Auf dem Weg zum industrietauglichen Evolutionären Algorithmus

Wilfried Jakob

Institut für Angewandte Informatik, Forschungszentrum Karlsruhe
Postfach 3640, 76021 Karlsruhe
Tel. (07247) 82-4663
Fax (07247) 82-2602
E-Mail: wilfried.jakob@iai.fzk.de

1 Motivation

Evolutionäre Algorithmen (EA) zeichnen sich durch ihre allgemeine Anwendbarkeit aus und sind dazu geeignet, Lösungen oder zumindest Näherungslösungen für Aufgaben zu finden, für die es z.B. auf Grund von Nichtlinearitäten keine mathematischen Lösungsverfahren gibt. Zu den in diesem Sinne erfolgreich bewältigten Aufgaben gehören auch viele NP-vollständigen Probleme. Diesem Vorteil steht der Nachteil der EA gegenüber, dass ihre Leistungsfähigkeit stark von einer geeigneten Wahl von zum Teil verfahrensspezifischen Strategieparametern wie z.B. der Populationsgröße oder Mutationsraten abhängt. Dazu kommt das Problem, dass EA in der Nähe eines Optimums schlecht konvergieren. Letzteres wird bei praktischen Anwendungen meist durch Hybridisierung gelöst, d.h. man unterstützt die evolutionäre Suche durch ein in der Regel anwendungsspezifisches (lokales) Suchverfahren (LSV) [1-5]. Dem Vorteil einer erheblichen Steigerung der Konvergenzgeschwindigkeit stehen zwei Nachteile gegenüber: Zum einen wird aus dem allgemein anwendbaren EA ein problemspezifisches Werkzeug und zum anderen kommen weitere Strategieparameter, nämlich die des lokalen Verfahrens hinzu. Zumindest muss die Intensität der lokalen Suche und damit die Aufteilung der Rechenleistung zwischen den beiden Algorithmen festgelegt werden. All das macht den (hybriden) EA zu einem Werkzeug für Spezialisten und man muss leider feststellen, dass sich Evolutionäre Algorithmen zwar durch ein breites Anwendungsspektrum, aber nicht durch eine breite Anwendung auszeichnen.

An ein industrietaugliches Optimierungs- oder Planungsverfahren können folgende Forderungen gestellt werden:

1. Klare Abgrenzung des Einsatzgebiets
2. Robustheit: Das Verfahren muss auch bei „ungünstigen“ Parametern der Anwendung oder Anwendungssituationen funktionieren.
3. Einfachheit: Möglichst wenig Verfahrensparameter, die nicht zu sensibel auf schlechte Einstellungen reagieren. Man sollte nur wenig Vorwissen für einen erfolgreichen Einsatz benötigen.
4. Es sollte möglich sein, existierende Lösungen in das Verfahren einzubeziehen. Es erhöht die Akzeptanz beim Anwender erheblich, wenn ein neues Verfahren zumindest nicht schlechter ist als das bisherige, da es dessen Lösungen verbessert.
5. Geschwindigkeit: Die beste Lösung nützt nichts, wenn sie zu spät kommt.

Für EA kann allgemein gesagt werden, dass sie als robust gelten und dass bei EA mit elitärer Akzeptanz der Nachkommen durch die Aufnahme existierender Lösungen in die Startpopulation zumindest deren Qualität als Ergebnis erreicht wird. Die Frage der Geschwindigkeit kann nur anwendungsabhängig beantwortet werden: Zum einen legt die Anwendung den verfügbaren Zeitrahmen für eine Optimierung fest und zum anderen bestimmt ihre Komplexität und die Anzahl ihrer Parameter den Zeitbedarf für einen Optimierungslauf genauso wie die Dauer der Evaluation einer vorgeschlagenen Lösung (Fitnessberechnung).

Der vorliegende Beitrag konzentriert sich auf die Einfachheit der Handhabung des Verfahrens unter Eingrenzung des Anwendungsgebiets auf *Parameteroptimierung*, wobei je nach Anwendungsfall von bis zu einigen Dutzend Parametern ausgegangen werden kann. Ziel ist eine für EA allgemein anwendbare Methode zur adaptiven Hybridisierung und dadurch eine deutliche Verringerung der Strategieparameter des resultierenden Verfahrens. Vom EA wird dabei gefordert, dass die Population größer als eins ist und dass die Nachkommen mit den Eltern um die Aufnahme in die Nachfolgegeneration konkurrieren (Plus-Strategie im Sinne der ES-Terminologie [6]). Außerdem ist es hilfreich, wenn pro Paarung mehrere Nachkommen erzeugt werden.

Kapitel 2 gibt einen Überblick über relevante Strategieparameter der hier behandelten Klasse hybrider evolutionärer Algorithmen, die auch unter der Bezeichnung *Memetische Algorithmen* bekannt sind. Außerdem werden die für die experimentellen Überprüfungen verwendeten Verfahren kurz vorgestellt. Kapitel 3 behandelt das neue Konzept zur adaptiven Hybridisierung und vergleicht es mit anderen Ansätzen. Über die zur Überprüfung der Tauglichkeit der neuen Methode durchgeführten Experimente wird schließlich im 4. Kapitel berichtet, bevor der Beitrag mit einer Zusammenfassung der Ergebnisse endet.

2 Evolutionäre und memetische Algorithmen

In Bild 1 ist der Pseudocode eines typischen EA dargestellt. Die kursiv hervorgehobene Integration eines LSVs in die Produktion der Nachkommen macht ihn zum Memetischen Algorithmus [3]. In der Regel werden durch Anwendung genetischer Operatoren wie Crossover und verschiedene Mutationen mehrere Nachkommen erzeugt, von denen entweder nur das Beste oder alle lokal verbessert werden. In der Literatur kommt auch noch die zufällige Auswahl eines Teils der zu verbessernden Nachkommen vor. Anschließend wird das Chromosom im Falle von Lamarckscher Evolution an die gefundene Verbesserung angepasst oder nicht, so dass nur der Fitnesszuwachs wirksam ist (Baldwin-Evolution). Die Frage, welcher der beiden Evolutionsarten der Vorzug zu geben ist, wird in der Literatur kontrovers diskutiert [7,8]. Meist wird der Baldwin-

```
Initialize and evaluate start population
REPEAT UNTIL stop condition is satisfied (generational loop)
  FOR all individuals of the population
    Choose partner (within neighborhood, ranking-based selection)
    FOR all genetic operations (set of genetic operators)
      Produce offspring and evaluate it or them
    IF improve best only
      Improve best offspring by preselected local searcher
    ELSE
      Improve all offspring by preselected local searcher
    IF Lamarckian evolution
      Update chromosome of the best offspring to its improved version
    Accept or reject best offspring according to the acceptance rule
Deliver best individual (and other, if required) as result
```

Bild 1: Typischer Ablauf eines EA mit integrierter lokaler Verbesserung (*kursiv*). Die Besonderheiten des nachfolgend beschriebenen EA sind durch den Times-Font hervorgehoben.

Evolution der Vorzug gegeben, da sie eine größere genetische Vielfalt über einen längeren Zeitraum bewirkt. Eigene Untersuchungen haben jedoch gezeigt, dass bei Verwendung des von Gorges-Schleuter [9] vorgeschlagenen Nachbarschaftsmodells, das der Gefahr vorzeitiger Konvergenz wirksam entgegentritt, die Lamarcksche Evolution zum Teil erheblich besser abschneidet [4,5]. Dies gilt übrigens auch für die Evolutionsstrategie [10]. Daher wurden die hier beschriebenen Experimente mit Lamarckscher Evolution durchgeführt.

2.1 Strategieparameter

Zu den wichtigsten EA-Strategieparametern zählen die Populationsgröße, Mutations- und Crossoverraten, die Anzahl der pro Paarung erzeugten Kinder und weitere Parameter, die von der konkreten Ausgestaltung des EA abhängen, wie z.B. der Rankingfaktor bei rankingbasierter Selektion. Es wird davon ausgegangen, dass sich diese Parameter bis auf die Populationsgröße EA-spezifisch so einstellen lassen, dass sie für ein breites Spektrum von Parameteroptimierungsaufgaben geeignet sind.

Die Integration lokaler Suche in einen EA kann als Voroptimierung der Startpopulation, als Nachoptimierung der EA-Ergebnisse oder in Form eines Memetischen Algorithmus erfolgen. Da eigene Untersuchungen die Überlegenheit der Memetischen Variante deutlich gezeigt haben [4,5], wird im Folgenden nur noch diese Art der Hybridisierung betrachtet. Hierbei sind folgende Strategieparameter von Bedeutung:

1. Auswahl des LSVs sofern mehrere zur Verfügung stehen
2. Auswahl der Anzahl zu verbessernder Nachkommen pro Paarung: nur das beste, alle, zufallsbestimmt
3. Genauigkeit der lokalen Suche (Anzahl der Iterationen, LSV-Abbruchschranke, …)

Die beiden letzten Parameter bestimmen letztlich die Aufteilung der Rechenzeit zwischen evolutionärer und lokaler Suche, eine Frage, auf deren grundlegende Bedeutung Goldberg und Voessner bereits 1999 hingewiesen haben [11]. Der in [12] weiterentwickelte systemtheoretische Ansatz kann leider für praktische Anwendung keine verwertbaren Empfehlungen geben, wie die Autoren selber einräumen.

2.2 GLEAM und HyGLEAM

Das von Blume 1990 vorgestellte GLEAM (**G**eneral **L**earning **E**volutionary **A**lgorithm and **M**ethod) [13] kann als repräsentativer EA-Vertreter angesehen werden, da GLEAM verschiedene Eigenschaften der klassischen EAs in sich vereint und durch seine flexible Codierung ein breites Anwendungsspektrum abdeckt. Geeignete lokale Suchverfahren sollen wegen der Bewahrung der allgemeinen Anwendbarkeit des resultierenden Hybrids möglichst wenig Anforderungen an den Suchraum stellen und Beschränkungen, die bei praktischen Problemen fast immer vorkommen, berücksichtigen können. Daher wurden zwei als robust und leistungsfähig bekannte Algorithmen ausgewählt, nämlich das Rosenbrock- [14] und das Complex-Verfahren [15], im Folgenden mit *R* und *C* abgekürzt. Da sich der vorliegende Beitrag auf eine adaptive Hybridisierung konzentriert, kann aus Platzgründen nur kurz auf die beteiligten Verfahren eingegangen werden und es wird auf die angegebene Literatur verwiesen. Eine ausführliche Beschreibung von GLEAM kann in [5] und von den beiden LSV in [16] gefunden werden.

GLEAM ist ein eigenständiger Evolutionärer Algorithmus, der Elemente der Evolutionsstrategie [6,16] und der (reellcodierten) Genetischen Algorithmen [2,17] mit Konzepten der Informatik (abstrakte Datentypen) verbindet. Die Codierung der zu optimie-

renden Parameter erfolgt in den so genannten Aktionen, die aus einem oder mehreren Parametern vom Typ Bool, Integer oder Real entsprechend ihrem anwendungsspezifisch definierten Aktionstyp bestehen. Eine Aktion drückt damit eine bestimmte Eigenschaft einer möglichen Lösung aus, z.B. bei einem Design die Anwesenheit einer bestimmten Komponente, die durch die Aktionsparameter näher bestimmt wird oder die Relationen zu anderen Komponenten entsprechend den Aktionsparametern eingeht. Im einfachsten Fall codieren die Aktionen nur bestimmte einzelne Parameter einer Lösung. Die Aktionen bilden eine Aktionskette (AK), die den Chromosomen des biologischen Vorbilds entspricht. Der hier interessierende AK-Typ ist der einfachste von GLEAM, bei dem die Aktionsreihenfolge keine Bedeutung trägt und jeder Aktionstyp mit genau einer Aktion in der AK vertreten ist. Bild 1 zeigt den grundsätzlichen Ablauf von GLEAM mit Reproduktion in der lokalen Nachbarschaft und ranking-basierter Partnerwahl. Als Akzeptanzstrategie für das beste Nachkommen wird die Konkurrenz mit dem Elter benutzt, wobei es entweder besser als das Elter sein muss oder besser als das lokal schlechteste und keine Verschlechterung der Nachbarschaft stattfinden darf (elitäre Ersetzungsstrategie). Das Codierungskonzept von GLEAM erlaubt eine einfache Abbildung von kombinatorischen Problemen auf die Parameteroptimierung und macht sie so einer Bearbeitung mit HyGLEAM zugänglich. Dazu wird jede Aktion um einen Permutationsparameter erweitert, der bestimmt, mit welcher anderen Aktion der Platz bei der AK-Auswertung getauscht wird. Der neue Parameter bestimmt so die Reihenfolge der Interpretation der Aktionen und die Aktionen müssen nun nicht mehr durch entsprechende Mutationen im Chromosom vertauscht werden.

Das Rosenbrock-Verfahren ist eine modifizierte Koordinatenstrategie und der Complex-Algorithmus ein Polyederverfahren, das für Restriktionen erweitert wurde. Beide Verfahren kommen ohne Ableitungen aus und werden gemäß der Implementierung von Schwefel [16] benutzt. Das Rosenbrock-Verfahren bricht ab, wenn ein Maß für die Länge des zurückgelegten Weges und die Richtungsänderung einen vorgegebenen Wert unterschreitet. Der Complex-Algorithmus terminiert, wenn entweder fünf mal hintereinander keine Verbesserung eintrat oder wenn die gleiche implizite Beschränkung fünf mal hintereinander eine unzureichende Kontraktion veranlasst hat. Die Implementierung beider Verfahren erlaubt außerdem die Vorgabe einer Iterationsgrenze.

In memetischen Teil von HyGLEAM (**Hy**brid **G**enera**l** Purpose **E**volutionary **A**lgorithm and **M**ethod) werden die beiden LSV wie in Bild 1 dargestellt in die Produktion der Nachkommen integriert. Dabei kann eingestellt werden, ob nur der beste oder alle Nachkommen lokal verbessert werden und ob der Genotyp angepasst werden soll oder nicht.

3 Adaption in memetischen Algorithmen

Hart [18] und Krasnogor [19] haben gezeigt, dass die Wahl eines geeigneten LSVs anwendungsabhängig ist und einen wesentlichen Einfluss auf die Suchgeschwindigkeit hat. Eine weitere wesentliche Größe zur Parametrierung ist die bereits erwähnte Aufteilung der Rechenzeit zwischen globaler und lokaler Suche. Im Folgenden wird ein Konzept zur adaptiven Einstellung beider Parameter vorgestellt und mit anderen Ansätzen verglichen.

3.1 Konzept der Kosten-Nutzen basierten Adaption

Die adaptive Steuerung beruht auf dem beobachteten Erfolg ausgedrückt durch den erzielten Fitnesszuwachs und den dazu notwendigen Kosten in Form von Fitnessberech-

nungen. Sie wird zunächst am Beispiel der Aufteilung zwischen den beiden lokalen Verfahren beschrieben. Anfänglich ist die Wahrscheinlichkeit für ihre Anwendung gleich. Die Anzahl der Anwendungen wird je Verfahren gezählt und der jeweils erreichte relative Fitnessgewinn *rfg* wird zusammen mit den dazu benötigten Evaluationen *eval* aufsummiert. Da ein bestimmter Fitnesszuwachs (Fitness nach der lokalen Suche (f_{LS}) abzüglich der Fitness nach Anwendung der genetischen Operatoren (f_{evo})) mit zunehmender Ausgangsfitness f_{evo} höher zu bewerten ist, wird statt des absoluten Wertes der Fitnessdifferenz ein relatives Maß benutzt, dass sich am noch möglichen Fitnessgewinn orientiert. Dazu wird die Fitnessfunktion im Bereich $0 .. f_{max}$ normiert und *rfg* wie folgt berechnet:

$$rfg = \frac{f_{LS} - f_{evo}}{f_{\max} - f_{evo}} \tag{1}$$

Die Ausführungswahrscheinlichkeiten der lokalen Verfahren werden neu justiert, wenn entweder jedes Verfahren mindestens $usage_{min}$ mal benutzt wurde oder nach spätestens $matings_{max}$ Paarungen. Die neue Relation zwischen den beiden lokalen Verfahren berechnet sich wie folgt:

$$\frac{\sum rfg_{i,compl}}{\sum eval_{i,compl}} : \frac{\sum rfg_{j,rosen}}{\sum eval_{j,rosen}} \tag{2}$$

Die Summen werden nach der Anpassung zurückgesetzt, um eine schnellere Anpassung zu erreichen. Wenn die Anwendungswahrscheinlichkeit eines der Verfahren dreimal hintereinander weniger als ein P_{min} (0.1 in den Experimenten) beträgt, wird es abgeschaltet. Um einer voreiligen Abschaltung entgegenzuwirken, wird bei erstmaliger Unterschreitung von P_{min} die berechnete Wahrscheinlichkeit auf P_{min} heraufgesetzt.

Der Ansatz kann leicht auf weitere Suchverfahren oder Parameter wie die Genauigkeit der lokalen Suche erweitert werden. Dazu wird der Wertebereich in eine Anzahl von Bereichen unterteilt, was am Beispiel der Iterationsgrenze erläutert werden soll. Hierfür mögen 10 *Level* genannte Bereiche genügen, für die beispielsweise Werte zwischen 100 und 2000 zu Grunde gelegt werden, siehe auch Tabelle 1. Es sind immer genau drei Level gleichzeitig aktiv, d.h. sie haben eine Wahrscheinlichkeit, ausgewählt zu werden, von $p > 0$. Die Anpassung der Wahrscheinlichkeiten erfolgt wie bei der LSV-Auswahl beschrieben. Wenn dabei der niedrigste oder höchste aktive Level eine Wahrscheinlichkeit von mehr als 50% erhält, erfolgt eine Levelanpassung, wie in Bild 2 dargestellt. Dazu wird zunächst der Level am anderen Ende deaktiviert und seine Wahrscheinlichkeit dem Nachbarlevel zugeschlagen (mittlere Zeile von Bild 2). Der neue Level (Level 6 im Bild) erhält 20% der Wahrscheinlichkeit der anderen beiden und wird damit aktiviert (letzte Zeile in Bild 2). Damit wird eine Bewegung der aktiven Levels auf der Skala möglicher Level entspre-

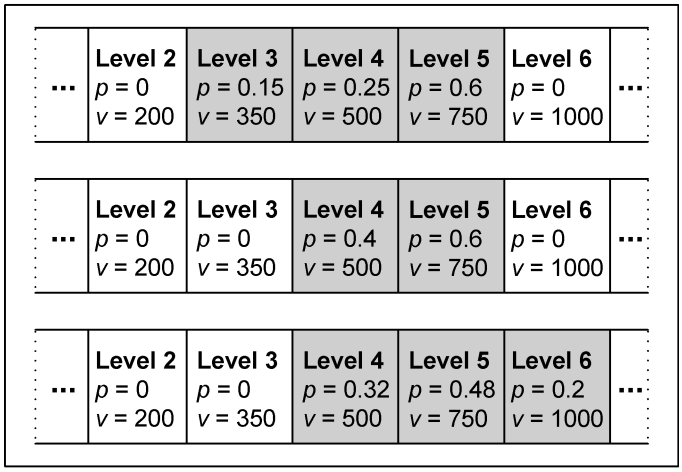

Bild 2: Parameteranpassung am Beispiel der Überschreitung des Wahrscheinlichkeitsgrenzwertes von Level 5. Die jeweils momentan aktiven Levels sind grau hinterlegt. Die Levelwerte sind mit *v* angegeben.

chend dem erreichten Fitnesszuwachs und der dazu erforderlichen Evaluationen erreicht. Da die geringste Levelwahrscheinlichkeit auf 10% festgesetzt wird, bleiben auch bei Erreichen des Skalenendes drei Level aktiv, wodurch die Mobilität dauerhaft gewährleistet bleibt. Mit Hilfe dieses Ansatzes können auch weitere Strategieparameter der Hybridisierung angepasst werden; konkret im vorliegenden Fall sind das neben den Iterationsgrenzen die normierte Abbruchschranke des Rosenbrock-Verfahrens th_R und die Wahrscheinlichkeit $all\text{-}impr_R$ und $all\text{-}impr_C$, mit der weitere Nachkommen einer Paarung außer dem besten lokal optimiert werden *(all-Verbesserung*).

Strategieparameter	**Werte**									
$limit_R$, $limit_C$	100	200	350	500	750	1000	1250	1500	1750	2000
th_R	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}	
$all\text{-}impr_R$, $all\text{-}impr_C$	0	0.2	0.4	0.6	0.8	1.0				

Tabelle 1: Adaptiv kontrollierte Strategieparameter mit Anzahl und Werten ihrer Levels.

Der Gedanke liegt nahe, die lokale Suche anfänglich nur recht grob zu betreiben und erst im Verlaufe der Evolution zu präzisieren. Schließlich genügt die ungefähre Bestimmung eines lokalen Optimums, solange sie genau genug ist, um zwischen den lokalen Optima korrekt differenzieren zu können. Daher werden die beiden Levels der Iterationsgrenzen ($limit_R$ und $limit_C$) beginnend mit dem niedrigsten mit Wahrscheinlichkeitswerten von 50%, 30% und 20% initialisiert, während die beiden anderen Levels jeweils mit einem Drittel vorbelegt werden. Tabelle 1 gibt die adaptiv angepassten Strategieparameter und ihre Level-Werte wieder.

Bild 3 zeigt den für das adaptive HyGLEAM/A erweiterten Pseudocode von Bild 1, der noch zwei extern einzustellende Optionen aufweist: Die Entscheidung über die Verbesserung nur des besten Nachkommen *(improve best only*, *best-Verbesserung*) oder aller und die über die Anpassung des Genotyps (*Lamarckian evolution*). Alle anderen Entscheidungen (*Choose …*) erfolgen durch den beschriebenen adaptiven Mechanismus. Dabei gibt es bei der all-Verbesserung folgende Besonderheit: Wenn nach erfolgter lokaler Verbesserung ein anderer Nachkomme als der Evolutionsbeste die größte Fitness hat, es also richtig war, nicht nur den Evolutionsbesten lokal zu verbessern, dann wird die Differenz zwischen der besten Fitness der Nachkommen und des verbesserten Evolutionsbesten als Belohnung zum Fitnesszuwachs hinzuad-

```
Initialize and evaluate start population
REPEAT UNTIL stop condition is satisfied (generational loop)
  FOR all individuals of the population
    Choose partner (within neighborhood, ranking-based selection)
    FOR all genetic operations (set of genetic operators)
      Produce offspring and evaluate it or them
    IF improve best only
      Choose local searcher and its parameters
      Improve best offspring by selected local searcher
      Record effort and fitness gain
    ELSE
      Choose local searcher and LS_probability
      FOR all offspring
        IF best evo_offspring OR improve according to LS_probability
          Choose parameters of the LS
          Improve offspring by selected local searcher
          Record effort and fitness gain for LS selection and
            parameterization
      IF best improved offspring other than best evo_offspring
        Add difference to fitness gain as reward
      Record effort and fitness gain for LS_probability
    IF Lamarckian evolution
      Update chromosome of the best offspring to its improved version
    Accept or reject best offspring according to the acceptance rule
Deliver best individual (and other, if required) as result
```

Bild 3: Um die Adaption erweiterter Pseudocode von Bild 1.

diert. Dieser Anteil wird damit praktisch zweimal gezählt.

Fitnessklasse	Fitnessbereiche
fc1	0.4 0.7 1.0
fc2	0.35 0.65 0.85 1.0
fc3	0.3 0.55 0.75 0.9 1.0

Tabelle 2: Fitnessklassen und ihre Wertebereiche als Anteil an f_{max}.

Erste Experimente haben gezeigt, dass die gegenseitige Beeinflussung der adaptiven Regelung aus unterschiedlichen Fitnessbereichen negative Auswirkungen haben kann. So kann es z.B. sinnvoll sein, anfänglich beide LSV zu verwenden und in einer späteren Phase der Evolution verstärkt auf nur eines der beiden zu setzen. Mit anderen Worten: *was gut für schlechte Individuen ist, muss nicht geeignet für gute sein und umgekehrt*. Um dieser Beobachtung gerecht zu werden, wurde eine getrennte Adaption für unterschiedliche Fitnessklassen eingeführt. Konkret wurden die Experimente mit den drei in Tabelle 2 wiedergegebenen Klassen durchgeführt. Wenn ein Individuum eine Fitness erreicht, deren Klasse noch unbenutzt ist, werden die Level-Einstellungen der nächst niedriger genutzten kopiert, um bereits gelernte Lektionen nicht noch einmal lernen zu müssen. Bei all-Verbesserung bestimmt die Fitness des Evolutionsbesten die Wahl des lokalen Verfahrens und von $all\text{-}impr_R$ bzw. $all\text{-}impr_C$, während die Parametrierung des gewählten LSVs ($limit_R$, $limit_C$ und th_R) entsprechend der Fitness eines jeden Nachkommen eingestellt wird.

Geschwindigkeit	LSV-Auswahl		Parameteranp.	
	$usage_{min}$	$matings_{max}$	$usage_{min}$	$matings_{max}$
schnell	3	15	3	12
mittel	5	20	4	15
langsam	8	30	7	25

Tabelle 3: Die Parametrierungen der drei Anpassungsgeschwindigkeiten für die LSV-Auswahl und die Parameteranpassungen.

Neben der Frage, ob mit all- oder best-Verbesserung gearbeitet werden soll, spielt noch die Anpassungsgeschwindigkeit der Adaption eine wichtige Rolle. Tabelle 3 zeigt die Werte für $usage_{min}$ und $matings_{max}$ der drei untersuchten Anpassungsgeschwindigkeiten *schnell*, *mittel* und *langsam* für die LSV-Auswahl und die Parameteranpassungen. Nur wenn es gelingt, für alle untersuchten Testanwendungen eine gemeinsame Parametrierung dieser neuen Strategieparameter der Adaption zu finden, kann von einem Fortschritt gesprochen werden und nur dann ist man dem eingangs genannten Ziel einer Verfahrensvereinfachung näher gekommen.

3.2 Andere Verfahren zur Adaption von Strategieparametern

Ong und Keane [20] benutzen einen ähnlichen Kosten-Nutzen basierten Ansatz mit dem Ziel „to promote cooperation and competition among the different LSs working together to accomplish the shared optimization goal".[1] Sie ermitteln eine Belohnung η an Hand von Formel (3) und berechnen daraus die Wahrscheinlichkeitsverteilung zur Auswahl eines ihrer insgesamt 9 LSV, wobei f_{Best} die Fitness des bisher besten Individuums ist.

$$\eta = \beta \frac{f_{LS} - f_{evo}}{eval_{LS}} \quad \text{mit} \quad \beta = \frac{f_{LS}}{f_{Best}} \tag{3}$$

Der Faktor β relativiert den Fitnesszuwachs bezogen auf die Fitness des besten Individuums was ähnlich zur vorgestellten relativen Fitness ist: Der Zuwachs wird nicht

[1] Die Ideen von Ong und Keane entstanden offenbar parallel zur vorliegenden Arbeit, siehe auch [5].

absolut genommen sondern am bisher erreichten gemessen. Ong und Keane vergleichen diesen Ansatz mit einem heuristischen, der auf dem Fitnesszuwachs genotypisch ähnlicher Individuen aufbaut, und kommen zu dem Ergebnis, dass das auf der Belohnung η basierende Verfahren besser ist und dass es erheblich von der Anwendung abhängt, welche LSV besser oder schlechter geeignet sind. Im Unterschied zu HyGLEAM/A wird die Intensität der lokalen Suche durch eine Begrenzung auf 100 Iterationen für alle LSV beschränkt. Da Ong und Keane sich auf die LSV-Auswahl konzentrieren, während die vorliegende Arbeit stärker die Adaption der Rechenzeitverteilung zwischen globaler und lokaler Suche zum Ziel hat, ergänzen sich beide Arbeiten.

Bisher war von einem Mechanismus zur Adaption die Rede, der außerhalb des bereits selbst Adaption bewirkenden evolutionären Mechanismus steht. Warum also nicht die Strategieparameter zum Chromosom hinzufügen und die Adaption der Evolution überlassen, wie es Krasnogor et al. [21] getan haben? Schließlich hat die Selbstadaption beispielsweise bei der Schrittweitensteuerung der ES gute Ergebnisse gebracht. Der wesentliche Unterschied ist hierbei, dass unterschiedliche Schrittweiten den Aufwand für die Mutation nicht vergrößern und es tatsächlich nur auf den Fitnessgewinn ankommt. Aber die Auswahl zwischen unterschiedlichen LSV bewirkt unterschiedliche Kosten, sofern sie nicht einheitlich vorzeitig gestoppt werden. Krasnogor et al. sagen selbst: „The rationale is to propagate local searchers that are associated with fit individuals, as those individuals were probably improved by their respective memes." Oder aber sie wurden durch die vorangegangene Evolution verbessert! Dass man dies der Fitnesssumme nicht mehr ansehen kann, rechtfertigt die Untersuchung des vorgeschlagenen Ansatzes zur Kosten-Nutzen basierten Adaption.

Mit Zitzler et al. [22] und Bambha et al. [23] wird der Gedanke des adaptiv gesteuerten Zuwachses der Genauigkeit der lokalen Suche im Verlauf der Adaption geteilt. Der wesentliche Unterschied besteht in der vorher festzulegenden Dauer der Optimierung, die ihrem Verfahren zu Grunde liegt. Insbesondere bei neuen noch nicht bearbeiteten Aufgabenstellungen ist die Abschätzung einer ausreichenden Laufzeit dagegen problematisch. So sehen die Autoren auch den wesentlichen Einsatzbereich ihres Ansatzes bei Aufgaben, die einer festen zeitlichen Restriktion unterliegen.

4 Experimente und ihre Ergebnisse

4.1 Benchmarkaufgaben

Die zur experimentellen Überprüfung des HyGLEAM/A-Ansatzes notwendigen Testaufgaben müssen repräsentativ hinsichtlich realer Anwendungen sein, aber auch eine schnelle Optimierung erlauben, damit statistische Untersuchungen in vertretbarer Zeit möglich sind. Tabelle 4 gibt einen Überblick über wichtige Eigenschaften der sieben benutzten Testaufgaben, von denen zwei Anwendungsprobleme sind, während die fünf anderen der GENEsYs-Sammlung [24] entnommen wurden und in der Tabelle mit ihren dortigen Funktionsnummern angegeben sind. Shekel's Foxholes und die verallgemeinerte Rastrigin Funktion wurden um 30° im Raum gedreht, um sie schwieriger zu machen, siehe auch [5]. Ein Problem wird in der Tabelle als *stark multimodal* (*stark mm.*) bezeichnet, wenn es mehr als das 20-fache seiner Dimension an Suboptima aufweist. Tabelle 4 gibt für die Testfunktionen auch die Wertebereiche und die Genauigkeit der Zielwerte an, wobei lediglich im Falle von Shekel's Foxholes das exakte Optimum gefordert wird. Bei der fraktalen Funktion ist es unbekannt.

Testaufgabe	Parameter	Komb. Optim.	Modalität	Implizite Restrikt.	Wertebereich	Zielwert
Schwefels Kugel (f1)	30 real	nein	unimodal	nein	$[-10^{10}, 10^{10}]$	0.01
Shekel's Foxholes (f5)	2 real	nein	multimod.	nein	[-500, 500]	0.998004
Verallg. Rastrigin F. (f7)	5 real	nein	stark mm.	nein	[-5.12, 5.12]	0.0001
Fletcher & Powell (f16)	5 real	nein	multimod.	nein	[-3.14, 3.14]	0.00001
Fraktale F. (f13)	20 real	nein	stark mm.	nein	[-5, 5]	-0.05
Designoptimierung	3 real	nein	stark mm.	nein		
Ressourcenoptimierung	87 int.	(ja)	multimod.	ja		

Tabelle 4: Wichtige Eigenschaften der sieben Testaufgaben.

Aus Platzgründen können die beiden Anwendungsaufgaben hier nur kurz vorgestellt werden und der interessierte Leser wird auf die angegebene Literatur verwiesen. Alle sind ausführlich in [5] beschrieben.

Bei der Designoptimierung geht es um den Entwurf eines mikrooptischen Bauteils (Heterodynempfänger) bestehend aus zwei Kugellinsen und einer Photodiode [25]. Der Abstand zwischen dem Lichtleiter und erster Kugellinse sowie die beiden Brechungsindizes sollen so bestimmt werden, dass nicht nur eine optimale Ausleuchtung der Photodiode im Kollimationspunkt erreicht wird, sondern die ganze Anordnung auch möglichst unempfindlich gegenüber Fertigungstoleranzen innerhalb vorgegebener Grenzen ist. Letzteres macht die ansonsten einfache Aufgabe zu einem trotz der nur drei Parameter schwierigem, stark multimodalen Optimierungsproblem.

Die Ressourcenoptimierung erfolgt in Zusammenhang mit einer Scheduling-Aufgabe aus der Verfahrenstechnik [26]. Dabei sind 87 Chargen mit jeweils unterschiedlichem Mitarbeiterbedarf währen der Abarbeitung jeder Charge so zu starten, dass der resultierende Mitarbeiterbedarf (Ressource menschliche Arbeitkraft) pro Schicht ein vorgegebenes Maximum nicht überschreitet und möglichst noch darunter bleibt. Gleichzeitig sind natürlich die üblichen Randbedingungen wie Lieferfristen, die Verfügbarkeit gemeinsam genutzter Anlagen und die Verfügbarkeit von durch Vorgängerchargen produzierten Vorprodukten zu beachten. Da Belegungskonflikte, die durch die Reihenfolge der Chargen in den Aktionsketten gelöst werden, auch durch geeignete Verschiebung der Starttermine beseitigt werden können, ist der kombinatorische Charakter der Aufgabenstellung begrenzt (eingeklammertes „ja“ in Tabelle 4). Das Optimum ist nicht bekannt und für die Erfüllung der Testaufgabe wird ein Wert nahe dem besten Ergebnis von [26] gefordert: Eine Reduktion des maximalen Mitarbeiterbedarfs von 12 auf 9 pro Schicht bei gleichzeitiger Verkürzung der ursprünglich manuell geplanten Produktionszeit auf 70%.

4.2 Ergebnisse für HyGLEAM

GLEAM oder ein Hybrid wird zusammen mit konkreten Werten seiner Strategieparameter als *Job* bezeichnet. Die Bewertung basiert auf dem durchschnittlichen Aufwand gemessen in Evaluationen von in der Regel 100 Läufen pro Job, wobei nur solche Jobs verglichen werden, die erfolgreich sind, d.h. bei denen alle Läufe das vorgegebene Ziel erreicht haben. Der Vergleich basiert dann auf der erreichten Verbesserung gegenüber GLEAM bzw. im nächsten Abschnitt auch gegenüber HyGLEAM. Tabelle 5 vergleicht die besten Jobs von GLEAM und HyGLEAM zusammen mit ihren Strategieparametern. Die Tabelle zeigt die durchgängige Überlegenheit der Lamarckschen Evolution und die enorme Spreizung geeigneter Populationsgrößen beim reinen EA (GLEAM) sowie

deren deutliche Verringerung bei allen Hybriden. Ansonsten bestätigt die Tabelle die Notwendigkeit einer adaptiven Auswahl des LSVs, seiner Genauigkeit und der Entscheidung zwischen best- oder all-Verbesserung.

Testaufgabe	GLEAM		HyGLEAM						
	Eval.	Pg	LSV	Eval.	Pg	th_R	b/a	Lamarck	Verb.
Fletcher	483 566	600	C	4 684	5		best	ja	103
Ressourcenopt.	5 376 334	1 800	R	69 448	5	10^{-1}	best	ja	77,4
Foxholes	103 192	350	C	8 831	20		best	ja	11,7
Rastrigin	3 518 702	11 200	R	315 715	70	10^{-2}	all	ja	11,2
Fraktal	195 129	20	R	30 626	5	10^{-2}	best	ja	6,4
Designopt.	5 773	210	C	1 041	5		best	ja	5,6
Kugel	kein Erfolg		R	29 595	5	10^{-6}	best	ja	

Tabelle 5: Ergebnisse der sieben Testaufgaben in absteigender Reihenfolge der erreichten Verbesserung. Abkürzungen: Eval: Evaluationen, Pg: Populationsgröße, R: Rosenbrock-Verfahren, C: Complex-Algorithmus. b/a: best- oder all-Verbesserung, Lamarck: Lamarcksche Evolution, Verb: Verbesserungsfaktor gegenüber dem besten GLEAM-Job

4.3 Ergebnisse für adaptives HyGLEAM (HyGLEAM/A)

Bild 4 zeigt exemplarisch, dass sich je nach Anwendung im Durchschnitt unterschiedliche Werte für die Rosenbrockwahrscheinlichkeit (*prob. R*) und die Levels am Ende eines Laufes für die verschiedenen Fitnessklassen ergeben. Es unterstreicht zusammen mit weiteren Beobachtungen die in Kapitel 3 gegebene Begründung für die Einführung der nach Fitnessklassen getrennten Adaption. Aus technischen Gründen sind die Indices der Größen zur LSV-Unterscheidung in den Bildern nicht tief gestellt.

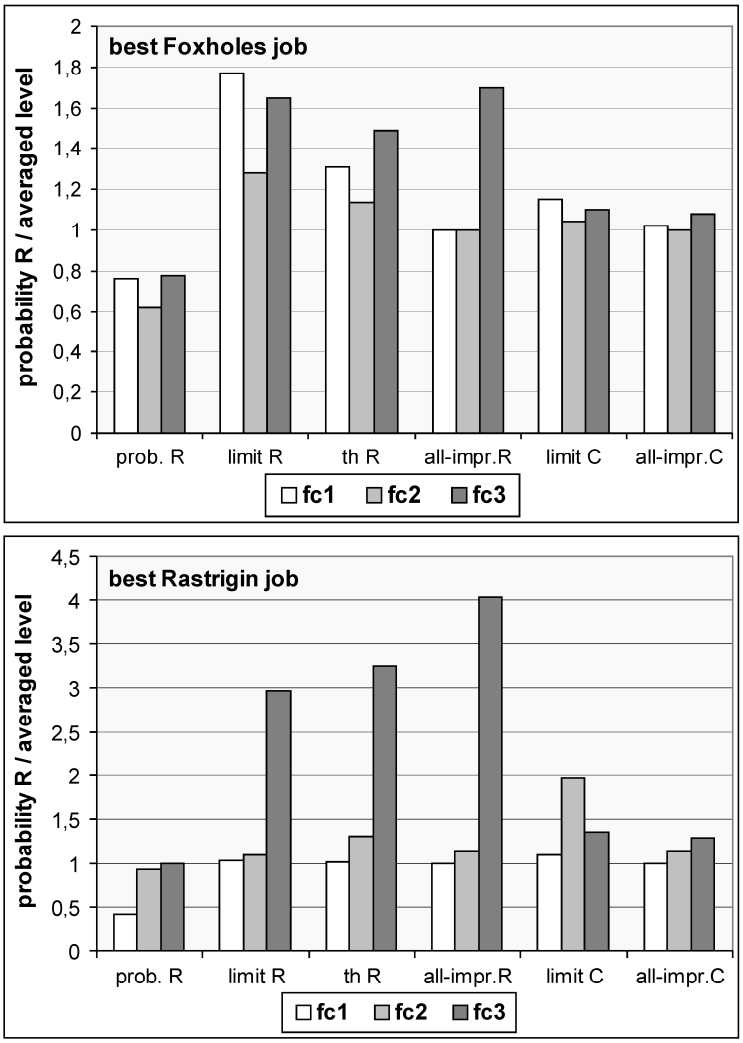

Bild 4: Exemplarische Entwicklung der Rosenbrockwahrscheinlichkeit (prob. R) und der Levels.

Beim Vergleich zwischen dem per Hand optimal eingestellten HyGLEAM und der Standardeinstellung von HyGLEAM/A (d.h. mit beiden LSV und all-Verbesserung) kann man beides erwarten, eine Verbesserung auf Grund der Adaption oder eine Verschlechterung

insbesondere dann, wenn ein nicht- oder schlecht geeignetes LSV abgeschaltet werden muss. So funktioniert der Complex-Algorithmus weder mit der Kugel-Funktion noch mit der Ressourcenoptimierung und mit der Rastrigin und der fraktalen Funktion schlechter als reines GLEAM. Letzteres trifft auch auf das Rosenbrock-Verfahren und die Designoptimierung zu. In der Tat lassen sich beide Effekte beobachten, wie Bild 5 zeigt, jedoch nicht bei allen Testfällen entsprechend ihrer Eignung für das eine oder andere LSV. So konnte z.B. im Falle der Kugel-Funktion trotz des ungeeigneten Complex-Verfahrens eine Verringerung der Evaluationen erreicht werden, während dies bei der Ressourcenoptimierung nicht möglich war. Offenbar ist es anwendungsabhängig, wie schnell ein ungeeignetes LSV verdrängt werden kann und ob es sozusagen nur „stört“ oder etwas zum Erfolg beiträgt, wenn auch mit höherem Aufwand.

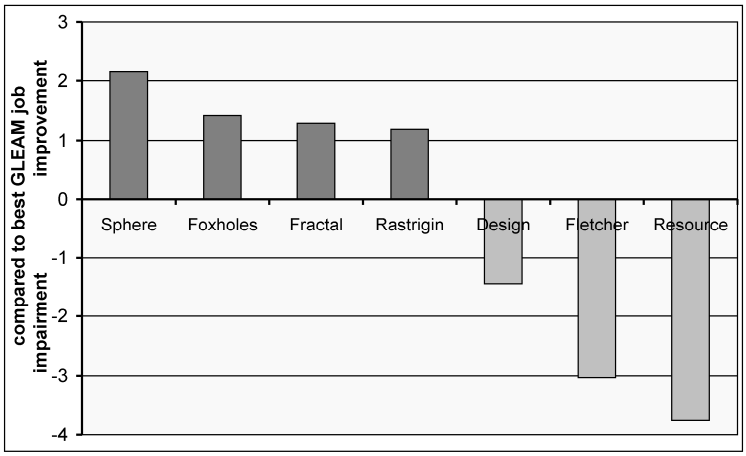

Bild 5: Verbesserungen und Verschlechterungen der besten Standard-HyGLEAM/A-Jobs verglichen mit den besten Jobs des per Hand optimierten HyGLEAMs.

Bei vier der sieben Testaufgaben (Foxhole, fraktale und Rastrigin Funktion sowie die Designoptimierung) bewirkt die Adaption keine relevante Leistungsänderung gegenüber HyGLEAM mit manueller Optimierung der Strategieparameter. Der beschleunigten Optimierung bei der Kugel-Funktion steht die zum Teil deutliche Verlangsamung bei Fletchers Funktion und bei der Ressourcenoptimierung gegenüber. Die beiden letzten fielen in Tabelle 5 durch herausragende Performancesteigerungen gegenüber GLEAM auf (Faktor 104 bzw. 77) und sie erreichen bei Standard-HyGLEAM/A immerhin noch eine Leistungssteigerung gegenüber GLEAM von einem Faktor von 35 bzw. 21 (siehe auch Tabelle 6), die ohne den bei HyGLEAM sonst notwendigen erheblichen manuellen Anpassungsaufwand für die Strategieparameter erreicht wurde.

Tabelle 6 fasst die Ergebnisse für die besten Jobs von GLEAM, manuell optimiertem HyGLEAM, Standard-HyGLEAM/A und dem besten HyGLEAM/A zusammen und vergleicht relevante Strategieparameter. Wenn Standard-HyGLEAM/A bereits am besten abschnitt, wurde in der Tabelle die Zeile für das beste HyGLEAM/A weggelassen. Aus der Tabelle lassen sich zwei Ergebnisse ablesen. Erstens fällt die deutliche und durchgängige Verringerung der Populationsgröße gegenüber GLEAM auf. Bild 6 zeigt ihren meist moderaten Einfluss auf den Aufwand für die fünf Test-

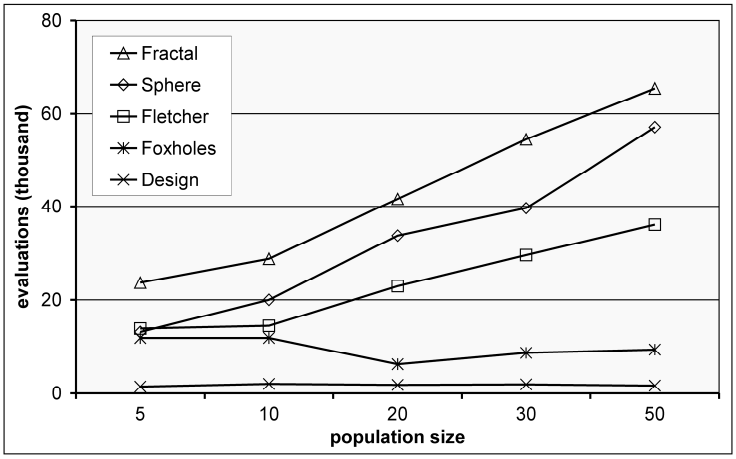

Bild 6: Einfluss der Populationsgröße auf den Aufwand für alle Testaufgaben, die auch bei kleinen Populationsgrößen eine Erfolgsrate von 100% aufweisen.

aufgaben, bei denen auch kleine Populationsgrößen eine Erfolgsrate von 100% liefern. Die Rastrigin Funktion konvergiert nicht mehr sicher ab einer Populationsgröße von 50 und weniger und die Ressourcenoptimierung ab 10 und weniger. Aber es kann als ein wesentliches Ergebnis festgehalten werden, dass der Bereich sinnvoller Populationsgrößen deutlich eingeengt werden konnte: Statt einem Wertebereich von 20 bis 11200 wie bei GLEAM kommt bei HyGLEAM/A nur noch ein Bereich zwischen 5 und 70 in Frage. Auf Grund des moderaten Einflusses steigender Populationsgrößen auf den Aufwand sollte man sicherheitshalber bei einem unbekannten Problem mit einer Populationsgröße von 20 beginnen und prüfen, ob eine Verringerung möglich ist. Wenn von einer erheblichen Komplexität auszugehen ist, empfehlen sich größere Populationen.

Testaufgabe	**Algorithmus**	**Evaluationen**	**Populationsgröße**	**Adaptionsgeschw.**	**Fitnessklasse**	**b/a**	**LSV**
Funktion nach Fletcher & Powell	GLEAM	483 566	600				
	HyG mit C	4 684	5				
	Stand.-HyG/A	13 841	5	schnell	fc2	all	beide
	Best HyG/A	11 808	10	schnell	fc2	best	beide
Ressourcenoptimierung	GLEAM	5 376 334	1 800				
	HyG mit R	69 448	5				
	Stand.-HyG/A	251 917	20	mittel	fc1	all	beide
	Best HyG/A	235 419	30	langsam	gc2	best	beide
Shekel's Foxholes	GLEAM	103 192	350				
	HyG mit C	8 831	20				
	Stand.-HyG/A	6 209	20	langsam	fc1	all	beide
verallg. Rastrigin Funktion	GLEAM	3 518 702	11 200				
	HyG mit R	315 715	70				
	Stand.-HyG/A	265 892	90	langsam	fc1	all	beide
Fraktale Funktion	GLEAM	195 129	20				
	HyG mit R	30 626	5				
	Stand.-HyG/A	23 649	5	schnell	fc2	all	beide
	Best HyG/A	20 753	5	mittel	fc1	all	beide
Designoptimierung	GLEAM	5 773	210				
	HyG mit C	1 041	5				
	Stand.-HyG/A	1 433	5	langsam	fc2	all	beide
	Best HyG/A	652	5	mittel	fc1	best	C
Schwefels Kugel	HyG mit R	29 595	5				
	Stand.-HyG/A	13 090	5	schnell	fc1	all	beide
	Best HyG/A	8 436	5	schnell	fc3	best	R

Tabelle 6: Ergebnisse der sieben Testfunktionen in absteigender Reihenfolge der Aufwandseinsparungen. Abkürzungen: HyG: HyGLEAM, Stand.-HyG/A: Standard-HyGLEAM/A, b/a: best- oder adaptive all-Verbesserung, weitere Abkürzungen siehe Tabelle 5.

Zweitens zeigt Tabelle 6 die Anwendungsabhängigkeit einer geeigneten Wahl der am Ende von Abschnitt 3.1 vorgestellten Strategieparameter der Adaption *Adaptionsgeschwindigkeit*, *Fitnessklasse* und *best/all-Verbesserung.* Obwohl auch die die *Menge der anwendbaren LSV (beide, Rosenbrock- oder Complex-Verfahren)* eine Rolle spielen kann, wie das Kugel-Problem und die Designoptimierung zeigen, soll aus Gründen der allgemeinen Anwendbarkeit die Wahl des LSVs der Adaption vorbehalten bleiben. Aus einer Analyse des gesamten Zahlenmaterials der Experimente kann folgendes abgeleitet werden:

1. Die Wahl der Fitnessklasse hat einen eher geringen Effekt.

2. Langsame und mittlere Adaptionsgeschwindigkeit funktioniert immer, während die schnelle Fehladaptionen vor allem hinsichtlich der LSV-Wahl mit sich bringen kann.
3. Die lokale Verbesserung nur des besten Nachkommen (best-Verbesserung) liefert zwar meist etwas bessere Resultate, ist aber im Falle der Rastrigin-Funktion fatal, da in diesem Falle nur all-Verbesserung zu einer Erfolgsrate von 100% führt.

Angesichts dessen und den moderaten Abweichungen bei den resultierenden Evaluationswerten in Tabelle 6 stellt sich die Frage, ob ein geeigneter „guter Mittelwert“ gefunden werden kann. Bild 7 vergleicht die besten zwei Parametrierungen jeweils für variable best/all-Verbesserung und für reine all-Verbesserung an Hand der erreichten Konvergenzgeschwindigkeit verglichen mit dem besten per Hand eingestellten HyGLEAM/A-Job. Der durch die vier Säulen rechts dargestellte durchschnittliche Leistungsverlust zeigt, dass die manuelle Wahl zwischen best- und all-Verbesserung die geringsten Performanceverluste bringt. Es wird aber auch deutlich, dass der Abstand zur besten Standard-HyGLEAM/A-Parametrierung gering ist und mit 82% der Leistung der besten manuell eingestellten HyGLEAM/A-Jobs nur geringe Leistungsverluste in Kauf genommen werden müssen.

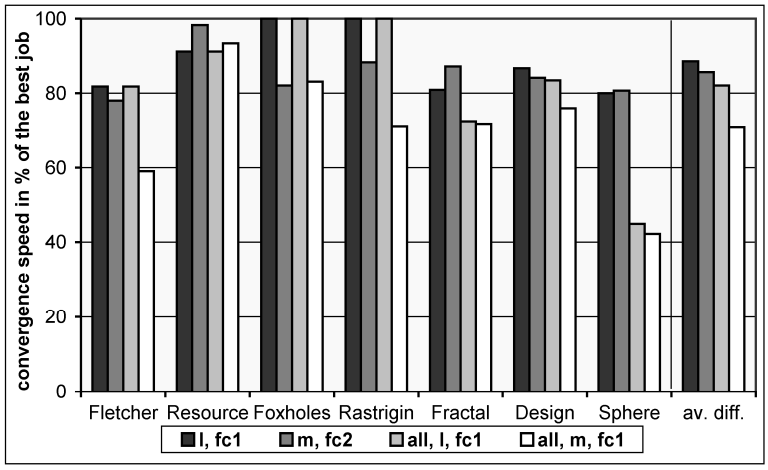

Bild 7: Vergleich der besten zwei Parametrierungen jeweils für manuell eingestellte best-/all-Verbesserung und für reine all-Verbesserung auf der Basis der erreichten Konvergenzgeschwindigkeit im Vergleich zum besten Job. Rechts der Durchschnitt für alle Testfälle (*av. diff.*). Abkürzungen: l, m: langsame, mittlere Adaptionsgeschwindigkeit

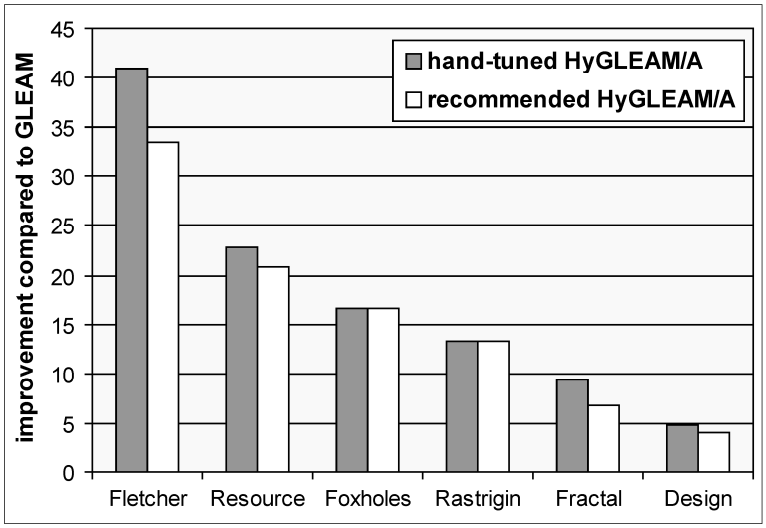

Bild 8: Vergleich des Leistungsgewinns gegenüber reinem GLEAM zwischen dem manuell eingestellten HyGLEAM/A und der empfohlenen Parametrierung von Standard-HyGLEAM/A.

Damit kann Standard-HyGLEAM/A mit langsamer Adaption, Fitnessklasse *fc1* bestehend aus drei Klassen und adaptiver all-Verbesserung sowie Verwendung beider LSV als die gesuchte günstige Parametrierung angesehen werden.

Bild 8 vergleicht die gegenüber GLEAM erreichten Performancegewinne der empfohlenen Parametrierung von Standard-HyGLEAM/A mit der besten manuellen Einstellung

von HyGLEAM/A für sechs der sieben Testaufgaben.[2] Das Bild verdeutlicht noch einmal den moderaten Leistungsverlust, der als Preis dafür zu zahlen ist, dass bis auf die Populationsgröße keine manuellen Einstellungen am resultierenden memetischen Algorithmus für Parameteroptimierungen mehr notwendig sind.

5 Zusammenfassung und Ausblick

Es konnte an Hand der sieben Testfälle gezeigt werden, dass das Ziel eines hybriden evolutionären Algorithmus mit möglichst wenig Verfahrensparametern durch den vorgeschlagenen Mechanismus zur Kosten-Nutzen basierten Adaption für memetische Algorithmen zur Parameteroptimierung nahezu erreicht werden konnte. Lediglich die Populationsgröße muss noch aufgabenspezifisch eingestellt werden, wobei der Bereich sinnvoller Größen deutlich eingeschränkt werden konnte, und zwar auf Werte zwischen 5 und 70 anstelle von 20 und 11200. Als beste Parametrierung der Adaption konnte folgendes ermittelt werden:

1. Verwendung beider lokaler Suchverfahren
2. Lokale Verbesserung eines adaptiv bestimmten Anteils aller Nachkommen einer Paarung (adaptive all-Verbesserung, siehe auch Abschnitt 3.1 und Tabelle 1)
3. Getrennte Adaption in drei Fitnessklassen (siehe auch Tabelle 2)
4. Langsame Adaptionsgeschwindigkeit (siehe auch Tabelle 3)

Die so eingestellte Adaption kontrolliert folgende Strategieparameter des Memetischen Algorithmus pro lokalem Suchverfahren: die Auswahl des lokalen Verfahrens, die Iterationsgrenze, eine konvergenzabhängige Abbruchschranke (nur beim Rosenbrock-Verfahren) und die Wahrscheinlichkeit, weitere Nachkommen lokal zu verbessern als nur das Beste einer Paarung (adaptive all-Verbesserung), siehe auch Tabelle 1. Mit der vorgeschlagenen Parametrierung konnte im Durchschnitt der Testfälle 82% der Leistungsfähigkeit (gemessen in Anzahl der Fitnessberechnungen) des besten manuell eingestellten adaptiven Hybriden erreicht werden.

Da der vorgestellte Mechanismus zu Adaption so konzipiert ist, dass er auch auf andere EA als dem hier verwendeten, anwendbar ist, ist man dem Ziel eines allgemein anwendbaren, robusten EA-Hybriden für Parameteroptimierung, der wenig Vorkenntnisse erfordert, ein großes Stück näher gekommen. Natürlich ist damit kein universelles Allzweckwerkzeug geschaffen, wie die eingangs gemachten Anwendungseinschränkungen zeigen und wie es auch im „no free lunch“-Theorem [27] zum Ausdruck kommt.

Da die Ergebnisse auf fünf Testfunktionen und zwei realen Anwendungen basieren, wäre eine Ausweitung der Testfälle zur besseren Absicherung der Resultate sicher sinnvoll. Wie die Arbeiten von Ong und Keane [20] gezeigt haben, ist auch von der Hinzunahme weiterer lokaler Suchverfahren eine Verbesserung zu erwarten.

6 Literatur

[1] Davis, L. (ed): *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991

[2] Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolutionary Programs*. 2nd ed., Springer, New York, USA, 1994

[2] Schwefels Kugel ist hier nicht enthalten, da GLEAM alleine das Problem nicht lösen konnte und damit kein Vergleichmaßstab verfügbar ist.

[3] Hart, W. E., Krasnogor, N., Smith, J. E. (eds), *Recent Advances in Memetic Algorithms*. Springer, Berlin, 2005
[4] Jakob, W.: *HyGLEAM – An Approach to Generally Applicable Hybridization of Evolutionary Algorithms*. In Conf. Proc. Parallel Problem Solving from Nature (PPSN VII), LNCS 2439, Springer, Berlin, 2002, S.527-536
[5] Jakob, W.: *Eine neue Methodik zur Erhöhung der Leistungsfähigkeit Evolutionärer Algorithmen durch die Integration lokaler Suchverfahren.* Dissertation, Fak. f. Maschinenbau, Universität Karlsruhe, FZKA 6965, Forschungszentrum Karlsruhe, März 2004. Siehe auch: http://www.iai.fzk.de/~jakob/HyGLEAM/
[6] Rechenberg, I.: *Evolutionsstrategie '94*. Frommann-Holzboog, Stuttgart-Bad Cannstatt, 1994
[7] Gruau, F., Whitley, D.: *Adding Learning to the Cellular Development of Neural Networks: Evolution and the Baldwin Effect*. Evolutionary Computation, Vol.1, 1993, S.213-233
[8] Whitley, D., Gordon, V., Mathias, K.: *Lamarckian Evolution, The Baldwin Effect and Function Optimization*. In: Davidor, Y. et al. (eds): Conf. Proc. PPSN III, LNCS 866, Springer Verlag, Berlin, 1994, S.6-14
[9] Gorges-Schleuter, M.: *Genetic Algorithms and Population Structures - A Massively Parallel Algorithm*. Dissertation, Dept. Comp. Science, University of Dortmund, 1990
[10] Gorges-Schleuter, M.: *A Comparative Study of Global and Local Selection in Evolution Strategies*. In Schwefel, H.-P., Männer, R. (eds.): Conf. Proc. PPSN V, LNCS 1498, Springer, Berlin, 1998, S.367-377
[11] Goldberg, D. E., Voessner, S.: *Optimizing Global-Local Search Hybrids*," in Conf. Proc. GECCO 99, Morgan Kaufmann, San Mateo, CA, USA, 1999, S.220-228
[12] Shina, A., Chen, Y., Goldberg, D. E.: *Designing Efficient Genetic and Evolutionary Algorithm Hybrids*. In [3], 2005, S.259-288
[13] Blume, C.: *GLEAM - A System for Intuitive Learning*. In: Schwefel, H.P., Männer, R. (eds): Conf. Proc. of PPSN I. LNCS 496, Springer Verlag, Berlin, 1990, S.48-54
[14] H.H. Rosenbrock: *An Automatic Method for Finding the Greatest or Least Value of a Function*. In: The Computer Journal, 3, 1960, S.175-184
[15] M.J. Box: *A New Method of Constrained Optimization and a Comparison with Other Methods*. In: The Computer Journal, 8, 1965, S.42-52
[16] H.-P. Schwefel: *Evolution and Optimum Seeking*. John Wiley & Sons, New York, USA, 1995
[17] H.J. Holland: *Adaptation in Natural and Artificial System*s. Dissertation, The University of Michigan Press, Ann Arbor, USA, 1975
[18] Hart, W. E..: *Adaptive Global Optimization with Local Search*. Dissertation, Univ. California, San Diego, CA, USA, 1994
[19] Krasnogor, N.: *Studies on the Theory and Design Space of Memetic Algorithms*. Dissertation, Faculty Comput., Math. Eng., Univ. West of England, Bristol, U.K., 2002
[20] Ong, Y. S., Keane, A. J.: *Meta-Lamarckian Learning in Memetic Algorithms*. IEEE Trans. on Evolutionary Computation, Vol. 8, Nr. 2, April 2004, S.99-110, Zitat S.100
[21] Krasnogor, N. , Blackburne, B. P., Burke, E. K., Hirst, J. D.: *Multimeme Algorithms for Protein Structure Prediction*. In Conf. Proc. PPSN VII, LNCS 2439, Springer, Berlin, 2002, S.769-778, Zitat S.772
[22] Zitzler, E., Teich, J., Bhattacharyya, S. S.: *Optimizing the Efficiency of Parameterized Local Search within Global Search: A Preliminary Study*. In Proc. Conference on Evolutionary Computation (CEC 2000), Piscataway, NJ, USA, IEEE press, 2000, S.365-372
[23] Bambha, N. K., Bhattacharyya, S. S., Teich, J., Zitzler, E.: *Systematic Integration of Parameterized Local Search Into Evolutionary Algorithms*. IEEE Trans. on Evolutionary Computation, Vol. 8, Nr. 2, April 2004, S.137-155
[24] Bäck, T.: *GENEsYs 1.0*, Uni. Dortmund, 1992, ftp://lumpi.informatik.uni-dortmund.de/pub/GA/
[25] Sieber, I., Eggert, H., Guth, H., Jakob, W., Scherer, K.-P., Ziegler, P.: *Design Optimization Considering Tolerance Effects of Microoptical Benches*. MicroSystem Technologies 98, VDE-Verlag, 1998, S.65-70
[26] Blume, C., Gerbe, M.: *Deutliche Senkung der Produktionskosten durch Optimierung des Ressourceneinsatzes*. In: atp 36, Nr. 5, 1994, S.25-29
[27] Wolpert, D. H., Macready, W. G.: *No free lunch theorems for optimization*. IEEE Trans. Evolutionary Computation, Vol. 1, April 1997, S. 67-82