lässt, birgt sie auch das Risiko eines hohen Verlustes, sollte sich die Anlage als Fehlschlag herausstellen. Um das Risiko eines solches Kapitalverlustes zu senken werden Portfolios eingesetzt. Für ein Gesamtvo-

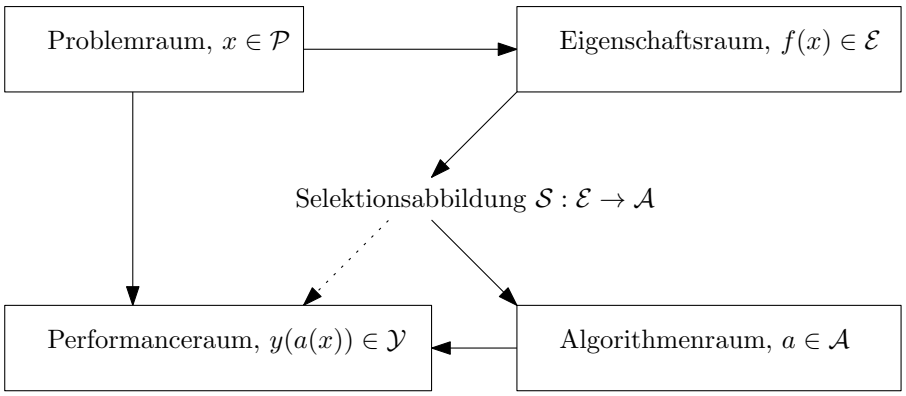

Abbildung I.1: Charakterisierende Mengen für das *Algorithm Selection Problem*. Eine optimale Selektionsabbildung $\mathcal{S}$ ist gesucht, welche folgende Eigenschaft erfüllt: $a = S(f(x)) \Rightarrow y(a(x)) = max(y(\alpha(x)) \; \forall \alpha \in \mathcal{A}$

lumen $\mathcal{F}$ an Finanzmittel und einer Menge von Anlagen $(a_1, a_2, ..., a_n)$ werden die Investitionen $(x_1, x_2, ..., x_n)$, $\sum_{i=1}^{n} x_i = \mathcal{F}, x_i \in \mathbb{Q}$ mithilfe eines Verteilungsvektors $\vec{v}$ mit $\sum_{i=1}^{n} v_i = 1, v_i \in \mathbb{Q}$ gestreut. Hauptproblem dabei ist es einen Kompromiss zwischen erwartetem Gewinn und Risiko, d. h. der Varianz des erwarteten Gewinns, zu finden [Markowitz, 1952].

Die Auswahl eines Portfolios kann in zwei Phasen unterteilt werden. Zum einen in eine Beobachtungsphase, in der der Markt betrachtet wird. Zum anderen in eine Analysephase, in der Prognosen über die Zukunft anhand der gemachten Beobachtungen berechnet werden. Die erstellten Prognosen dienen dann als Grundlage um die Portfolios zu finden, die einen optimalen Kompromiss zwischen Gewinn und Risiko bieten. Der optimale Kompromiss ist jener, der für einen bestimmten Erwartungswert die niedrigste Varianz bzw. für eine bestimmte Varianz den höchsten Erwartungswert liefert. Durch diese Betrachtung ergeben sich mehrere optimale Portfolios. Sie liegen auf der von Markovitz 'efficient frontier' benannten Linie (siehe Abbildung I.2). Neben diesen beiden Kriterien müssen aber auch weitere Punkte beachtet werden. Beispielsweise ist auf eine Vielfalt der gewählten Anlagenbereiche zu achten, da ein einseitiges Portfolio gegen einen Einbruch der gewählten Branche anfällig wäre.

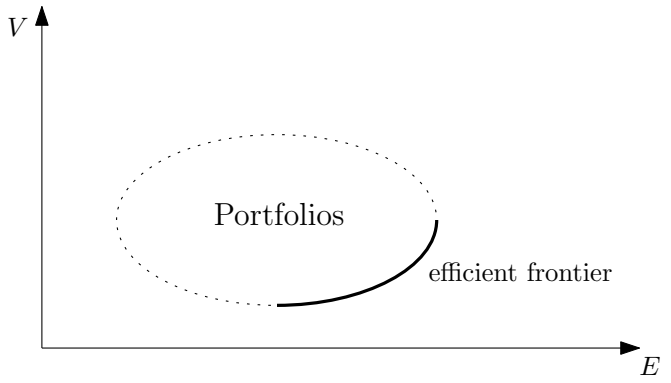

Abbildung I.2: Innerhalb der Ellipse liegen alle möglichen Portfolios. Der markierte Rand der Ellipse ist die 'efficient frontier' auf der alle optimalen Portfolios liegen.

3 Simulationsprobleme

Um Aussagen über Modelle treffen zu können müssen viele Experimente mit gleichen, aber auch mit verschiedenen Parameterkombinationen durchgeführt werden. Jede dieser Parameterkombination wird in einer sog. Replikation benutzt. Diese Replikationen müssen mehrmals ausgeführt werden, da die Durchführung stochastisch sein kann. Anders als bei komplexitätstheoretisch schwierigen Problemen ergibt sich bei Simulationsalgorithmen der größte Teil der Laufzeit- und Platzkomplexität anhand der Modellgröße, der geforderten Replikationen und der Simulationslaufzeit. Als Folge der sich oft wiederholenden Berechnungsarten ist es möglich mit geringen Performanceverbesserungen bei einzelnen Algorithmen eine spürbare Verbesserung der Laufzeit des Gesamtsystems zu bewirken.

Durch die große Anzahl an Modellen und das unvorhersehbare Verhalten bei verschiedenen Parameterkombinationen in Experimenten, dazu den vielen verschiedenen Algorithmen um Experimente auf Modellen auszuführen kann meistens nicht vorhergesagt werden, welcher Algorithmus die besten Ergebnisse erzielen würde. Aus diesem Grund ist eine Analyse der Algorithmenportfolios und ihrer Theorie im Zusammenhang mit Simulationsproblemen von Interesse, damit Leistungsänderungen durch Anwendung von Portfolioansätzen besser einschätzt werden können.

Im folgenden werde ich mich bei der Betrachtung von Algorithmenportfolios im Zusammenhang mit Simulationsproblemen meistens auf zelluläre Automaten beziehen, da sich mit ihnen die meisten Sachverhalte einfach und anschaulich beschreiben lassen. Weitere Informationen bzgl. zellulärer Automaten und Simulationsalgorithmen bietet [Rybacki et al., 2009].

II Algorithmenportfolios

1 Allgemeines

Obwohl Rice das ASP schon 1976 vorstellte und die Formalisierung des Ökonomieportfolios mehr als 50 Jahre alt ist, wurde die Idee der Algorithmenportfolios erst im Zusammenhang mit Restartstrategien 1997 das erste Mal betrachtet [Huberman et al., 1997]. Durch Restartstrategien werden Algorithmen neugestartet, sobald bestimmte Kriterien erfüllt sind (siehe Abschnitt 1.1).

Algorithmenportfolios bieten Möglichkeiten mit den Schwierigkeiten des ASP umzugehen. Grundlegender Gedanke dabei ist es nicht nur einen einzelnen Algorithmus für die Berechnung von Probleminstanzen auszuwählen, sondern eine Untermenge der verfügbaren Algorithmen. Wichtig ist zu erwähnen, dass ein Algorithmus mehrmals in einem Portfolio vorkommen kann. Die ausgewählten Algorithmen sollen dann für möglichst viele Problemklassen gute Leistungen erzielen und eine kleine Varianz in der Laufzeit zwischen Probleminstanzen des gleichen Typs liefern.

Beide Kriterien, Laufzeit und Varianz, sind ähnlich den beiden Kriterien der Ökonomieportfolios, erwartetem Gewinn und Varianz dessen. Jedoch gibt es auch gravierende Unterschiede, weshalb sich die Anwendung der Theorie der Portfolios aus der Ökonomie auf die Algorithmenportfolios teilweise schwierig gestaltet [Gomes und Selman, 2001]. Zur Verdeutlichung seien an dieser Stelle zwei Schwierigkeiten erwähnt. Zum einen muss man sich bei Algorithmenportfolios eine Ausführungsart der Algorithmen überlegen, obgleich sich der Markt in der Wirtschaft ohne eigenes Zutun weiterentwickelt. Und zum anderen summieren sich Gewinne und Verluste von Anlagen eines Portfolios, während mehrere Algorithmen mit guten Laufzeiten für die gleiche Probleminstanz keinen Mehrwert gegenüber einem einzigen guten Algorithmus im Portfolio für diese Probleminstanz haben.

Ein Problem bei der allgemeinen Betrachtung von Algorithmenportfolios anhand der vorhandenen Forschung ist vielfach die Spezialisierung der Analysen auf einzelne Problemtypen, welche meist Entscheidungs- oder Optimierungsprobleme sind. So wird beispielsweise in [Fukunaga, 2000] das *Traveling Salesman Problem* (TSP) zur Evaluierung her-

angezogen, das *Mixed Integer Problem* (MIP) in [Gomes und Selman, 2001] und oft das *Erfüllbarkeitsproblem* (SAT) wie zum Beispiel in [Kautz et al., 2002]. Aus diesem Grund müssen viele Ansätze und Methoden kritisch betrachtet werden, da sie keiner allgemeineren Evaluierung unterzogen wurden.

1.1 Restartstrategien

Stochastische Algorithmen können mit verschiedenen Parametern und Startwerten für Zufallszahlengeneratoren unterschiedliche Leistungen erbringen. Eine Ursache dafür kann sein, dass ein Algorithmus für die Berechnung der Lösung eines Problems den Suchraum stochastisch verkleinert. Ist eine Lösung innerhalb dieses kleineren Suchraums kann sie nun schneller gefunden werden. Ist keine Lösung im verkleinerten Suchraum sind die Berechnungen für diesen Suchraum überflüssig. Werden wiederholt Suchräume ohne Lösung ausgewählt verlängert sich dementsprechend die Laufzeit. Variiert die Leistung des Algorithmus stark, seien solche Algorithmen an dieser Stelle als riskant, ansonsten als sicher bezeichnet. Riskante Algorithmen haben den Vorteil bei guten Läufen meistens wesentlich bessere Ergebnisse zu erzielen als sichere Algorithmen. Belegt ein SAT Algorithmus beispielsweise einige Formelvariablen zufällig mit Werten und es stellen sich diese als Werte einer potentiellen Lösung dar, wird der Suchraum, wie oben beschrieben, verkleinert und es kann schneller eine Lösung gefunden werden. Führen die zufälligen Belegungen jedoch zu keiner Lösung hat dieser Teil der Suche nur unnötige Kosten verursacht. Um solche Verfahren zu systematisieren werden beispielsweise Heuristiken eingesetzt [Brélaz, 1979, Gomes und Selman, 2001].

Ursprünglich wurden Restartstrategien verwendet um Algorithmen, welche bei lokalen Maxima festhingen, davon loszulösen [Kautz et al., 2002]. Mittlerweile verfolgen Restartstrategien hauptsächlich das Ziel stochastische Algorithmen neuzustarten, sobald sich erkennen lässt, dass ein ungünstiger Lauf durchgeführt wird. Besonders effektiv ist diese Methode bei 'heavy-tailed' Laufzeitverteilungen (siehe Abbildung II.1). Die Annahme dahinter ist, dass die Wahrscheinlichkeit mit der ein stochastischer Algorithmus eine Lösung zum aktuellen Zeitpunkt findet mit zunehmender Laufzeit sinkt, sodass ein Neustart ab einem bestimmten Zeitpunkt effektiver ist.

Der optimale Zeitpunkt für den Neustart zur Minimierung der erwarteten Laufzeit kann berechnet werden, sofern komplettes Wissen über die Laufzeitverteilung vorhanden und die Laufzeit das einzige Kriterium zur Bewertung des Algorithmus ist [Kautz et al., 2002]. Im

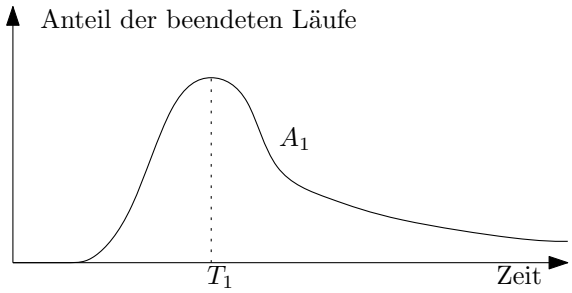

Abbildung II.1: Zum Zeitpunkt T_1 terminieren die meisten Läufe des Algorithmus A_1. Mit zunehmender Zeit nimmt diese Anzahl nur langsam ab, sodass einige Ausführungen eine sehr lange Laufzeit haben. Für solch eine Ausführung ist ein Neustart oft effektiver als den Algorithmus terminieren zu lassen.

Allgemeinen ist dieses Wissen nicht verfügbar. Daher gibt es einige Methoden um dieses Problem zu umgehen. Eine Möglichkeit ist es eine Trainingsphase durchzuführen, in der verschiedene Zeiten und verschiedene Probleme benutzt werden um einen guten Zeitpunkt zu berechnen. Der gefundene Zeitpunkt kann dann durch dynamisches Anpassen zwischen Problemberechnungen oder direkt nach einem Neustart weiter verbessert werden. Diese Anpassung ist bei einer Trainingsphase mit wenigen Problemlösungen sehr effektiv.[1] Sie ermöglicht jedoch keine prinzipiell besseren Ergebnisse als eine Trainingsphase mit vielen Problemlösungen [Fukunaga, 2000].

Eine interessante Erkenntnis dieser Lernmethoden ist, dass sogar bessere Ergebnisse als mit dem optimalen Zeitpunkt erzielt werden können [Kautz et al., 2002]. Das hängt damit zusammen, dass der optimale Zeitpunkt mathematisch mithilfe der Wahrscheinlichkeitsverteilung für die Terminierung eines Algorithmus berechnet wird. Diese Verteilung tritt bei einer konkreten Durchführung natürlich nicht genau in dieser Form auf, sodass der optimale Zeitpunkt für einen konkreten Fall nicht mit dem allgemeinen optimalen Zeitpunkt übereinstimmen muss.

Die Idee der Restartstrategien kann auch auf Algorithmenportfolios übertragen werden. Wichtig dabei ist, dass ein Algorithmus mehrmals in einem Portfolio vorkommen kann. Dieser wird dann bei der Durchführung parallel mit unterschiedlichen Parametern und/oder Anfangswerten für Zufallszahlengeneratoren ausgeführt. Folglich kann im Extremfall ein Algorithmenportfolio nur aus einem Algorithmus bestehen. In [Gomes und Selman, 2001] wird dieses Szenario genauer analysiert. Es werden 2 Algorithmen, einer davon riskant und einer sicher, zur Lösung des MIP als Kandidaten für ein Algorithmenportfolio betrachtet.

[1] In [Gagliolo und Schmidhuber, 2007] werden bis zu sechs mal bessere Ergebnisse mit zusätzlicher Anpassung des Zeitpunkts erzielt.

Zudem werden vier Größen (2, 5, 10, 20) des Portfolios untersucht. Das Ergebnis dieser Arbeit ist folgendes: Je mehr Algorithmen im Portfolio vorhanden sind, desto besser ist es wenn nur der riskante Algorithmus mehrfach vorhanden ist. Die Erklärung ist naheliegend. Wird der riskante Algorithmus öfter ausgeführt steigt die Wahrscheinlichkeit einen guten Lauf unter den Ausführungen zu erhalten.

Im Zusammenhang mit Simulationsproblemen sind Restartstrategien nur bedingt sinnvoll einsetzbar. Zum einen haben Simulationsalgorithmen nicht immer eine 'heavy-tailed' Laufzeitverteilung. Zum anderen müssen sie auch nicht stochastisch sein, sodass ein Neustart nur gleiche neue Berechnungen verursachen würde. Auf Grund dessen muss über den Einsatz von Restartstrategien im Einzelfall entschieden werden. Darüber hinaus muss folgendes Szenario bedacht werden: Benötigt ein Lauf eines Simulationsalgorithmus viel Laufzeit, kann es daran liegen, dass dieser einen unwahrscheinlichen und aufwändigen Verlauf des Modells berechnet. Ein Neustart würde diesen Verlauf dann aus den beobachteten Durchläufen löschen. Das würde bedeuten, dass am Ende der Simulation kein aufwändiger Verlauf berechnet wurde und die Endergebnisse verfälscht und unbrauchbar sind.

1.2 Auswahl der Algorithmen

Das Auswahlverfahren der Algorithmen für ein Portfolio ist oft nicht näher erläutert oder dem Benutzer überlassen. In dem viel referenzierten Werk von Schmidhuber und Gagliolo [Gagliolo und Schmidhuber, 2006] heißt es: „At the lowest level, the choice of the algorithms composing $\mathcal{A}$ [...] is still left to the user“. In [Fukunaga, 2000] wird ein einfacher Algorithmus zur Konstruktion von Portfolios angegeben. Dieser benötigt jedoch eine große Datenbasis um zweckmäßige Ergebnisse zu liefern und kann nur bei Portfolios angewendet werden die eine konstante Laufzeit T haben. In der Datenbasis werden die einzelnen Leistungen der Algorithmen nach T Zeiteinheiten für eine gewisse Anzahl an Problemen gespeichert und zur Berechnung des nach diesen Daten optimalen Portfolios genutzt. In Folge dessen ist das gefundene Portfolio letztendlich für die bereits gelösten Probleme optimal, jedoch nicht zwangsweise für weitere Probleme. Beide Kriterien für diesen Konstruktionsalgorithmus werden im Allgemeinen nicht erfüllt.

Des Weiteren ist die Anzahl der Algorithmen in den betrachteten Portfolios meistens kleiner als 10 [Fukunaga, 2000, Gomes und Selman, 2001]. Ob sich gute Ergebnisse auch mit einer großen Anzahl von Algorithmen erzielen lassen bleibt unklar.

Im Zusammenhang mit Restartstrategien haben Gomes und Saleman [Gomes und Selman, 2001] gezeigt, dass es im Allgemeinen besser ist Algorithmen in das Portfolio aufzunehmen,

welche für wenige Probleme sehr gute Ergebnisse statt für viele Probleme durchschnittliche Ergebnisse erzielen.

Letztendlich ist aus den genannten offenen Fragen auch im Zusammenhang mit Simulationsproblemen noch Forschung notwendig um herauszufinden, wie gute Portfolios konstruiert werden sollten. Die zuletzt genannte Annahme von Gomes und Saleman sollte sich auch auf Simulationsalgorithmen übertragen lassen. Auf zelluläre Automaten übertragen, würde man solche Algorithmen nehmen, welche in bestimmten Situationen sehr gute Leistungen erbringen. Ist das Feld eines deterministischen zellulären Automaten beispielsweise hauptsächlich homogen und wenig Zellen ändern ihren Zustand bei einer Zustandsüberführung, wären Algorithmen im Vorteil, welche die sich nicht ändernden Flächen als solche erkennen und für entsprechende Zustandsüberführungen nicht weiter betrachten. Wechseln dagegen viele Zellen ihren Zustand ist womöglich ein Algorithmus besser, welcher immer das gesamte Feld betrachtet. Solche speziellen Algorithmen sind, sofern sie ihre Struktur optimal einsetzen können, allgemeineren Algorithmen überlegen. Folglich sollte ein Portfolio die spezielleren Algorithmen enthalten, wenn mindestens ein Algorithmus für jede Situation sehr gute Leistung erbringen wird.

1.3 Problemtypen und Leistungsbewertung

Zwei grundlegende Problemtypen werden bei den meisten Arbeiten unterschieden: Entscheidungs- und Optimierungsprobleme [Gagliolo und Schmidhuber, 2006, Gomes und Selman, 2001]. Des Weiteren wird davon ausgegangen, dass alle Algorithmen terminieren. Während nun Algorithmen für Entscheidungsprobleme ein eindeutiges Ende besitzen, sobald sie eine Lösung gefunden oder die Unlösbarkeit des Problems bewiesen haben, ist es bei Optimierungsproblemen nicht so eindeutig. Man muss sich überlegen nach welchen Kriterien man die Berechnung abbrechen möchte. Entweder ab dem Zeitpunkt zu dem eine an dem Optimum genügend dicht liegende Lösung gefunden wurde oder nach einer bestimmten Zeit oder beides (siehe Abbildung II.2). Je nach Entscheidung muss sich das Portfolio anpassen. Wenn man die Berechnung beispielsweise durch eine Zeit begrenzt, dann müssen die Algorithmen in dieser Zeit so gut wie möglich vorankommen. Dann ist es irrelevant, ob ein Algorithmus zu einem späteren Zeitpunkt wesentlich bessere Ergebnisse als alle anderen liefern würde. Setzt man dagegen ein Qualitätsminimum könnte genau dieser Algorithmus der einzige sein, welcher dieses Minimum je erreicht.

In der Simulation treten beide Problemtypen in dieser standardisierten Form nicht auf. Beispielsweise die Zustandsüberführung eines zellulären Automaten ist weder ein Entschei-

dungs- noch ein Optimierungsproblem. Nichtsdestotrotz kann man die beiden Kriterien der Laufzeit und Qualität einer Berechnung auch für Simulationsprobleme nutzen. Jedoch ist eine einfache Übernahme der theoretischen Erkenntnisse über Entscheidungs- und Optimierungsprobleme im Zusammenhang mit Algorithmenportfolios nicht möglich.

Ein weiterer wichtiger Bestandteil bei der Arbeit mit Algorithmenportfolios ist die Bewertung ihrer Leistung. In [Gomes und Selman, 2001] wird ein Kompromiss zwischen erwarteter Laufzeit und dessen Varianz zur Lösung eines Problems als Maßstab herangezogen. Bei Optimierungsproblemen kann zusätzlich noch die Qualität der Lösungen eine Rolle spielen. Ein weiteres Kriterium kann die Vorhersagegenauigkeit des Portfolios sein. Damit ist gemeint, ob die vorhandenen Ressourcen auch zum größten Teil dem besten Algorithmus zur Verfügung gestellt werden. Probleme bereiten dynamische Portfolios, die ihre Leistung mit zunehmender Anzahl an gelösten Problemen verbessern und daher von Beginn an unter Berücksichtigung ihres Potentials anders bewertet werden müssen (siehe Abschnitt 3.2).

Neben den bisher genannten Kriterien der Laufzeit und Qualität müssen bezüglich der Laufzeit bei Simulationsalgorithmen weitere Sachverhalte beachtet werden. Man sollte zur Bewertung eines Portfolios nicht nur die physische Laufzeit, sondern auch Simulationszeit heranziehen. Muss ein Algorithmus beispielsweise mehr Events als ein anderer in derselben Simulationszeit ausführen, wäre ein direkter Vergleich der physikalischen Zeit ungerecht. In Folge dessen muss auch die Anzahl der ausgeführten Events mit berücksichtigt werden. Nicht zu vergessen mögliche unterschiedliche Berechnungsaufwände von Events. Bei zellulären Automaten zum Beispiel wäre eine Zustandsüberführung eines beinahe homogenen Feldes, bei dem sich nur einige Zellen ändern einfacher zu berechnen als die Zustandsüberführung eines Feldes, bei dem sich fast alle Zellen ändern. Eine analoge Situation wie in Abbildung II.2 kann auch auf zelluläre Automaten übertragen werden, sofern sie diskret ereignisorientiert sind (siehe Abbildung II.3).

1.4 Kooperation von Algorithmen

Obwohl bereits in einer der ersten Arbeiten über Algorithmenportfolios Leistungsverbesserungen durch Kooperation von Algorithmen beschrieben wurden [Huberman et al., 1997], wird in vielen Artikel lediglich der sogenannte Blackboxansatz verfolgt [Gagliolo und Schmidhuber, 2006, Fukunaga, 2000, Gomes und Selman, 2001]. Bei diesem wird die konkrete Implementierung der Algorithmen im Portfolio nicht näher betrachtet und eine Kommunikation zwischen ihnen erfolgt nicht. Der Vorteil dabei ist, dass die Strategien für diese Portfolios auf neue Algorithmen übertragen werden können und keine Anpassungen bei

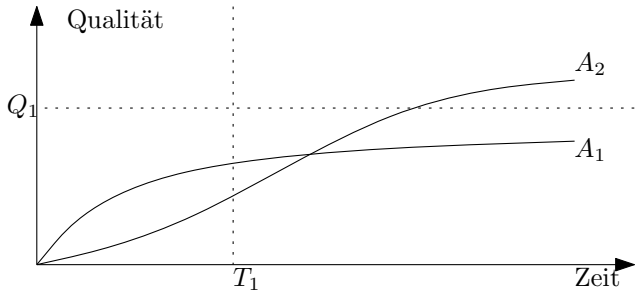

Abbildung II.2: Je nachdem wie die Leistung eines Algorithmus bewertet wird ändert sich der zu wählende Algorithmus. Wenn die Algorithmen beispielsweise nur T_1 Zeiteinheiten laufen wäre A_1 die beste Wahl. Die Qualitätsgrenze Q_1 würde jedoch A_2 eher erreichen.

neuen Problemklassen vorgenommen werden müssen. Der Nachteil ist jedoch der Verlust der Möglichkeit, Wissen zwischen Algorithmen auszutauschen und Lösungsstrukturen auszunutzen. Grundlegendes Problem beim Austausch von Wissen zwischen Algorithmen ist die meist unterschiedliche Darstellung von Zwischenergebnissen, sodass Transformationen dieser Ergebnisse durchgeführt werden müssten um sie für andere Algorithmen nutzbar zu machen. Dabei kann nicht garantiert werden, dass so eine Transformation überhaupt möglich ist und zudem sind solche Transformationen oft mit einem hohen Aufwand verbunden [Roberts, 2006]. Auf der einen Seite viel menschlicher Aufwand in der Erforschung möglicher Transformationen und zum anderen ein hoher Berechnungsaufwand. Eine weitere Frage ist, wie unterschiedliche Zwischenergebnisse von Algorithmen kombiniert werden können. Bei SAT beispielsweise kann eine Formel mehrere Lösungen besitzen, welche sich gegenseitig ausschließen. Haben zwei Algorithmen nun jeweils eine andere dieser Lösungen zum Teil konstruiert, können Zwischenergebnisse beider Algorithmen gegenseitig nicht genutzt werden.

Die genannten Probleme treten im allgemeinen auch bei Simulationsalgorithmen auf. Für zelluläre Automaten existieren mehrere Darstellungsformen. Zum Beispiel Zellen als Objekte oder Zustände oder den ganzen Automaten als Bild darzustellen. Aufgrund einer wahrscheinlich sehr großen Anzahl an Zellen könnten Umformungen zwischen diesen Repräsentationen viel Zeit in Anspruch nehmen. Bei gleicher Datenrepräsentation wäre eine Anwendung von Kooperationsverfahren zweckmäßiger. Eine Bewertung solcher Verfahren wäre in dem Fall durchaus von Interesse.

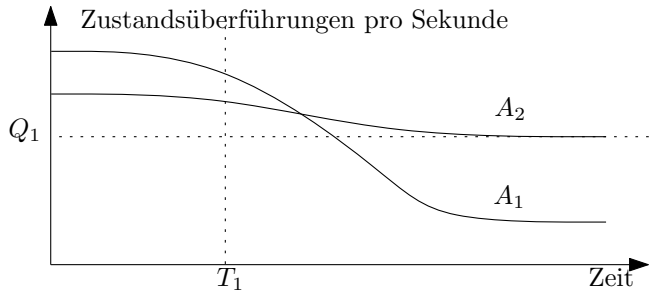

Abbildung II.3: Angenommen das Startfeld eines diskret ereignisorientierten zellulären Automaten ist hauptsächlich homogen und die Anzahl der Zellenänderungen nimmt mit jedem Zustandsübergang zu. A_1 könnte ein Algorithmus sein der auf wenig Zellenänderungen in einem großen Automaten spezialisiert ist. Nimmt nun die Anzahl der Zellenänderungen zu sinkt die Performance. A_2 hätte eine konstantere Leistung beispielsweise durch komplexe Datenstrukturen. Berechnet man nun für T_1 Zeiteinheiten den Automaten ist Algorithmus A_1 die bessere Wahl, ohne Zeitbegrenzung ist es allerdings A_2. Interessant wäre auch der Fall, wenn der Automat mit der Zeit wieder einen homogenen Zustand erreicht und dann wieder A_1 die bessere Performance aufweist. Das würde die Auswahl weiter verkomplizieren.

2 Ausführungsarten

Dieser Abschnitt beschäftigt sich mit der Art der Ausführung der Algorithmen innerhalb eines Portfolios. Es werden dabei drei Typen unterschieden. Die parallele, die verschränkte und die einzelne Ausführung.

2.1 Parallele Ausführung

Bei der parallelen Ausführung werden die Algorithmen des Portfolios echt parallel auf mehreren Recheneinheiten ausgeführt (siehe Abbildung II.4). Beispielsweise Gomes und Selman [Gomes und Selman, 2001] verfolgen diesen Ansatz in Kombination mit einer Restartstrategie. Durch die parallele Ausführung müssen vorhandene Ressourcen einer Recheneinheit nicht geteilt werden, sodass der schnellste Algorithmus ohne Behinderung seine Leistung voll einbringen und ein gutes Ergebnis erzielen kann. Dazu werden natürlich genug Recheneinheiten benötigt. Des weiteren können unnötig laufende Algorithmen keine Ressourcen abgeben, sodass viele Algorithmen trotz einer erwarteten schlechten Laufzeit entweder umsonst weiter ausgeführt werden oder die entsprechenden Recheneinheiten durch einen sinnvollen Abbruch untätig sind.

Parallelität spielt bei Simulationen eine große Rolle [Ewald et al., 2010, Rybacki et al.,

2009]. Modelle werden geteilt um sie parallel bearbeiten zu können. Replikationen werden parallel ausgeführt. Es gibt viele weitere Möglichkeiten allerlei Berechnungen zu parallelisieren. Die Frage dabei ist, ob eine weitere Parallelisierung bezüglich eines Algorithmenportfolios noch zweckmäßig wäre. Des Weiteren müssten langsame Algorithmen bei Simulationen, in denen jeder Verlauf in die Endergebnisse mit einfließen muss, weiter ausgeführt werden. Dann ist das Portfolio jedoch nichts anderes als eine Parallelisierung mehrerer Ausführungen einer Replikation, sofern das Portfolio nicht geändert wird. Eine Möglichkeit das zu tun wäre es beispielsweise langsame Algorithmen nach einer Durchführung aus dem Portfolio zu entfernen [Ewald et al., 2010].

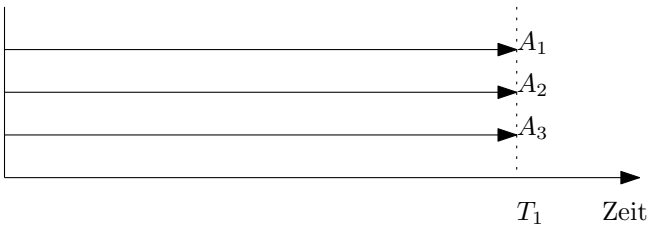

Abbildung II.4: Die Algorithmen des Portfolios werden lediglich parallel ausgeführt und sobald einer von ihnen terminiert werden alle beendet.

2.2 Verschränkte Ausführung

Um mit einer einzelnen Recheneinheit mehrere Algorithmen parallel auszuführen wird die verschränkte Ausführung benutzt [Gagliolo und Schmidhuber, 2006]. Die Algorithmen werden nicht echt parallel, sondern abwechselnd innerhalb eines konstanten Zeitraums ausgeführt (siehe Abbildung II.5). Dieser Vorgang wiederholt sich dann solange bis ein Algorithmus terminiert oder ein anderes Terminierungskriterium erfüllt ist. Dabei gibt es mehrere Faktoren zu beachten. Zum einen die Dauer einer Durchführung aller Algorithmen und zum anderen die Verteilung dieser Zeit auf die einzelnen Algorithmen. Verschiedene Möglichkeiten die Ressourcenverteilung anzupassen werden im Kapitel über die Entwicklung von Algorithmenportfolios näher betrachtet. Interessant ist die Tatsache, dass sich diese Art der Ausführung auch parallel mit mehreren Recheneinheiten lohnt [Gagliolo und Schmidhuber, 2008].

Bezüglich Simulationsalgorithmen ist diese Variante interessant zu Betrachten. Zum einen benötigt sie nur eine Recheneinheit und teilt sich Ressourcen selbst ein. Sollten weitere Recheneinheiten zur Verfügung stehen können sie ohne Probleme mit genutzt werden,

notwendig sind sie aber nicht. Zum anderen benötigt die verschränkte Ausführung kein umfassendes Wissen über die Laufzeitverteilungen der einzelnen Algorithmen. Auch ohne dieses Wissen können gute Ergebnisse erzielt werden.

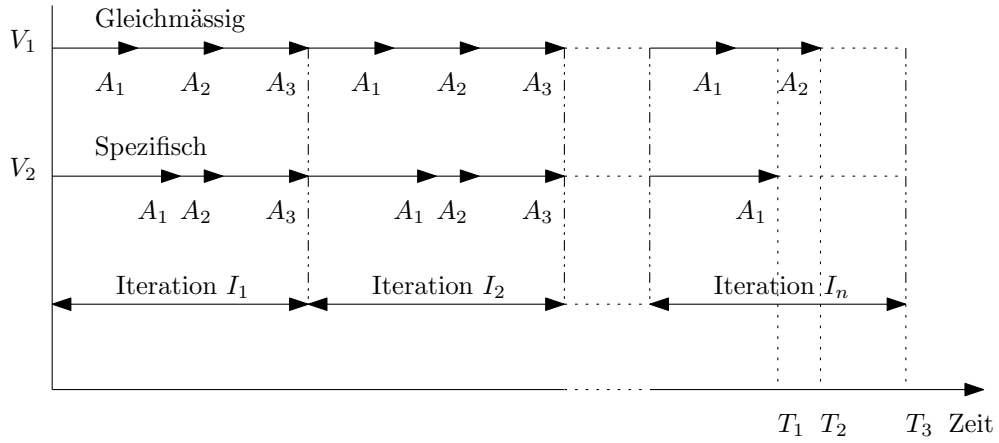

Abbildung II.5: Veranschaulichung der verschränkten Ausführung eines Portfolios mit drei Algorithmen. Innerhalb einer Iteration werden alle Algorithmen einen bestimmten Anteil lang ausgeführt. Bei der Gleichbehandlung (Verlauf V_1) der Algorithmen terminiert A_2 nach n Iterationen und die Ausführung wird beendet. Wird A_1 jedoch bevorzugt (Verlauf V_2) terminiert dieser zuerst nach n Iterationen. Obwohl diese Variante schneller ist, wäre es besser A_2 mehr Ressourcen zur Verfügung zu stellen, da noch bessere Ergebnisse erzielt werden könnten.

2.3 Einzelne Ausführung

Die einzelne Ausführung ist im wesentlichen nichts anderes als das ASP angewendet auf das Portfolio (siehe Abbildung II.6). Es wird ein Algorithmus anhand erstellter Leistungsdaten für eine Probleminstanz ausgewählt und nur dieser wird dann zur Problemlösung ausgeführt. Der Vorteil gegenüber dem allgemeinen ASP ist die kleinere Menge an Algorithmen im Portfolio. Zusätzlich ist eine Wettbewerbsphase möglich. In dieser werden alle Algorithmen ähnlich wie bei der verschränkten Ausführung bis zu einem bestimmten Zeitpunkt ausgeführt. Die aktuelle Leistung der Algorithmen wird dann verglichen und der beste wird zur alleinigen Ausführung ausgewählt. Die nach [Roberts, 2006] am meisten verwendeten Faktoren zur Berechnung der Leistung sind das aktuelle Zwischenergebnis, bei Simulationen beispielsweise der aktuelle Zustand eines zellulären Automaten, die vorhergesagte Restlaufzeit und die bisherige Auswahlrate des Algorithmus.

Es ist naheliegend, dass dieser Ansatz mit genügend Leistungsdaten der Effektivste ist. Jedoch sind genau diese Leistungsdaten meistens nicht vorhanden. Das ist auch bei Simulationsalgorithmen nicht anders. Ein Vorteil dieser Ausführung ist, dass jeder Lauf bis zum Ende ausgeführt wird und in der Ergebnisanalyse berücksichtigt werden kann.

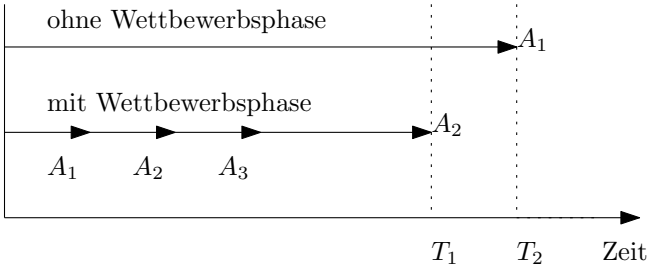

Abbildung II.6: Beispiel für die einzelne Ausführung eines Portfolios mit drei Algorithmen. Ohne Wettbewerbsphase wird direkt ein Algorithmus für die Problemlösung ausgewählt, in dem Fall A_1. Nach einer Wettbewerbsphase der drei vorhandenen Algorithmen würde jedoch A_2 als der erfolgversprechendere Kandidat erkannt. Dieser ist dann so performant, dass trotz Wettbewerbsphase ein besseres Ergebnis als bei der alleinigen Ausführung von A_1 erreicht wird.

3 Entwicklung eines Algorithmenportfolios

Bei der Entwicklung von Portfolios geht es hauptsächlich um eine Neuverteilung der verfügbaren Ressourcen während der Konstruktion und Benutzung eines Portfolios (siehe Abbildung II.7). Grundsätzlich muss zwischen der Entwicklung des Portfolios an sich und der Entwicklung der einzelnen Algorithmen unterschieden werden. Auf die Entwicklung von Algorithmen soll an dieser Stelle nicht näher eingegangen werden. Es sei lediglich erwähnt, dass lernfähige Algorithmen zusätzliche Redundanzen in der Datenbasis schaffen könnten und dann eine unnötige Belastung für das Portfolio erzeugen [Roberts, 2006]. Zudem muss bei einem Portfolio mit lernfähigen Algorithmen immer darauf geachtet werden, dass mit einer zunehmenden Anzahl von Problemlösungen eine Erhöhung der Leistung nicht nur durch eine Verbesserung des Portfolios, sondern auch durch die verbesserten Algorithmen zustande kommt.

Vor allem die verschränkte Ausführung von Algorithmen findet bei der Analyse der Entwicklungsstrategien große Beachtung. Parallele Portfolios können eine Verteilung der verfügbaren Ressourcen nicht ändern und Auswahlportfolios können auf die Theorie des ASP und in der Wettbewerbsphase auf die Ergebnisse der verschränkten Portfolios zurückgreifen.

Die Entwicklung von Algorithmenportfolios lässt sich grob in zwei Phasen unterteilen. Zum einen in eine Trainingsphase, in der Lösungen für repräsentative Probleminstanzen $R_1, ..., R_n$ berechnet werden. Sowohl repräsentative Probleminstanzen für die Trainingsphase, als auch eine optimale Anzahl jener zu finden sind schwierige Aufgaben [Gagliolo und Schmidhuber, 2006]. Mithilfe der Ergebnisse der Trainingsphase wird eine Datenbasis erstellt, welche für die zweite Phase, die Ausführungsphase genutzt wird. In dieser werden Lösungen für alle weiteren auftretenden Probleme berechnet.

Die besten Ergebnisse werden bei der Entwicklung von Portfolios erzielt, in denen jeweils ein dominierender Algorithmus für eine Probleminstanz existiert [Gomes und Selman, 2001, Gagliolo und Schmidhuber, 2006]. Diese werden ab einem bestimmten Punkt den Großteil der Ressourcen zur Verfügung haben. Existieren mehrere ähnliche Algorithmen werden sich die Ressourcen ungünstigerweise gleichmäßig unter ihnen aufteilen.

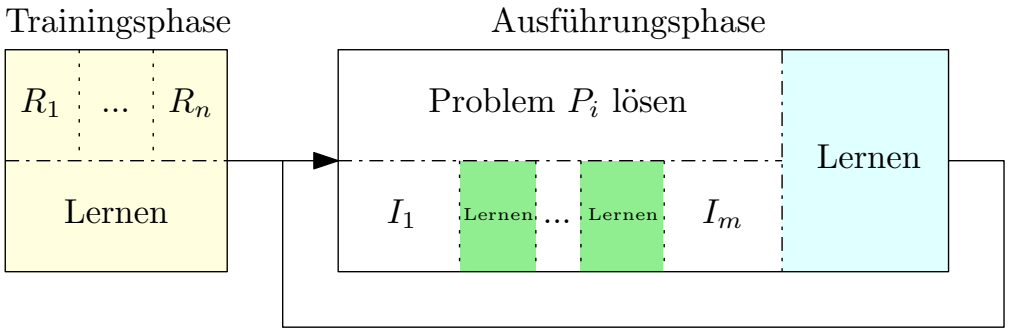

Abbildung II.7: Überblick über die Entwicklungsarten eines Portfolios. In der Trainingsphase (gelb) werden n repräsentative Probleme R_1 bis R_n gelöst und eine Datenbasis wird anhand der Ergebnisse erstellt. Diese Datenbasis wird in der Ausführungsphase eingesetzt und bei einer Offlineentwicklung (blau) nach jeder Problemlösung durch weiteres Lernen verbessert. Bei der Onlineentwicklung (grün) wird zusätzlich zwischen jeder Iteration innerhalb einer Problemlösung die Datenbasis für dieses jeweilige Problem angepasst.

3.1 Statische Entwicklung

Die statische oder vergessliche [Gagliolo und Schmidhuber, 2006] Entwicklung hält sich an die klare Trennung von Trainings- und Ausführungsphase. Es wird zwischen der Berechnung einzelner Probleminstanzen keine Veränderung der Strategie des Portfolios vorgenommen. Ein Nachteil dabei ist, dass die Trainingsphase viele Probleme umfassen muss um gute

Ergebnisse zu liefern und dennoch wird das System unflexibel gegenüber neuartigen Problemen sein. Ein Lösungsansatz zur Vermeidung solch einer umfangreichen Trainingsphase ist es Ungenauigkeiten in der Datenbasis zu akzeptieren. Beispielsweise indem langsame Algorithmen nach einer bestimmten Laufzeit oder einer bestimmten Quote von beendeten Algorithmen abgebrochen werden [Gagliolo und Schmidhuber, 2006]. Dadurch werden unnötig lange Berechnungszeiten vermieden.

Die daraus resultierende Ungenauigkeit hat keine größeren Auswirkungen auf die Ergebnisse des gesamten Portfolios, da sehr langsame Algorithmen sowieso nicht bis zur Beendigung ausgeführt werden, sodass diese Methode gute Ergebnisse erwarten lässt. Das einzige Problem ist es einen guten Zeitpunkt für den Abbruch zu finden.

Für Simulationsalgorithmen ist der Einsatz dieser Art der Entwicklung wahrscheinlich nicht zweckmäßig. Ein Simulationslauf kann viele Iterationen ähnlicher Berechnungen haben, sodass sich die negativen Auswirkungen einer ungünstigen Ressourcenverteilung vielfach addieren würden [Ewald et al., 2010].

3.2 Dynamische Entwicklung

Neben der klaren Trennung zwischen Trainings- und Ausführungsphase gibt es dynamische Ausführungsphasen, in denen sich die Strategie des Portfolios verändern kann. Es werden zwei Typen unterschieden, die Offline- und die Onlineentwicklung.

Im Zusammenhang mit Simulationsalgorithmen kann keine der beiden Varianten generell favorisiert werden. Je nach Anwendung kann die eine oder die andere Methode besser sein. Würde man beispielsweise als Probleminstanz die Berechnung von 1000 Zustandsübergängen eines zellulären Automaten heranziehen, könnte eine Verteilung der Ressourcen zu Beginn der Berechnungen nicht optimal sein. Eine Anpassung innerhalb der Berechnungen wäre zweckmäßig. Würden jedoch viele solcher Replikationen berechnet werden, könnte eine Offlineentwicklung möglicherweise nach einigen Problemen schon eine gute Verteilung der Ressourcen liefern, sodass die Onlineentwicklung nur wenig Anpassung durchführen würde, welche den zusätzlichen Aufwand nicht rechtfertigt.

Offline

Bei der Offlinemethode werden Messdaten einer durchgeführten Berechnung in der Ausführungsphase genutzt um die vorhandene Strategie zu verbessern. Die Verteilung von Ressourcen an Algorithmen kann sich somit zwischen dem Lösen von Probleminstanzen ändern. Diese Maßnahme erhöht die Flexibilität des Portfolios gegenüber neuen Problemarten, wel-

che in der Trainingsphase unberücksichtigt waren. In Folge der immer neu erhaltenden Daten kann die Trainingsphase verkürzt werden. Damit einseitige Probleminstanzen zu Beginn der Ausführungsphase das Portfolio nicht zu stark in eine Richtung spezialisieren sollte der Grad der Berücksichtigung neuer Ergebnisse zu Beginn niedrig sein und nur langsam ansteigen [Gagliolo und Schmidhuber, 2006].

Online

Die Onlinemethode geht noch einen Schritt weiter und aktualisiert die Ressourcenverteilung des Portfolios während einer Problemlösung. In [Gagliolo und Schmidhuber, 2006] wird dieser Ansatz verfolgt und ausführlich analysiert. Nach einer Iteration aller beteiligten Algorithmen werden die aktuellen Zwischenergebnisse oder die Vorhersagen über die zu erwartende Restlaufzeit oder weitere Kriterien benutzt um die Ressourcen neu aufzuteilen. Auf der einen Seite kann konkret auf Probleminstanzen individuell reagiert werden. Auf der anderen Seite jedoch werden starke Startalgorithmen womöglich gegenüber im Endergebnis besseren Algorithmen bevorzugt. In Folge dessen kann die Gesamtleistung des Systems schlechter sein als bei einer gleichmäßigen Ressourcenverteilung (siehe Abbildung II.8). Um solchen Fehlentwicklungen entgegenzuwirken muss genau wie bei der Offlineentwicklung der Grad der Berücksichtigung aktueller Ergebnisse am Anfang niedrig sein und langsam ansteigen.

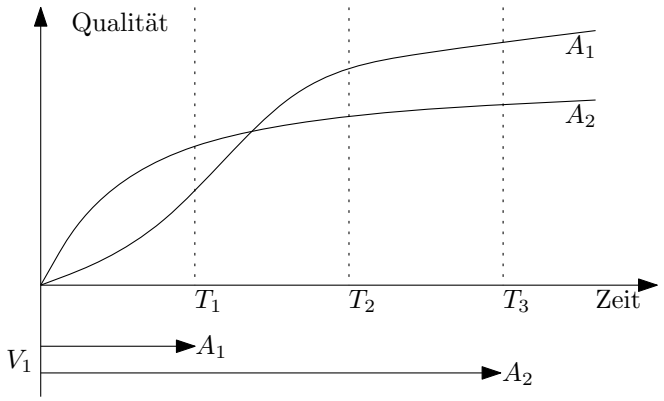

Abbildung II.8: Veranschaulichung des grundlegenden Problems der Onlineentwicklung. $2 \cdot T_2$ Zeiteinheiten stehen zur Verfügung und sie werden aufgrund der guten Anfangsergebnisse von A_2 entsprechend des Verlaufs V_1 verteilt. A_2 läuft bis T_3 und A_1 bis T_1. Das Ergebnis ist dadurch sogar schlechter, als würden beide Algorithmen gleichberechtigt T_2 Zeiteinheiten ausgeführt.

III Zusammenfassung

In der vorliegenden Arbeit wurden Algorithmenportfolios und ihre Anwendbarkeit auf Simulationsprobleme betrachtet. Sie bieten einen interessanten Ansatz mit den Schwierigkeiten des von Rice vorgestellten Algorithm Selection Problems umzugehen.

Die Konstruktion von Algorithmenportfolios ist ein in der Literatur wenig beachtetes Thema, die meisten Arbeiten betrachten von vornherein nur kleine Algorithmenmengen (2-3) und erstellen daraus ihre Portfolios.

Die große Komplexität von Simulationsproblemen macht es schwierig Algorithmenportfolios allgemein im Zusammenhang mit ihnen zu beurteilen. Restartstrategien ermöglichen mitunter gute Leistungssteigerungen, sie sind jedoch für Simulationsalgorithmen nur bedingt anwendbar. Die Einordnung von Simulationsalgorithmen in die oft betrachteten Kategorien der Entscheidungs- und Optimierungsprobleme lässt sich nicht durchführen, wodurch bisherige Ergebnisse nicht direkt übertragen und immer kritisch begutachtet werden müssen.

Die Kooperation von Algorithmen innerhalb eines Portfolios und die Ausnutzung von Problemstrukturen ist ein interessanter Ansatz zur Verbesserung der Leistung. Für eine zweckmäßige Umsetzung und Evaluierung dieses Ansatzes müssten Portfolios individuell betrachtet werden, was einer automatisierten Konstruktion effektiver Portfolios widerspricht.

Die drei verschiedenen Ausführungsarten haben ihre jeweiligen Vor- und Nachteile. Die einzelne Ausführung eines Algorithmus wäre für Simulationsprobleme gut geeignet, sofern genügend Leistungsdaten gesammelt sind und/oder alle Ausführungen in die Simulationsergebnisse mit einfließen sollen und kein aufwändiger, länger zu berechnender Verlauf ignoriert werden darf. Ist dies nicht der Fall bietet die verschränkte Ausführung womöglich bessere Aussichten. Eine parallele Ausführung der Algorithmen scheint aufgrund der vielfach vorhandenen Parallelität innerhalb eines Simulationsystem nicht zweckmäßig zu sein. Zudem müssten, sofern jeder Verlauf berücksichtigt werden muss, alle Algorithmen beendet werden!

Die Offline- oder Onlineentwicklung für Simulationsprobleme zu nutzen scheint sinnvoll, eine statische Entwicklung wäre zu unflexibel gegenüber neuen Problemarten.

Insgesamt bieten Algorithmenportfolios interessante Möglichkeiten die Performance von Simulationen zu beeinflussen. Aufgrund der großen Anzahl an Variationen und möglichen

Problemen müsste sich jedoch für jeden Einsatz überlegt werden welchen Typ Portfolio man konstruieren und benutzen möchte. Die wichtige Frage, ob signifikante Leistungssteigerungen mit ihrer Hilfe erzielt werden können, kann im Allgemeinen noch nicht beantwortet werden.

Literaturverzeichnis

[Brélaz, 1979] Brélaz, D. (1979). New methods to color the vertices of a graph. *Communications of the ACM*, 22:251–256.

[Ewald et al., 2010] Ewald, R., Schulz, R., und Uhrmacher, A. M. (2010). Selecting Simulation Algorithm Portfolios by Genetic Algorithms. In *IEEE Workshop on Principles of Advanced and Distributed Simulation (PADS)*, Seiten 1–9.

[Fukunaga, 2000] Fukunaga, A. (2000). Genetic algorithm portfolios. In *Proceedings of the 2000 Congress on Evolutionary Computation*, Band 2, Seiten 1304 – 1311.

[Gagliolo und Schmidhuber, 2006] Gagliolo, M. und Schmidhuber, J. (2006). Learning dynamic algorithm portfolios. *Annals of Mathematics and Artificial Intelligence*, 47:3–4.

[Gagliolo und Schmidhuber, 2007] Gagliolo, M. und Schmidhuber, J. (2007). Learning Restart Strategies. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, Seiten 792–797.

[Gagliolo und Schmidhuber, 2008] Gagliolo, M. und Schmidhuber, J. (2008). Towards Distributed Algorithm Portfolios. In *International Symposium on Distributed Computing and Artificial Intelligence*, Band 50 in *Advances in Soft Computing*, Seiten 634–643. Springer.

[Gomes und Selman, 2001] Gomes, C. P. und Selman, B. (2001). Algorithm portfolios. *Artificial Intelligence*, 126:43–62.

[Guyon und Elisseeff, 2003] Guyon, I. und Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182.

[Huberman et al., 1997] Huberman, B. A., Lukose, R. M., und Hogg, T. (1997). An Economics Approach to Hard Computational Problems. *Science*, 275:51–54.

[Kautz et al., 2002] Kautz, H. A., Horvitz, E., Ruan, Y., Gomes, C. P., und Selman, B. (2002). Dynamic Restart Policies. In *Proceedings of the 18th National Conference on Artificial Intelligence*, Seiten 674–681.

[Kira und Rendell, 1992] Kira, K. und Rendell, L. A. (1992). The Feature Selection Problem: Traditional Methods and a New Algorithm. In *Proceedings of the 10th National Conference on Artificial Intelligence*, Seiten 129–134.

[Markowitz, 1952] Markowitz, H. (1952). Portfolio selection. *Journal of Finance*, 7:77–91.

[Rice, 1976] Rice, J. R. (1976). The Algorithm Selection Problem. *Advances in Computers*, 15:65–118.

[Roberts, 2006] Roberts, M. (2006). Harnessing Algorithm Bias A Study of Selection Strategies and Evaluation for Portfolios of Algorithms.

[Rybacki et al., 2009] Rybacki, S., Himmelspach, J., und Uhrmacher, A. M. (2009). Experiments with Single Core, Multi-core, and GPU Based Computation of Cellular Automata. In *Proceedings of the 2009 First International Conference on Advances in System Simulation*, Seiten 62–67. IEEE Computer Society.

[Smith-Miles, 2009] Smith-Miles, K. A. (2009). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys*, 41:1–25.

Abbildungsverzeichnis