Evolution als Optimierungsprozeß - Prinzip und Anwendung beim Entwurf elektronischer Schaltungen

Andreas Rummler **Gerd Scarbata**

Technische Universität Ilmenau
Institut für elektronische Schaltungen und Systeme
98684 Ilmenau
e-mail : arummler@acm.org
e-mail : gerd.scarbata@inf-technik.tu-ilmenau.de

Zusammenfassung

Dieser Artikel beschreibt die Grundlagen und die Anwendung von Optimierungsverfahren basierend auf dem Prinzip der biologischen Evolution. Nach einer kurzen Einleitung werden im zweiten Abschnitt grundlegende Prinzipien dargelegt, die zum weiteren Verständnis notwendig sind. Anhand des Beispiels der Plazierung von Standardzellen wird das Prinzip im dritten Abschnitt veranschaulicht. Darüberhinaus werden grundlegende Eigenschaften diskutiert, die alle evolutionären Algorithmen auszeichnen. Abschließend wird ein kurzer Ausblick auf weitere Entwicklungen und mögliche Anwendungen auf diesem Gebiet gegeben.

1 Einleitung

Problemstellungen aus den Gebieten der Entwurfsautomatisierung und Verifikation elektronischer Schaltungen fallen zunehmend in Komplexitätbereiche, bei denen konventionelle Lösungverfahren versagen. Ohne Frage müssen neue Wege beschritten und neue Algorithmen entwickelt werden. Ein Optimierungsverfahren oder genauer eine Klasse von Verfahren neueren Datums ist simulierte Evolution.

Charles Darwin erkannte die Bedeutung der Evolution in der modernen Biologie als erster. Nach [Dar59] beruht Evolution auf einigen wenigen, relativ einfachen Prinzipien wie Auswahl von Lebewesen (Selektion), Fortpflanzung (Rekombination) und Tod. Individuen mit genetischen Vorzügen sind besser an ihre Umwelt angepaßt (Adaption), haben aus diesem Grund bessere Überlebens- und Fortpflanzungchancen und sind in der Lage diese Vorzüge an Nachkommen zu vererben. Dieser Prozeß läuft seit der Entstehung des Lebens auf der Erde ab.

Aus ingenieurwissenschaftlicher Sicht ist die Evolution nichts anderes als ein Optimierungsverfahren. Sie war trotz ihrer Einfachheit (oder gerade deshalb) in der Lage über Jahrmillionen hinweg unsere heutige Umwelt zu schaffen. Augenscheinlich ist die Evolution als Optimierungprozeß ziemlich leistungsfähig. Es stellt sich daher die Frage, ob es möglich ist, die Erkenntnisse der Evolutionsbiologie auf ingenieurtechnische Problemstellungen zu übertragen und für praxisrelevante Optimierungsaufgaben nutzbar zu machen.

2 Prinzip evolutionärer Algorithmen

Um das Prinzip der Evolution ingenieurtechnisch nutzen zu können, ist es notwendig einige Analogien zwischen Biologie und technischer Optimierung zu suchen. Tabelle 1 zeigt einige wichtige Analogien.

Ein Individuum in diesem Schema stellt eine potentielle Lösung zu einem Optimierungsproblem dar. Aus diesem Grund muß es Informationen über diese Lösung beinhalten, die sogenannte genetische Repräsentation (ähnlich dem Chromosomensatz von Lebewesen, der den 'Bauplan' für Nachkommen enthält). Traditionell kommen an dieser Stelle binäre Vektoren ([Hol75] und [Gol89]) oder reelle Zahlen ([Rec73]) zum Einsatz. Diese Ansätze werden für eine Anzahl von Problemstellungen auch heute noch verwendet, sind allerdings recht unflexibel zu handhaben.

Individuum	potentielle Lösung eines Optimierungsproblems
Population	momentan betrachteter Ausschnitt aus dem Optimierungssuchraum
Chromosomensatz	genetische Repräsentation eines Individuums
Fitneß	Güte einer potentiellen Lösung
Rekombination	gerichtete Suche im Suchraum, Vererbung von Eigenschaften der Eltern
Mutation	ungerichtete Suche im Suchraum

Tabelle 1: Analogien zwischen Evolution und Optimierungsprozessen

Eine Anzahl von Individuen zusammengenommen wird als Population bezeichnet und stellt einen Ausschnitt aus der Gesamtheit der möglichen Lösungen dar. Mit anderen Worten: eine Population ist Stück des Suchraumes, den ein Optimierungsprozeß zu durchlaufen hat. Um Individuen qualitativ einstufen zu können muß eine Gütebewertung stattfinden. Zur Festlegung der Güte (auch Zielfunktionswert) wird eine problemspezifische Zielfunktion herangezogen. In der Praxis wird zum Teil noch einmal zwischen der Güte und der Fitneß eines Individuums unterschieden und eine sogenannte Fitneßskalierung vorgenommen. Darauf soll an dieser Stelle nicht weiter eingegangen werden, nähere Information zu dieser Thematik finden sich in [Poh00]. Verschiedene Ansätze für die Skalierung werden in der Fachliteratur diskutiert, so u.a. in [Bak87], [Gol89] oder [MSV93]. Im nachfolgenden Abschnitten dieses Artikels wird auschließlich der Begriff der Güte verwendet.

Mit Hilfe dieser Analogien läßt sich der Zyklus der Evolution in ein Schema für Algorithmen überführen, welches in Abbildung 1 dargestellt ist.

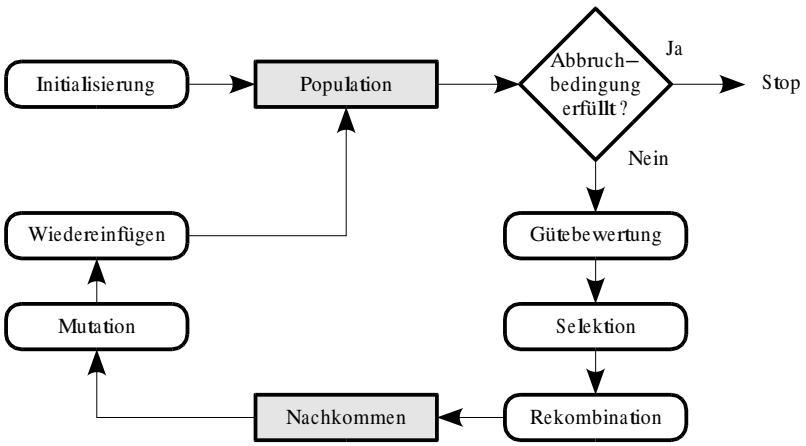

Abbildung 1: Schema eines evolutionären Algorithmus

Eine Population muß zunächst mit einer Anzahl von Individuen initialisiert werden. Diese Individuen werden in der Regel zufällig generiert, allerdings ist es an dieser Stelle auch möglich problemspezifisches Wissen einfließen zu lassen. Nachfolgend findet die Gütebewertung der Individuen mit Hilfe der Zielfunktion statt. Der eigentliche Optimierungszyklus beginnt mit der Selektion von Individuen. Dafür existieren verschiedenartige Ansätze, so z.B. in [Bak87] und [GD91]. Einerseits ist es möglich die Güte der Individuen als Bewertungskriterium heranzuziehen, andererseits aber auch deren Rang innerhalb der Population. Aus den selektierten Individuen werden mit Hilfe eines geeigneten Rekombinationsoperators Nachkommen erzeugt. Analog des biologischen Hintergrunds werden dabei Eigenschaften der Eltern an Nachkommen weitergegeben. Im Kontext einer Optimierung entspricht dieser Schritt einer gerichteten Suche im Suchraum, bei der aus zwei (oder mehreren) guten Lösungen eine bessere erzeugt wird bzw. werden soll. Anschließend werden die neu erzeugten Individuen mit einer gewissen Wahrscheinlichkeit verändert (mutiert). Auf diesem Weg wird eine Aufweitung des Ausschnitts aus dem Suchraum erreicht und vorzeitige Konvergenz des Algorithmus in einem lokalen Extremwert vermieden. Die Mutation stellt eine ungerichtete Suche dar. Die Wahrscheinlichkeit für eine Mutation der

erzeugten Individuen darf dabei nicht zu hoch sein[1], um Zwischenergebnisse, die durch den gerichteten Suchschritt erzielt wurden, nicht zu zerstören. Als letzter Schritt werden alle oder ein Teil der neuen Individuen in die bestehende Population übernommen, wobei ein Teil der Individuen aus der letzten Generation gelöscht wird. Auch dafür existieren verschiedene Ansätze ([Poh00]). Der gesamte Zyklus wird so lange durchlaufen bis eine vorher festgelegte Abbruchbedingung erfüllt ist. Beispiele für solche Kriterien sind zum Beispiel das Unterschreiten eines vorher festgelegten Zielfunktionswertes, das Erreichen einer bestimmten Standardabweichung aller Zielfunktionswerte in der aktuellen Generation oder einfach das Erreichen einer vorgegebenen Zahl von Generationen[2].

3 Anwendung bei Plazierung von Standardzellen

Um das im vorangegangenen Abschnitt erläuterte Prinzip zu veranschaulichen, wird in diesem Abschnitt auf die Anwendung von evolutionären Algorithmen am Beispiel der Plazierung von Standardzellen eingegangen. Um das Prinzip zu demonstrieren und die Anschaulichkeit zu bewaren werden einige Vereinfachungen vorgenommen.

Allgemein müssen drei grundlegende Voraussetzungen erfüllt sein, um eine gegebene Problemstellung auf das besprochene Schema abbilden zu können:

- geeignete Repräsentation einer potentiellen Problemlösung
- geeignete Gütebewertung
- geeignete(r) Rekombinationsoperator(en)

Ein weiterer Gesichtspunkt ist ein geeigneter Mutationsoperator, der sich allerdings meist recht einfach aus der Art des verwendeten Rekombinationsoperators herleiten läßt.

Die Problemdefinition beim Placement von Standardzellen lautet wie folgt: Es existiert eine Anzahl von Modulen mit festgelegten Ein- und Ausgängen und einer festgelegten Verdrahtungsstruktur. Ziel des Placements ist es, eine Anordnung der Zellen zu finden, bei der die Größe des Layouts sowie die Gesamtlänge der Verdrahtungsstruktur möglichst minimal wird. Die Eingangsgrößen sind i.d.R. eine Liste mit Beschreibungen der einzelnen Module aus denen deren Größe und die Lage der Terminals entnehmbar ist, sowie eine Netzliste, die die Verbindungen zwischen den Modulen beschreibt. Ausgangsgröße ist entsprechend eine Liste der X- und Y-Koordinaten der Zellen, u.U. auch deren Orientierung im Layout.

Für unser Beispiel lassen sich folgende Randbedingungen anführen, die zwingend eingehalten werden müssen:

- Zellen dürfen sich nicht überlappen
- Zellen müssen innerhalb der Chipgrenzen liegen
- Zellen müssen in Reihen und Spalten angeordnet werden

Zusätzlich könnten noch andere Bedingungen formuliert werden, wie z.B. daß das Netz mit dem kritischen Pfad möglichst kurz sein muß, daß Crosstalk minimiert werden soll oder daß mögliche thermische Randbedingungen eingehalten werden müssen. Diese Punkte sollen hier allerdings unberücksichtigt bleiben.

Für unser Beispiel nehmen wir an, daß zehn Zellen unterschiedlicher Größe plaziert werden sollen. Alle Zellen werden mit Buchstaben von A bis J markiert. Wenn man vereinfachend annimmt, daß im Layout eine Fläche mit einer vorgegebenen maximalen Breite zur Verfügung steht, läßt sich die Repräsentation einer potentiellen Lösung als Vektor schreiben (siehe dazu Bild 2).

[1] In der Regel werden hier Werte zwischen 0.01 und 0.05 verwendet.

[2] In der Praxis werden meist mehrere Abbruchkriterien verwendet und durch ODER miteinander verknüpft, z.B. das Unterschreiten eines vorgegebenen Wertes und das Erreichen einer maximalen Laufzeit des Algorithmus.

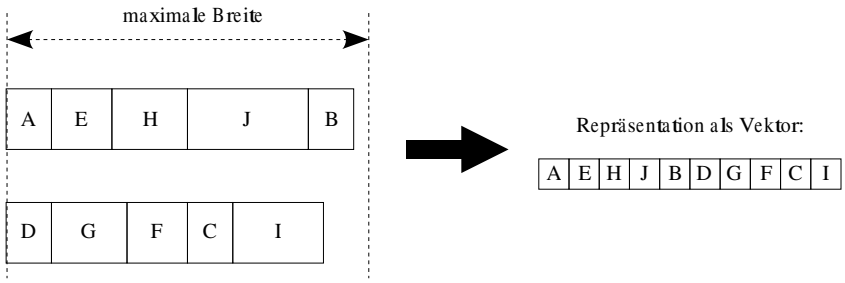

Abbildung 2: genetische Repräsentation

Der nächste Schritt ist die Festlegung der Gütefunktion. Der Einfachheit halber soll dabei nur die Summe der Längen aller Netze betrachtet werden, da die überdeckte Chipfläche sich nahezu von selbst aus der Anordnung der Zellen in Reihen und Spalten ergibt. Bei Routing nach dem Manhattan-Schema läßt sich für die Berechnung der Gesamtnetzlänge eine Approximation nach der Umfangsmethode (*Semiperimeter Method*) verwenden. Dabei wird die Länge eines Netzes als der halbe Wert des Umfangs des Rechtecks, das alle angeschlossenen Terminals umfaßt, angenommen. Die Güte G einer potentiellen Lösung läßt sich daher wie folgt definieren:

$$G = \sum_{\text{Netz } i=1}^{n} l_i \tag{1}$$

mit l_i = Länge des Netzes i

Die fehlende dritte Voraussetzung ist ein geeigneter Rekombinationsoperator. Grundlage für einen solchen Operator ist, daß er in der Lage sein muß, aus zwei gültigen Eltern wiederum ein oder mehrere gültige Nachkommen zu erzeugen. Für unser Beispiel mit Vektoren existieren bereits mehrere verschiedene Verfahren, wie z.B. *Edge Recombination* ([WSF89]), *Order Crossover* ([SMM+91]), *Position Crossover* ([Sys91]) oder *Edge Assembly Crossover* ([NK97]). Position Crossover[3] soll nachfolgend erläutert werden. Der Operator besteht aus drei Schritten, die in Abbildung 3 dargestellt sind:

1. Auswahl einer Anzahl von zufälligen Positionen in ersten Elternteil

2. Kopieren der Elemente von diesen Positionen in Nachkomme unter Beibehaltung der Positionen

3. Kopieren der verbleibenden Elemente aus dem zweiten Elternteil in Nachkomme unter Beibehaltung ihrer Reihenfolge im Vektor

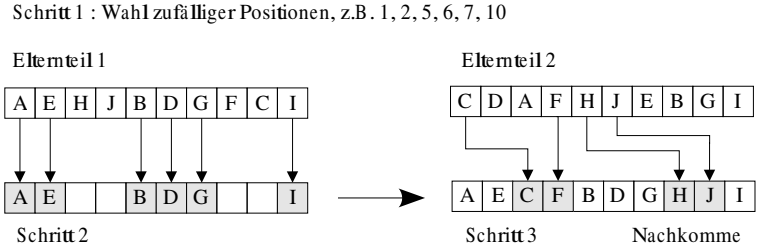

Abbildung 3: Prinzip des Position Crossover

Durch *Position Crossover* wird die relative Ordnung des Vektors gewahrt, so daß im Layout eine Anzahl von Zellen ihre Position behält. Analog lassen sich an dieser Stelle natürlich auch andere Rekombinationsoperatoren für Vektoren verwenden.

[3] In der Literatur wird häufig auch das Kürzel PX verwendet.

Prinzipiell läßt sich mit den besprochenen Voraussetzungen bereits ein evolutionärer Algorithmus aufbauen, der Vollständigkeit halber sei an dieser Stelle allerdings noch ein einfacher Mutationsoperator erwähnt, der sich in dieses Schema einbauen läßt. So ist es beispielsweise möglich zwei oder mehrere zufällig gewählte Elemente im Vektor zyklisch zu vertauschen. Auch eine komplette Umkehrung der Reihenfolge der Elemente ist denkbar. Der Vektor wird dabei verändert, behält aber in jedem Fall seine Gültigkeit.

Das hier beschriebene, recht einfache Verfahren wird sich in dieser Form in der Praxis nur bedingt einsetzen lassen, da aus Gründen der Anschaulichkeit einige Vereinfachungen getroffen wurden. Das dargelegte Schema läßt sich allerdings auf einfache Weise erweitern. Falls beispielsweise zusätzliche Randbedingungen eingehalten werden sollen, muß lediglich die Gütefunktion erweitert werden. Auch eine andere genetische Repräsentation ist denkbar (z.B. durch Vektoren mit den Koordinaten der Zellen). Für eine ausführlichere Beschreibung sei an dieser Stelle auf [MR99] verwiesen, wo sich außerdem eine Variante zur Plazierung von Makrozellen finden läßt.

4 Eigenschaften evolutionärer Algorithmen

Evolutionäre Algorithmen gehören ebenso wie Simulated Annealing, das häufig im EDA-Bereich eingesetzt wird, der Kategorie stochastischer Suchprozesse an. Trotzdem existieren einige grundlegende Unterschiede. Zunächst arbeiten evolutionäre Algorithmen immer mit mehreren potentiellen Lösungen gleichzeitig. Die Suche wird parallel von vielen Punkten im Suchraum aus durchgeführt, wobei der Suchraum besser als bei konventionellen Verfahren abgedeckt wird. Daraus resultiert auch die Tatsache, daß nach Abbruch bzw. Konvergenz des Algorithmus in der Regel mehrere verschiedene Lösungen zur Verfügung stehen, aus denen ein Anwender nach unterschiedlichen Gesichtspunkten die für ihn passende heraussuchen kann. Das kann sich insbesondere für Problemstellungen als nützlich erweisen, von denen bekannt ist, daß nicht nur eine einzige Lösung existiert, sondern mehrere, wie es ja z.B. bei Placement/Floorplanning oder Routing der Fall ist. Weiterhin ist in diesem Zusammenhang anzuführen, daß evolutionäre Algorithmen durch die Kombination von gerichteter und ungerichteter Suche von sich aus in der Lage sind, lokale Optima zu umgehen.

Der Mechanismus evolutionärer Algorithmen läßt sich für viele unterschiedliche Problemklassen anwenden. Diese können sowohl kontinuierlicher (z.B. Extremwertsuche bei mathematischen Funktionen) als auch diskontinuierlicher Art sein (z.B. Scheduling). Da die Zielfunktion frei wählbar ist, funktioniert der dargelegte Mechanismus in beiden Funktionsräumen gleichermaßen gut.

Bei praxisrelevanten Optimierungsproblemen, die auf das Evolutionsschema abgebildet wurden, hat sich erwiesen, daß 90–95% der Rechenzeit des gesamten Algorithmus für Berechnung der Zielfunktion, d.h. auf die Gütebewertung der Individuen aufgewendet werden. Da diese Berechnungen voneinander vollkommen unabhängig sind, läßt sich ein solcher Algorithmus sehr gut parallelisieren und auf verschiedene Rechner in einem lokalen Netzwerk verteilen.

An dieser Stelle sollen allerdings auch einige Nachteile evolutionärer Algorithmen nicht verschwiegen werden. Zunächst sollten diese Algorithmen nicht bei Problemen eingesetzt werden, bei denen ein spezieller Lösungsweg bekannt ist. Da in einem speziellen Lösungsverfahren in der Regel eine Menge an problemspezifischem Wissen steckt, muß ein stochastisches Suchverfahren von Natur aus schlechtere Ergebnisse produzieren bzw. eine höhere Laufzeit aufweisen. Zwar läßt sich auch in verschiedene Operatoren eines evolutionären Algorithmus Expertenwissen einbauen (z.B. in Initialisierung oder Rekombination) und damit die Startbedingungen verbessern oder sogar die Suche steuern. Allerdings sind solche Steuerprozesse aufgrund der hohen Dynamik innerhalb des Algorithmus sehr schwierig in ihrer Anwendung. Weiterhin ist es aufgrund des stochastischen Charakters mit hoher Wahrscheinlichkeit unmöglich, eine bestimmte Lösung zweimal zu erzeugen (daher die Ausrichtung auf Probleme bei denen *eine* gute Lösung gesucht ist und nicht *die* beste).

Wie bereits angesprochen, muß die Zielfunktion sehr häufig berechnet werden. Daher verbietet sich der Einsatz bei Problemstellungen, bei denen die Berechnung der Zielfunktion sehr aufwendig ist. Allerdings ist Grenze dafür durch die Erhöhung der Kapazität zukünftiger Rech-

nergenerationen fließend. Auch läßt sich dem durch massive Parallelisierung entgegenwirken.

5 Zusammenfassung und Ausblick

Evolutionäre Algorithmen sind in der Regel recht einfach und flexibel anzuwenden und haben sich als ziemlich robust erwiesen. Im Vergleich zu den Vorgängen in der Natur sind solche Algorithmen allerdings noch stark vereinfacht. Es wird daher Gegenstand zukünftiger Arbeiten sein, ob eine genauere Simulation der Evolution (z.B. Zusammenhang Genotyp und Phänotyp oder Reaktion von Individuen auf wechselnde Umweltbedingungen) zu wesentlich besseren oder schnelleren Algorithmen führt. Vielversprechend sind auch Ansätze, die mit mehreren Population arbeiten, die sich parallel auf unterschiedliche Weise weiterentwickeln und darüberhinaus in der Lage sind Individuen untereinander auszutauschen.

Fortschritte auf dem Gebiet der evolutionären Algorithmen werden sich mit großer Sicherheit auch im Bereich EDA niederschlagen. Schon heute werden solche Algorithmen nicht nur bei der Plazierung von Standard- und Makrozellen oder beim Technology Mapping von programmierbaren Bausteinen angewendet, sondern auch bei der Generation von Testpattern für die Simulation von digitalen Schaltkreisen oder sogar bei der Suche nach dem kritischen Pfad in Schaltungen. In Zukunft werden sich weitere Anwendungsfelder eröffnen und bestehende Verfahren weiter verbessert werden, um qualitativ bessere Lösungen zu ermöglichen.

Literatur

[Bak87] J. E. Baker. Reducing Bias and Inefficiency in the Selection Algorithm. In *Proceedings of the Second International Conference on Genetic Algoritms and their Application*, pages 14–21, 1987.

[Dar59] Charles Darwin. *On the Origin of Species by Means of Natural Selection.* John Murray, London, 1859.

[GD91] D. E. Goldberg and K. Deb. A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. In *Foundations of Genetic Algorithms*, pages 69–93. Morgan Kaufmann Publishers, San Mateo, 1991.

[Gol89] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison Wesley, Reading, Massachussetts, 1989.

[Hol75] John Henry Holland. *Adaption in Natural and Artificial Systems.* MIT Press, Cambridge, Massachussetts, 1975.

[MR99] Pinaki Mazumder and Elizabeth M. Rudnick. *Genetic Algorithms for VLSI Design, Layout & Test Automation.* Prentice Hall PTR, 1999.

[MSV93] H. Mühlenbein and D. Schlierkamp-Voosen. Predictive Models for the Breeder Genetic Algorithm: I. Continuous Parameter Optimization. In *Evolutionary Computation*, pages 25–49. GMD - Forschungszentrum Informationstechnik, 1993.

[NK97] Y. Nagata and S. Kobayashi. Edge Assembly Crossover: A High-Power Genetic Algorithm for the Traveling Salesman Problem. In *Proceedings of the Seventh International Conference on Genetic Algorithms and their Application*, pages 450–457. Van Nostrand Reinold, 1997.

[Poh00] Hartmut Pohlheim. *Evolutionäre Algorithmen.* Springer-Verlag Berlin, Heidelberg, New York, 2000.

[Rec73] Ingo Rechenberg. *Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Fromman-Holzboog, Stuttgart, 1973.

[SMM+91] T. Starkweather, S. McDaniel, K. Mathias, D. Whitley, and C. Whitley. A Comparison of Genetic Sequencing Operators. In *Proceedings of the Fourth International Conference on Genetic Algorithms and their Application*, pages 69–76, 1991.

[Sys91] G. Syswerda. Schedule Optimization Using Genetic Algorithms. In *Handbook of Genetic Algorithms*, pages 133–140. Van Nostrand Reinold, 1991.

[WSF89] D. Whitley, T. Starkweather, and D. Fuquay. Scheduling Problems and the Travelling Salesman: The Genetic Edge Recombination Operator. In *Proceedings of the Third International Conference on Genetic Algorithms and their Application*, pages 133–140, 1989.