CHIMIA 51 (1997) Nr. 1/2 (Januar/Februar)

Computer Chemistry

Chimia 51 (1997) 39–44

ISSN 0009–4293

Heuristik, Genetischer Algorithmus und andere Zufälligkeiten in der Computerchemie

Ivar K. Ugi, Martin J. Heilingbrunner und **Bernhard Gruber***

Gewidmet Prof. *Victor A. Snieckus* zum 60. Geburtstag

Abstract. Heuristic procedures have been used increasingly in recent years in synthesis planning, chemometrics and combinatorial chemistry. Applications of heuristic procedures have been presented in the chemical literature, but their use has been discussed only poorly. However, it has become evident that heuristic methods are not equally applicable to all problems. Indeed, the usefulness of the genetic algorithm can be deduced from the problem to be tackled: the most important prerequisite is that the quality does not play a significant role, *i.e.* a second best or even worse solution would be adequate and that the best solution is not necessary to solve the problem. Beyond that, the use of heuristics must be justified by the absence or high degree of complexity of non-heuristic methods.

1. Perspektive

'Algorithmen zur Lösung komplexer Probleme verbessert man häufig durch Strategien, die oft auf Hypothesen und Vermutungen aufbauen und die mit höherer Wahrscheinlichkeit (jedoch ohne Garantie) das Auffinden einer Lösung beschleunigen sollen. Solche Strategien heissen Heuristiken [1].'

Heuristische Algorithmen stützen sich auf Regeln, deren Gültigkeitsbereich unklar ist und deren Sinn im allgemeinen auch in keinem wissenschaftlichen Kontext erklärt ist. Ein gutes Beispiel für eine Heuristik ist das althergebrachte Sammelsurium an Bauernregeln, das sich für den Landwirt teils immer noch besser bewährt als die wissenschaftliche Meteorologie.

*Korrespondenz: Dr. rer. nat. B. Gruber
Lehrstuhl 1 für Organische Chemie
Institut für Organische Chemie und Biochemie
Technische Universität München
Lichtenbergstrasse 4
D-85747 Garching

Am Beispiel des Genetischen Algorithmus wird im folgenden die Einsetzbarkeit von Heuristiken gezeigt. Der Genetische Algorithmus ist ein heuristisches Verfahren, das auf der einzigen Regel basiert, dass die Kombination von guten Teillösungen wahrscheinlich eine gute Gesamtlösung ergibt. Ansonsten ist der Genetische Algorithmus mit der *Monte-Carlo*-Methode gleichzusetzen: Man suche solange zufällig nach einer Lösung, bis man sie gefunden zu haben glaubt. Es ist offensichtlich, dass derartige Verfahren nur in Ausnahmefällen sinnvoll anwendbar sind.

Die Erfahrung hat gezeigt, dass dort, wo analytische Verfahren in annehmbarer Zeit keine guten Lösungen zulassen, heuristische Algorithmen als Notlösung sehr wohl dazu in der Lage sein können. Heuristiken führen oftmals zu sehr ermutigenden Ergebnissen, nämlich sind sie immer dann ein wertvolles Hilfsmittel, wenn gute, aber nicht unbedingt optimale Lösungen für hochkomplexe Probleme in möglichst kurzer Zeit gefunden werden sollen. Heuristische Optimier- oder Suchverfahren liefern in manchen Bereichen relativ schnell Lösungen, die in der Nähe der bestmöglichen Lösung liegen.

Über die Ergebnisqualität heuristischer Algorithmen sind allerdings in der Regel keine Aussagen möglich. Heuristische Algorithmen sollten deshalb nur bei Problemstellungen Verwendung finden, bei denen andere Verfahren offensichtliche Mängel aufweisen, beispielsweise hinsichtlich des Rechenzeit- und Speicherplatzbedarfs. Sie sind hingegen völlig unbrauchbar, wenn eine Problemstellung nur optimale Lösungen zulässt.

1.1. *Chemische Anwendungen des Genetischen Algorithmus*

Die Anwendung heuristischer Rechenverfahren hat der computergestützten Bearbeitung chemischer Problemstellungen in den letzten Jahren neue und vielversprechende Wege eröffnet, insbesondere bei der Syntheseplanung [2][3] und in der Chemometrie [4]. Anhand von drei chemischen Beispielen wird gezeigt, dass je nach Problem der Genetische Algorithmus *a*) das einzig effektive, *b*) das effizientere und *c*) das ungeeignete Verfahren sein kann.

Eine sinnvolle Anwendung des Genetischen Algorithmus zur Steigerung der Effektivität ist beispielsweise die Berechnung niedrigenergetischer Konformationen von grossen Molekülen [5]. Im Rahmen der Kraftfeldmethode wird versucht, das Minimum der Konformationsenergie eines Moleküls zu finden. Die Kraftfeldfunktionen der meisten Moleküle sind so komplex, dass mit deterministischen Suchverfahren mit hoher Wahrscheinlichkeit lokale Minima gefunden werden. Der Genetische Algorithmus vermag die Hürden zwischen lokalen Minima zu überwinden und macht damit das globale Minimum auffindbar und die Suche effektiv.

Das Programm RAIN [6] benutzt den Genetischen Algorithmus zur Berechnung der chemischen Distanz zweier isomerer Molekülensembles. Für die Erzeugung eines Reaktionsnetzwerkes ist es von Belang, dass diese Distanz einen gewissen Grenzwert nicht überschreitet, um die Zahl der Verzweigungen in dem Netzwerk gering zu halten. Da dieses Problem sehr zeitaufwendig ist und kein exaktes Ergebnis benötigt wird, verwendet man den Genetischen Algorithmus.

Ein Beispiel für den inadäquaten Gebrauch von Heuristiken stellt die Suche nach maximalen gemeinsamen Substrukturen von zwei Molekülen dar [7]. Definitionsgemäss sind für dieses Problem nur exakte Lösungen brauchbar. Ein geeignetes Lösungsverfahren muss also die Aussage liefern, dass tatsächlich eine maximale gemeinsame Substruktur gefunden wurde. Im Gegensatz zu den hierzu existierenden deterministischen Verfahren wird genau das von heuristischen Verfahren jedoch *a priori* nicht erfüllt.

1.2. *Warum werden heuristische Verfahren eingesetzt?*

Es gibt zwei Gründe, weswegen man zu heuristischen Verfahren greift: Zum einen, wenn nicht genügend Wissen vorliegt, um einen axiomatischen Algorithmus zu formulieren. Zum andern, wenn ein Problem eine zu hohe Komplexität besitzt um mit einem axiomatischen Algorithmus gelöst werden zu können.

1.2.1. *Heuristik statt Deduktion*

Deduktiven Algorithmen liegen stets Theorien zugrunde. Eine Theorie besteht aus einer Sprache und einer Menge von Axiomen, die in dieser Sprache formuliert sind. Die Algorithmen werden aus den Axiomen der Theorie erzeugt.

Steht nur unzureichend Information für die Herleitung einer Theorie zur Verfügung, fehlt damit auch die Grundlage für einen deduktiven Algorithmus. In anderen Worten: Kennt man die Zusammenhänge der beobachteten Fakten nicht, lässt sich kein Verfahren finden, ein Problem, das sich aus den Beobachtungen ergibt, deduktiv zu lösen. In diesem Kontext steht der Begriff Heuristik synonym für 'nichts wissen'. Als einzige Alternative zum Einsatz von Heuristiken bleibt die Sammlung weiterer Fakten, bis sich eine Theorie herleiten lässt.

1.2.2. *Heuristik bei hoher Komplexität* [8]

Ein wesentlicher Aspekt der Programmentwicklung ist die Analyse des Speicher- und Zeitbedarfs einzelner Algorithmen. Der aufwendigste Algorithmus eines Programms ist dessen schwächstes Glied. Der Aufwand hängt im allgemeinen von den Eingabedaten ab. Eine Tabelle von Daten zu sortieren hängt beispielsweise von der Länge der Tabelle ab, zwei Vektoren elementweise zu addieren von der Dimension der Vektoren.

Grundsätzlich gibt es zwei Aspekte unter denen man den Speicher- und Zeitbedarf misst: Den grösstmöglichen (*worst case*) und den durchschnittlichen (*average case*) Bedarf. (Der dritte Aspekt *best case* wirkt sich nicht wesentlich auf das Verhalten des Programms aus.) *Worst case* und *average case* unterscheiden sich manchmal grundlegend, da die Werte der Parameter (Belegung) eine wesentliche Rolle spielen. Eine Tabelle zu sortieren, die bereits teilweise (oder ganz) sortiert ist, benötigt weniger Aufwand als eine Tabelle in totaler Unordnung.

Der Zeitbedarf [11] eines Algorithmus lässt sich als die Anzahl der benötigten primitiven Schritte definieren. Damit ist man von der verwendeten Maschine unabhängig. Gemäss der Abhängigkeit des Zeitbedarfs von den Eingabedaten lassen sich Algorithmen klassifizieren: Der Zeitbedarf bei elementweiser Addition von Vektoren steigt mit der Länge n der Vektoren *linear*, der Algorithmus ist damit *linear komplex*. Man spricht von Ordnung n, notiert als $O(n)$.

Analog gibt es Algorithmen, die logarithmisch, quadratisch oder auch exponentiell ansteigen und mit $O(\log n)$, $O(n^2)$ bzw. $O(n^n)$ bezeichnet werden (*Fig. 1*).

Man kann sich natürlich zu jedem Problem einen *ineffizienten* Algorithmus ausdenken. Grundsätzlich ist jedoch eine bestimmte Komplexität nicht zu unterschreiten, also probleminhärent. Es gibt sogar eine Klasse von Problemen, die sich der Lösungsfindung durch ihre Komplexität entzieht: Bei derartigen Problemen wachsen Zeit- und/oder Speicherplatzbedarf exponentiell mit der Grösse der Eingangsparameter [9]. Man hat wenig Auswahl bei den Mitteln, diese Probleme anzugehen, weswegen man heuristische Methoden an dieser Stelle daher oft bevorzugt.

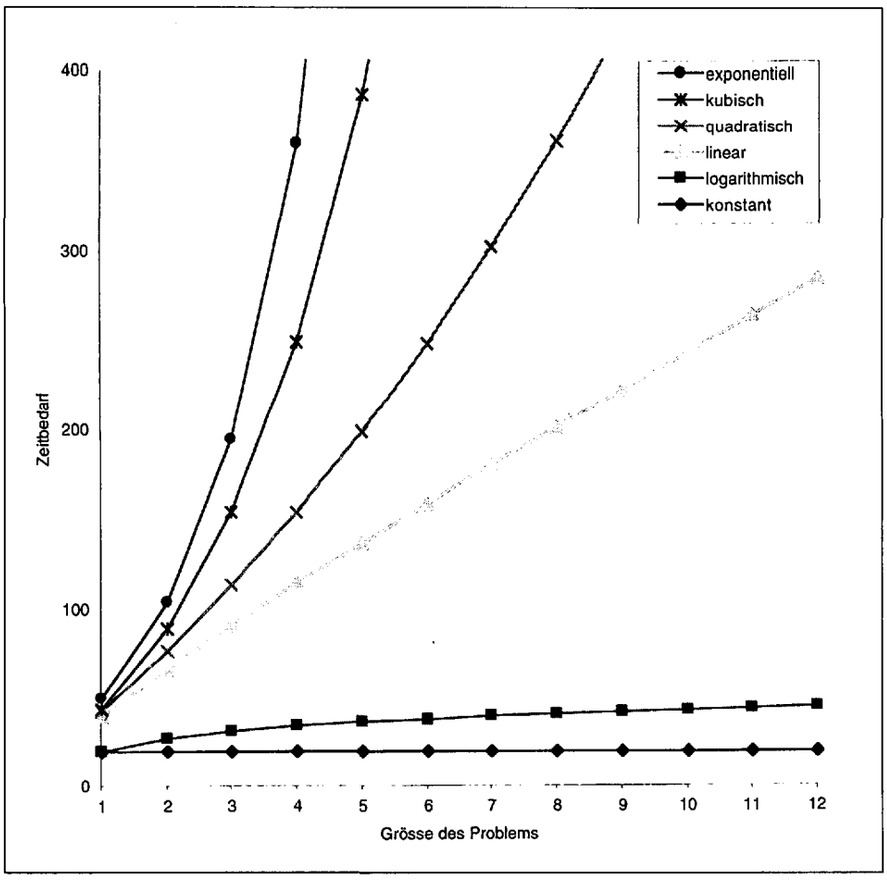

Fig. 1. *Die Komplexität eines Problems ist entscheidend für die Lösungsfindung.* Die Abbildung stellt den Zeitbedarf bei konstanter, logarithmischer, linearer, quadratischer, kubischer und exponentieller Komplexität gegenüber.

1.3. *Nützlichkeit heuristischer Verfahren*

Die im vorigen Abschnitt vorgebrachten Gründe für den Einsatz von Heuristiken sind in erster Linie effektivitätsorientiert: Würde man nicht Heuristiken verwenden, erhielte man in der verfügbaren Zeit (z. B. Lebensdauer des Experimentators) überhaupt kein Ergebnis.

Man kann Heuristiken jedoch auch zur Steigerung der Effizienz nutzen, indem man sie mit exakten Verfahren kombiniert. Ein typisches Beispiel sind Konformationsoptimierungen, bei denen globale energetische Minima erst durch den Genetischen Algorithmus angenähert und an-

CHIMIA 51 (1997) Nr. 1/2 (Januar/Februar)

schliessend mit Gradientenmethoden exakt berechnet werden (s. *Kap. 3*).

Essentiell für die Entscheidung für oder wider heuristische Verfahren bleibt die Beurteilung der Komplexität und der Güte bereits existierender Lösungsverfahren. Bei allen Problemstellungen, bei denen die Qualität der Lösung überprüfbar sein muss, gilt es heuristische Verfahren um jeden Preis zu vermeiden.

2. Der Genetische Algorithmus

Ein Wesensmerkmal heuristischer Algorithmen ist, dass über die Qualität der Ergebnisse nur relative Aussagen möglich sind, es sei denn, das optimale Ergebnis ist bereits im voraus bekannt. Analytische Begriffe wie Minimum/Maximum oder globales Minimum/globales Maximum sind im Zusammenhang mit einer rein heuristischen Optimierung nicht verwendbar.

Dennoch zeigt die Erfahrung, dass der Genetische Algorithmus als Optimier- oder Suchverfahren sehr schnell Lösungen liefern kann, die in der Nähe der bestmöglichen Lösung liegen. Je näher sich das gefundene Ergebnis bereits am (unbekannten) Optimum befindet, um so länger dauert eine weitere Verbesserung, da zufällige Schritte im Suchraum mit steigender Wahrscheinlichkeit zu relativ schlechteren Lösungen führen, je besser die bereits gefundene ist (Gesetz des abnehmenden Grenznutzens). Der Genetische Algorithmus sollte deshalb nur bei Problemen eingesetzt werden, bei denen neben der besten Lösung andere Lösungen (geringfügig) schlechterer Güte von Interesse sind (z. B. Optimierung von reellwertigen Parametern in der Konformationsanalyse) [12].

Die auf diese Weise gefundenen 'fast optimalen' Lösungen können anschliessend mit analytischen Verfahren nachbearbeitet werden. Für Probleme, bei denen nur die *exakte* Lösung von Interesse ist, ist der Genetische Algorithmus nicht geeignet, da er dies nicht leisten kann.

2.1. *Funktionsweise*

Der Genetische Algorithmus erzeugt aus einer Menge von Genomen, einer sogenannten Population, nach einem evolutionären Prinzip eine neue Population [13]. Jedes Genom besteht aus einzelnen Genen und wird als Lösung des Problems interpretiert. Dabei entspricht jedes Gen einem Teilaspekt der Lösung. Beispielsweise können bei der Konformationsoptimierung die Gene als Bindungswinkel, Bindungslängen usw. interpretiert werden.

Im Laufe mehrerer Generationen erzeugt der Genetische Algorithmus durch Paarung, Mutation und Selektion zunehmend Populationen mit besseren Individuen, so dass schliesslich überwiegend 'gute' Lösungen vorliegen (*Fig. 2*).

Die interpretierten Genome werden mit einer Fitnessfunktion bewertet. Dadurch ist die Möglichkeit gegeben, die relativ besseren Lösungen zu erkennen. Diese dienen als Elternpopulation für die nächste Generation.

Als *Paarung* (Rekombination) bezeichnet man die zufallsgesteuerte Erzeugung neuer Individuen unter Verwendung von Genen mindestens zweier Individuen der Elternpopulation. Es werden beliebige Genome der Elternpopulation gewählt. Die Elternpopulation bleibt Bestandteil der neuen Generation. Erst durch die Selektion können Genome dieser Population verloren gehen. Aus der Paarung zweier Genome G_a und G_b der Elternpopulation resultiert eine zufällige Kombination: Mit je 50prozentiger Wahrscheinlichkeit wird Gen i des neuen Genoms den Wert von Gen i des Genoms G_a oder den Wert von Gen i des Genoms G_b übernehmen. Prinzipiell können auch andere Arten der Gen-Verknüpfung gewählt werden, z.B. die Berechnung des Durchschnitts der Gene. Der statt der Paarung häufig verwendete Operator *Crossover* beschreibt den zufallsgesteuerten Austausch von Genen zwischen zwei Individuen. Der Effekt ist der gleiche wie bei der Paarung. Mathematisch besteht kein Unterschied [14].

Mutation ist die zufällige Veränderung eines Gens. Die Operatoren Paarung und Mutation bewirken, dass sich die Population in jeder Generation aus möglichst vielfältigen Individuen zusammensetzt.

Die *Selektion* braucht ein Kriterium, nach dem entschieden wird, *welche* Genome als nächste Elternpopulation dienen. Eine Ordnungsfunktion legt mittels einer Genombewertungsfunktion die Rangfolge der Genome fest. Es werden alle Genome in die nächste Generation übernommen, die einer gewissen Selektionsstrenge genügen.

Inwieweit dient dem Genetischen Algorithmus die Evolution als Vorbild? Die der Evolution immanenten Gesetzmässigkeiten bewirken eine Anpassung der Spezies an die Umweltbedingungen. Die Weiterentwicklung ist ein durch Resourcenkonkurrenz der Spezies hervorgerufenes Nebenprodukt. Eine Optimierung ähnelt jedoch eher der Aufgabe eines Tierzüchters, der in Kenntnis der natürlichen Anpassungsfähigkeit einer Spezies von Aussen ein Optimierungsziel vorgibt und die Selektion entsprechend durchführt. Insofern stellt der Genetische Algorithmus kein evolutionäres Verfahren dar, sondern eine Züchtung auf der Basis evolutionärer Mechanismen. Ähnlich wie der Züchter sein Optimierungsziel wesentlich schneller erreicht als die Evolution Verbesserungen hervorbringt, optimiert auch der Genetische Algorithmus schneller als eine Evolutionssimulation.

Mühlenbein hat bereits 1993 eine auf diesen Erkenntnissen beruhende Implementation des Genetischen Algorithmus

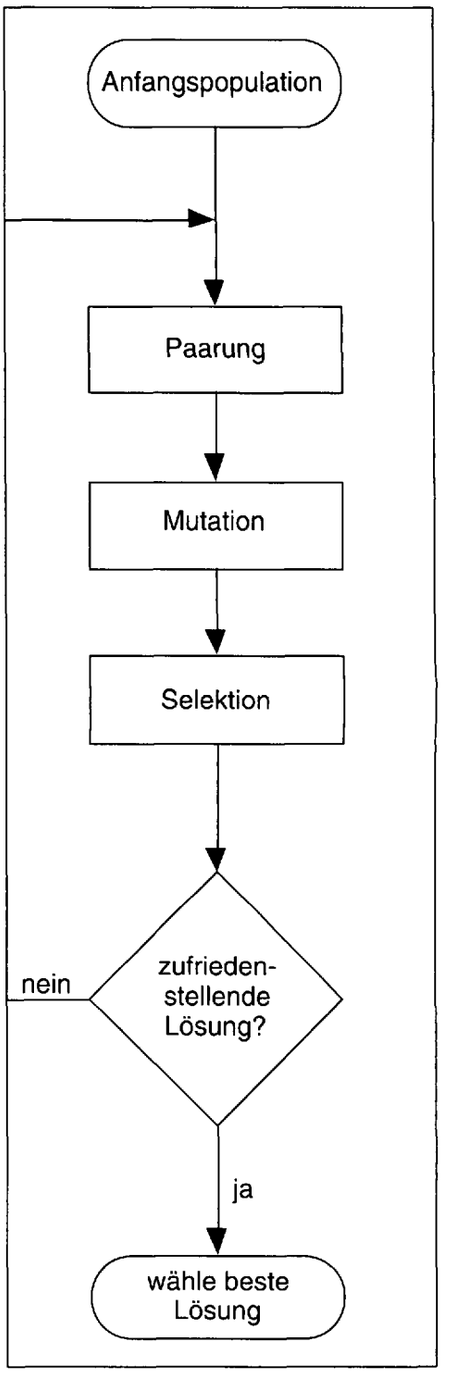

Fig. 2. *Ablaufschema des Genetischen Algorithmus*

CHIMIA 51 (1997) Nr. 1/2 (Januar/Februar)

(den BGA, breeder genetic algorithm) vorgestellt [14]. Bislang durchgeführte Untersuchungen zeigen, dass Selektion, Paarung und Mutation in Verbindung mit Gradientenmethoden die effizienten und effektiven Operatoren des Genetischen Algorithmus darstellen und auf zusätzliche Operatoren, insbesondere auf die im Zusammenhang mit der *building-block*-Hypothese [15] beschriebenen Operatoren *Umordnung* des Genoms, *Translokation* und *Segregation* [13], verzichtet werden kann [16–18].

Problemspezifische (häufig deterministische) Optimieroperatoren (auch *Hill-Climber* genannt [14]) fand man früher oft als Operatoren des Genetischen Algorithmus. Tatsächlich stehen diese Verfahren funktionstechnisch auf der selben Stufe wie der Genetische Algorithmus: Durch einen übergeordneten Entscheidungsprozess wird je nach Situation mal das eine, mal das andere Verfahren aufgerufen. Zusätzliche problemspezifische Optimierungsverfahren machen keinen *neuen* Genetischen Algorithmus, denn der Genetische Algorithmus ist lediglich in das Gesamtverfahren eingebettet. Problemspezifische Genetische Operatoren sind strikt unnötig.

2.2. *Formale Definition des Genetischen Algorithmus*

Lösungen eines Problems lassen sich als Eigenschaftsvektoren beschreiben: die Konformation eines Moleküls beispielsweise als Vektor von Winkeln und Bindungslängen. Jeder dieser Eigenschaftsvektoren entspricht einem Genom. Jede Eigenschaft korreliert mit einem Gen des Genoms. Ein Genom G, das einem Vektor e von n Eigenschaften entspricht, ist somit ein Vektor der Dimension n von Genen.

Bei dem Genetischen Algorithmus GA handelt es sich um eine Funktion höherer Ordnung, da das zu lösende Problem im allgemeinen eine Funktion darstellt und Parameter des Genetischen Algorithmus ist [19].

Neben der Anfangspopulation $p_0 \in 2^P$ muss dem Genetischen Algorithmus für die Selektion eine Funktion $\gamma: P \rightarrow \mathbb{N}$ zur Verfügung gestellt werden, die die Genome der Population bewertet. Diese Genombewertungsfunktion setzt sich aus der Interpretationsfunktion und der Fitnessfunktion zusammen. Beide Funktionen sind problemspezifisch. Der Genetische Algorithmus erzeugt aus der n-ten Population p_n die n+1-te Population, indem er sequentiell die Paarung C, die Mutation M mit Mutationsrate w_M und die Selektion S mit Selektionsstrenge w_S ausführt:

$$GA: W^2 \times 2^P \times \mathbb{N} \times [P \times \mathbb{N}] \rightarrow 2^P$$
$$GA(w_S, w_M, p, q, \gamma) = S(w_S, M(w_M, C(p, m)), \gamma)$$
$$P_{n+1} = GA(w_S, w_M, p_n, q, \gamma)$$

Die in den Eigenschaftsvektoren enthaltene Information geht in den Genetischen Algorithmus ausschliesslich in Form der Genombewertungsfunktion ein.

Der Genetische Algorithmus benötigt als nichtdeterministisches Verfahren auch nichtdeterministische Operatoren, die aus einer Population p gleichverteilt beliebige Genome auswählen. Des weiteren wird u. a. für die Mutation eine nichtdeterministische *Boole*sche Funktion D benötigt, die gemäss der Belegung eines Parameters $w \in [0;1]$ L oder O liefert. Der Verlauf ist linear: $D(w)$ liefert mit Wahrscheinlichkeit w den Wert L. Insbesondere liefert $D(0)$ mit 0prozentiger Wahrscheinlichkeit L und $D(1)$ mit 100prozentiger Wahrscheinlichkeit L. Bei der Mutation ist der Parameter w die Mutationsrate w_M.

2.3. *Parametrisierung statt Variation des Genetischen Algorithmus*

Eine Trennung des zu lösenden Problems vom Genetischen Algorithmus ist notwendig, da mit dem selben Genetischen Algorithmus eine Vielzahl von Problemen effizient lösbar ist. Das Problem selbst ist lediglich ein Parameter des Genetischen Algorithmus. Durch Interpretation erhalten Genome ihre Bedeutung.

Um eine Trennung von Genetischem Algorithmus und Optimierungsproblem zu erreichen, müssen Gene eindeutig in Eigenschaften abgebildet werden. Jedes Gen ist mit einem Wert aus einer Definitionsmenge belegt, wobei die Grösse der Definitionsmenge von den Details der Implementation des Genetischen Algorithmus abhängt. Die Operatoren des Genetischen Algorithmus dürfen die Gene beliebig verändern. Durch die Interpretation in eine Eigenschaft erhält jedes Gen seine Bedeutung für ein spezielles Problem. Dies ist im Laufe jeder Generation während der Fitnessberechnung erforderlich. Im Rahmen der Interpretation ist der Definitionsbereich der Eigenschaften festgelegt. Dies hat den Vorteil, dass Gene innerhalb des Genetischen Algorithmus beliebig manipuliert werden dürfen und dennoch immer Eigenschaften repräsentiert werden. Es ist auch keineswegs erforderlich, die Gene binär zu 'codieren', denn der Wert eines Gens, d.h. die Information, die ein Gen repräsentiert, ist unabhängig von der Art der Repräsentation dieser Information.

Ein Optimierungsproblem, beschrieben als Interpretationsfunktion und Fitnessfunktion, ist also ein funktionaler Parameter des Genetischen Algorithmus. Der Arbeitsaufwand, der für Problemlösungen mittels Genetischem Algorithmus notwendig ist, beschränkt sich somit auf den Entwurf einer geeigneten Interpretationsfunktion und einer Fitnessfunktion. Es ist also nicht erforderlich, einen Genetischen Algorithmus selbst zu implementieren, sondern sinnvoll, die *via* Internet zugänglichen Implementierungen [20] zu nutzen.

3. Anwendungen des Genetischen Algorithmus in der Chemie

3.1. *Konformationsanalyse*

Um die Konformere eines Moleküls zu bestimmen, müssen Geometrien minimaler Konformationsenergie gefunden werden. Das stabilste Konformer entspricht dem globalen energetischen Minimum des Konformationsraumes.

Ausgangspunkt der Konformationsoptimierung mit analytischen Verfahren ist eine Startkonformation, die unter Berücksichtigung des Gradienten des Force-Field iterativ verbessert wird bis der Gradient minimal ist [21]. Eine wesentliche Limitierung dieser Verfahren ist ihre prinzipielle Unfähigkeit, energetische Barrieren zu überwinden. Aufgrund der grossen Komplexität der Konformationsräume der meisten Moleküle ist die Wahrscheinlichkeit, nur lokale Minima zu finden sehr gross. Aus diesem Grund muss eine Minimumsuche mit verschiedenen Startkonformationen wiederholt werden. Das erste und entscheidende Problem rein analytischer Verfahren ist es, eine genügend grosse Anzahl verschiedener aussichtsreicher Startkonformationen zu erzeugen [22–29].

In der Begriffswelt analytischer Verfahren stellt eine Konformationsanalyse mit einem Genetischen Algorithmus eine Startkonformationssuche mit anschliessender Voroptimierung (Prefit) in einem Schritt dar. Während jedes Ablaufs des Genetischen Algorithmus werden neue Startkonformationen auf der Basis der besten bisherigen Ergebniskonformationen gebildet.

Die berechnete Konformationsenergie kann direkt als Fitnesswert verwendet werden [30]; es können jedoch auch andere aus einem Molekül abgeleitete abstraktere, z. B. topologische Informationen zur

Bestimmung der Güte einer Konformation herangezogen werden [31].

Da zur Fitnessberechnung keine Differentiale des Kraftfelds benötigt werden, erfordert die Konformationsoptimierung mit dem Genetischen Algorithmus erheblich weniger Rechenzeit. Es werden also mit relativ geringem Rechenzeitaufwand gute Startkonformationen erzeugt. Dies impliziert auch, dass die Vorteile der bisherigen Verfahren in einem nachfolgenden analytischen Optimierungsschritt besser genutzt werden können.

3.2. *Der Genetische Algorithmus zur Bestimmung der minimalen chemischen Distanz*

Das Computerprogramm RAIN (*R*eactions *A*nd *I*ntermediates *N*etwork) [2] erzeugt zwischen einem Eduktensemble und einem Produktensemble Reaktionspfade. Dazu werden mit einem Reaktionsgenerator von beiden Ensembles von Molekülen ausgehend isomere Ensembles erzeugt. Sobald die aus den beiden Richtungen kommenden Reaktionspfade sich treffen, ist ein Reaktionspfad zwischen dem Eduktensemble und dem Produktensemble gefunden.

Um sicherzustellen, dass sich die Reaktionspfade zwischen Edukt- und Produktensemble aufeinander zubewegen, werden Molekülensembles bevorzugt, die eine geringere chemische Distanz zum Produktensemble aufweisen als zum Eduktensemble (und umgekehrt für den 'retrosynthetischen' Pfad geringere Distanz zum Eduktensemble). Dazu muss bei jedem neuen Zwischen-Ensemble die minimale chemische Distanz zwischen dem Zwischen-Ensemble und dem Produktensemble bestimmt werden. Dieses Problem ist NP-vollständig [32], also sehr zeitaufwendig.

Die mit dem Genetischen Algorithmus ermittelte Distanz ist stets grösser oder gleich der minimalen chemischen Distanz der beiden Molekülensembles und somit eine Obergrenze der tatsächlichen Distanz. Liegt sie unterhalb des Grenzwertes, unterschreitet diesen auch die minimale chemische Distanz. Die Verwendung des Genetischen Algorithmus in RAIN bedeutet den Verzicht auf die Vollständigkeit der Menge der Lösungen und nichtdeterministische Ergebnisse zugunsten höherer Effizienz. Existierende deterministische Verfahren benötigen deutlich mehr Zeit.

Gute, aber nicht optimale Ergebnisse sind also im Rahmen der Methode zulässig. Somit sind beide Kriterien für den sinnvollen Einsatz des Genetischen Algorithmus gegeben.

3.3. *Suche der maximal gemeinsamen Substruktur zweier Moleküle* (MCSS)

Das Problem der Suche von maximalen gemeinsamen Substrukturen zweier beliebiger Graphen, insbesondere der global grössten Substruktur, ist ausgesprochen zeitaufwendig. Chemische Graphen unterliegen jedoch einigen Einschränkungen. Die Konnektivität eines Knotens ist endlich und liegt für organische Moleküle im Durchschnitt bei etwa 2,3. Die Knoten chemischer Graphen sind Atome, d. h. es sind gefärbte Knoten. Die Kanten chemischer Graphen sind chemische Bindungen mit unterschiedlichen Bindungsordnungen, d. h. auch die Kanten sind gefärbt. Nur gleich gefärbte Knoten und Kanten können jeweils aufeinander abgebildet werden. Für die Bildung eines Ringes sind mindestens drei Knoten erforderlich.

All diese Einschränkungen verkleinern den Suchraum einer Substruktursuche erheblich. Die durchschnittliche Komplexität der Substruktursuche ist bei chemischen Graphen daher polynomial (Grad $\leq$ 5) [33]. Dies erklärt auch, warum die Vielzahl der bisher entwickelten analytischen Verfahren zur Substruktursuche erfolgreich waren [34], darunter als Ausgangspunkt zahlreicher Folgearbeiten das Programm CORREL von *Friedrich* [35]. Nichtsdestotrotz wurde in einer Arbeit ein Verfahren zur Suche der MCSS mittels des Genetischen Algorithmus beschrieben [7][36].

Bei der Frage, ob hier ein heuristischer Lösungsansatz eine Verbesserung gegenüber den zahlreichen deterministischen Verfahren bietet, sind die Ergebnisqualität und die Ausführungsgeschwindigkeit in Betracht zu ziehen. Ein Verfahren zur Suche der MCSS muss natürlich diese auch finden. Diesbezüglich ist der Genetische Algorithmus ineffektiv. Verbesserungen sind also ausschliesslich hinsichtlich der Geschwindigkeit zu erwarten, mit der die MCSS gefunden wird.

Weil man bei dem Genetischen Algorithmus nicht davon ausgehen kann, dass das bestmögliche Ergebnis gefunden wird, sind eine hinreichend grosse Anzahl von Wiederholungen ein und desselben Experiments notwendig, um mit grösserer *Wahrscheinlichkeit* das beste Ergebnis zu finden. Für Moleküle mit etwa 25 Atomen werden 50 Wiederholungen als ausreichend angegeben [7]. Es ist jedoch zu erwarten, dass die Anzahl der notwendigen Wiederholungen mehr als linear mit der Anzahl der Atome der zu untersuchenden Strukturen ansteigt. Hierüber wurden in [7] keine Angaben gemacht.

Selbst wenn man davon ausgeht, dass ein einzelner Durchlauf einer MCSS-Suche mit dem Genetischen Algorithmus tatsächlich schneller ist als andere Verfahren, steht doch zu erwarten, dass das Verfahren insgesamt deutlich langsamer als bereits existierende Verfahren ist. Dies gilt insbesondere für Suchen in chemischen Strukturdatenbanken, für die sogar hardwareoptimierte Verfahren existieren, die sich auch in der Praxis bewährt haben [34].

4. Schlussbemerkung

Die alte Einsicht, dass man bei Problemen bei denen ausschliesslich die beste Lösung von Interesse ist, mit rein heuristischen Methoden auf keinen grünen Zweig kommt, scheint also auch in der Chemie Bestätigung gefunden zu haben. Die Zulässigkeit annähernd bester Ergebnisse bleibt für die sinnvolle Anwendung des Genetischen Algorithmus die *conditio sine qua non*. Die Verwendung des Genetischen Algorithmus in der Chemie ist ein selbstverständliches Vorgehen wie die Addition zweier Zahlen und braucht nicht bei jedem Einsatz erneut diskutiert werden. Die wissenschaftliche Arbeit liegt ausschliesslich in der Klassifizierung des vorliegenden Problems. Erkennt und akzeptiert man die Eigenschaften des Genetischen Algorithmus, werden auch seine Anwendbarkeit und Grenzen offensichtlich.

Wir danken der DFG für die Unterstützung unseres Projekts Ug-1/73-1 'Konformationsanalyse mit einem Genetischen Algorithmus' und Herrn *Oliver Kern* für die englische Kurzfassung.

Eingegangen am 24. Oktober 1996

[1] H. Engesser, Ed., V. Claus, A. Schwill, Bearb., Duden Informatik, Dudenverlag Mannheim, Wien, Zürich, 1988.
[2] E. Fontain, *Anal. Chim. Acta* **1992**, *265*, 227.
[3] I. Ugi, J. Bauer, K. Bley, A. Dengler, A. Dietz, E. Fontain, B. Gruber, R. Herges, M. Knauer, K. Reitsam, N. Stein, *Angew. Chem.* **1993**, *105*, 210; *ibid. Int. Ed.* **1993**, *32*, 201.
[4] R. Wehrens, C. Lucasius, L. Buydens, G. Kateman, *J. Chem. Inf. Comput. Sci.* **1993**, *33*, 245.
[5] M. Heilingbrunner, 'Optimierung molekularer Konformation mit einem genetischen Algorithmus', Diplomarbeit, Technische Universität München, München, 1993.
[6] E. Fontain, K. Reitsam, *J. Chem. Inf. Comput. Sci.* **1991**, *31*, 96.
[7] M. Wagener, J. Gasteiger, *Angew. Chem.* **1994**, *106*, 1245; *ibid. Int. Ed.* **1994**, *33*, 1189; ausführlicher in: M. Wagener, J. Gasteiger, in 'Software-Entwicklung in der Chemie 7, Proceedings of the 7th Work-

shop 'Computer in Chemistry'', Ed. D. Ziessow, Gosen/Berlin, 1993, p. 153.

[8] Der Begriff 'Komplexität' ist ausführlich in [9] und an praktischen Beispielen in [11] behandelt.

[9] J.E. Hopcroft, J.D. Ullman, 'Introduction to Automata Theory, Languages, and Computation', Addison-Wesley, Reading, Mass., 1979.

[10] Es ist möglich, den Speicherbedarf eines Algorithmus auf Kosten des Zeitbedarfs zu reduzieren und umgekehrt [9], p. 285.

[11] K. Mehlhorn, Datenstrukturen und effiziente Algorithmen, 2. Aufl.,Teubner, Stuttgart, 1988.

[12] I. Ugi, M. Heilingbrunner, B. Gruber, *Nachr. Chem. Tech. Lab.*, eingereicht.

[13] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, Mass., 1989.

[14] H. Mühlenbein, D. Schlierkamp-Voosen, *Evolutionary Computation* **1993**, *1*, 25.

[15] J. H. Holland, Technical Report ORA Projects 01252 and 08226, University of Michigan, Michigan, 1968.

[16] H. Mühlenbein, D. Schlierkamp-Voosen, *Neural Network World* **1993**, *3*, 907.

[17] H. Mühlenbein, *Der GMD-Spiegel* **1995**, *2*, 12.

[18] H. Mühlenbein, in 'Foundations of Genetic Algorithms', Ed. G. Gawlins, Morgan-Kaufman, San Mateo, 1991, p. 316.

[19] Mit *IN* wird im folgenden die Menge der Natürlichen Zahlen bezeichnet; mit *W* die Menge der Reellen Zahlen im Bereich [0;1]; mit *P* die Multimenge (Mengen, in der Elemente mehrfach enthalten sein können) aller Genome; mit 2^P die Menge aller Genommultimengen über *P* (Potenzmenge).

[20] Erhältlich *via* Internet, ftp/anonymous bei ftp.aic.nrl.navy.mil im Verzeichnis /pub/galist/src.

[21] M. Saunders, K.N. Houk, Yun-Dong Wu, C. W. Still, M. Lipton, G. Chang, C.W. Guida, *J. Am. Chem. Soc.* **1990**, *112*, 1419.

[22] N.L. Allinger, V. Burkert, 'Molecular Mechanics', ACS Monograph *177*, ACS, Washington, 1982.

[23] M. Lipton, C.W. Still, *J. Comput. Chem.* **1988**, *9*, 343.

[24] R. Bruccoleri, M. Karplus, *Biopolymers* **1987**, *27*, 137.

[25] R.A. Dammkoehler, S.F. Karasek, E.F.B. Schands, G.R. Marshall, *J. Computer-Aided Mol. Design* **1989**, *3*, 3.

[26] G. Chang, C.W. Guida, C.W. Still, *J. Am. Chem. Soc.* **1989**, *11*, 4379.

[27] W.F. van Gunsteren, H.J.C. Berendsen, *Angew. Chem. Int. Ed.* **1990**, *29*, 992.

[28] H.J. C. Berendsen, J.P.M. Postma, W.F. van Gunsteren, A. DiNola, J.R. Haak, *J. Chem. Phys.* **1984**, *81*, 3684.

[29] J.P. Ryckaert, G. Ciccotti, H.J.C. Berendsen, *J. Comput. Phys.* **1977**, *23*, 327.

[30] D.B. McGarrah, R.S. Judson, *J. Comput. Chem.* **1993**, *14*, 1385; R.S. Judson, E.P. Jaeger, A.M. Treasurywala, M.L. Peterson, *ibid.* **1993**, *14*, 1407; R.S. Judson, *J. Phys. Chem.* **1992**, *96*, 10102.

[31] T. Dandekar, P. Argos, *J. Mol. Biol.* **1994**, *236*, 844; T. Dandekar, P. Argos, *Protein Engineering* **1992**, *5*, 6375.

[32] E. Fontain, *J. Chem. Inf. Comput. Sci.* **1992**, *32*, 748.

[33] I. Ugi, J. Bauer, C. Blomberger, J. Brandt, A. Dietz, E. Fontain, B. Gruber, A. von Scholley-Pfab, A. Senff, N. Stein, *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 3.

[34] J.B. Barnard, *J. Chem. Inf. Comput. Sci.* **1993**, *33*, 532.

[35] J. Friedrich, I. Ugi, *J. Chem. Res., Synop.* **1980**, 70; *J. Chem. Res. Miniprint* **1980**, 1301.

[36] Die hierin formulierten 'problemspezifischen Operatoren' Creep und Crunch sind ein Beispiel für lokale deterministische Optimierer und als solche nicht Bestandteil des Genetischen Algorithmus, sondern funktionale Parameter desselben.