

Diskrete Optimierung mit der Evolutionsstrategie auf parallelen Systemen

Martin Bernreuther

Kurzfassung: Der Beitrag erläutert zunächst die Grundlagen des parallelen Rechnens auf einem Workstationcluster. Dabei kommt das Softwarepaket PVM (parallel virtual machine) zum Einsatz. Nachfolgend wird ein Überblick über die Evolutionsstrategien gegeben und im besonderen auf die Parallelisierbarkeit dieser Methode eingegangen. Den Abschluß bildet ein konkretes Beispiel aus dem Bereich der diskreten Strukturoptimierung.

1 Paralleles Rechnen auf verteilten Systemen

1.1 Rechnerklassifikation

Workstationcluster fallen in die Gruppe der MIMD- (Multiple Instruction, Multiple Data) Rechner. Hierbei führt jedes PE (Processing Element) eigenständig unterschiedliche Programminstruktionen aus. Beim Cluster sind die einzelnen Rechner über ein Verbindungsnetzwerk mit Nachrichtenaustausch lose gekoppelt und bilden ein verteiltes Rechnersystem mit asynchroner Parallelität (siehe Abb. 1).

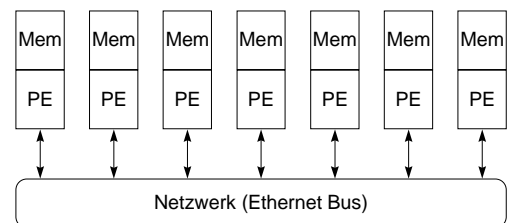


Abb. 1: Schematische Darstellung des Workstationclusters

1.2 Verbindungsstrukturen

Beim Workstationcluster wird als Verbindungsstruktur ein Bus-Netzwerk verwendet. Da nur eine Verbindung pro PE nötig ist, sind die „Produktionskosten“ optimal. Der Abstand zwischen zwei beliebigen PEs ist immer konstant gleich 2: $PE_i \rightarrow \text{Bus} \rightarrow PE_j$.

Zu einem bestimmten Zeitpunkt kann bei einem Bus nur eine Verbindung aufgebaut werden. Das Senden von Daten an mehrere Empfänger zugleich (broadcast) ist möglich, aber ein Empfänger kann nicht von mehreren Sendern zugleich empfangen. Auch ist der Austausch von Daten zwischen verschiedenen PE-Paaren nicht parallel realisierbar. Infolge der konstanten Bandbreite des Busses kann diese Struktur nicht skaliert werden.

Aufgrund der geringen Kommunikationsleistung des Cluster-Verbindungsnetzwerkes bezüglich der Übertragungsgeschwindigkeit (Ethernet: 10÷15 MBit/sec) und der hohen Latenzzeit ist das auf der Prozedurebene parallele Message-Passing-Konzept das einzig durchführbare Programmiermodell auf diesem Rechnertyp.

1.3 Netzwerkprotokoll

Auf die Verbindungshardware des Ethernetbusses wird das UDP-TCP/IP-Protokoll (User Datagram - Transmission Control Protocol / Internet Protocol) aufgesetzt. Auf der Transportschicht unterscheiden sich die verbindungsorientierte TCP-Kommunikation und die verbindungslose UDP-Kommunikation. Die drei Stufen einer TCP-Verbindung bestehen aus dem Verbindungsaufbau, der eigentlichen Datenübertragung und dem Abbau der Verbindung. Das ist vergleichbar mit einem Telefon. Hierbei ist die Anzahl der Sockets bzw. Anschlüsse begrenzt. - Im Gegensatz dazu werden beim UDP einzelne Datagramme verschickt: Die Daten werden in einzelne Pakete zerteilt, verschickt und beim Empfänger wieder zusammengesetzt. Die Reihenfolge und der Weg dieser Pakete sind unbestimmt. Das Analogon hierfür stellen Briefe dar. Der Verwaltungsaufwand ist gering, und pro Rechner ist nur ein Socket notwendig. Das UDP ist im Gegensatz zu TCP ein „unsicheres“ Protokoll, d. h. Pakete können verloren gehen oder doppelt ankommen.

1.4 PVM

PVM (Parallel Virtual Machine) ist ein frei verfügbares Softwaresystem, das es ermöglicht, mehrere vernetzte Computer als einen (virtuellen) Parallelrechner zu verwenden. Die einzelnen Rechner können von verschiedener Bauart sein. Das System, das auf einem Message-Passing-Konzept aufbaut, stellt in einer Bibliothek Funktionen zur Verfügung, die direkt in C- und Fortran-Programme einbindbar sind. PVM stellt also keine parallele MIMD-Programmiersprache dar.

1.4.1 Funktionsweise

Auf jedem Host der PVM-Konfiguration wird die Kommunikation von einem PVM-Dämon gesteuert. Ein Dämon fungiert hierbei als „Master“. Dieser Master kann über den remote-shell-Dienst weitere Dämonen und Tasks starten oder stoppen. Damit ist eine dynamische Anpassung der Konfiguration während der Laufzeit möglich. Die Kommunikation zwischen dem Dämon und einem Task auf dem jeweiligen Computer wird mit einer TCP-Verbindung bewerkstelligt. Die Dämonen untereinander verständigen sich auf Basis des UDP-Dienstes (vgl. 1.3) (siehe Abb. 2). Zum Datenaustausch müssen die beteiligten Prozesse synchronisiert werden. Beim

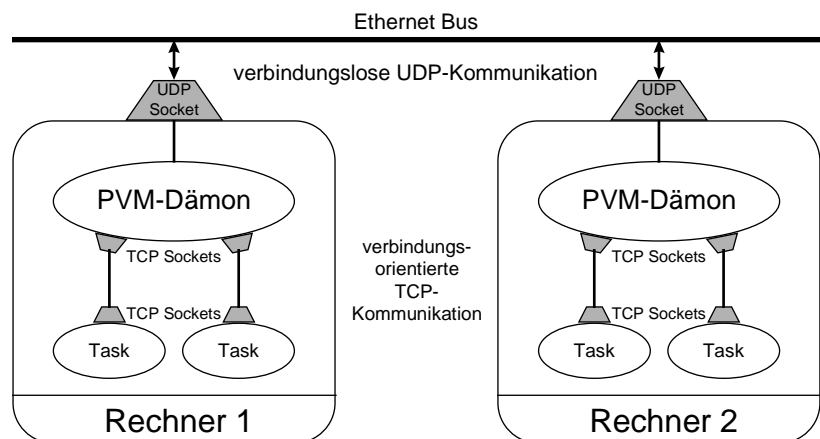


Abb. 2: PVM Kommunikation

blockierenden Empfangen wird vom Empfänger solange auf die Nachricht gewartet, bis sie eingetroffen ist. Nichtblockierendes Empfangen bedeutet hingegen, daß über den Rückgabewert der Funktion angezeigt wird, ob die Nachricht bereits vorliegt. Im Fall einer noch nicht vorliegenden Nachricht kann das Programm anderweitige Aufgaben bearbeiten. Das blockierende Empfangen mit Zeitbeschränkung stellt ein Bindeglied dar. Sollen Daten zwischen Architekturen mit unterschiedlicher interner Darstellung der Daten ausgetauscht werden, so müssen diese in das XDR-Format (eXternal Data Representation) transformiert und beim Empfänger rücktransformiert werden. Aufgrund des im Verhältnis zur Rechenleistung des Computers langsamen Netzwerkes ist es ratsam, möglichst selten zu kommunizieren, und dafür die einzelnen Pakete ausreichend groß zu halten.

1.4.2 Lastausgleich

Der Lastausgleich (Load Balancing) wird von PVM nur sehr unzureichend verwirklicht. Vom Anwendungsprogramm kann ein statischer Lastausgleich erzielt werden, indem die Aufgaben der Leistungsfähigkeit entsprechend auf die einzelnen Rechner verteilt werden. Beim dynamischen Lastausgleich wird die Aufgabenverteilung während der Laufzeit bestimmt, und es wird auf die wechselnden Belastungen der Rechner Rücksicht genommen. Bei der verbreitetsten Methode des Master/Slave Konzepts, dem „Pool of Tasks“, werden im Masterprogramm die freien Slaves in einer Warteschlange verwaltet. Ein effizienter Lastausgleich stellt bei einem Workstationcluster eine nur schwer durchführbare Aufgabe dar, da auf einem Multiusersystem die Last ständig sehr stark und unkontrollierbar schwankt.

2 Diskrete Optimierung mit Evolutionsstrategien

2.1 Evolutionsstrategien

Inspiziert von dem Vorbild der biologischen Evolution wird versucht diese natürliche Optimierung auf technisch-mathematische Ingenieur Anwendungen zu übertragen. Wird von einer Systemparameter-Optimierung ausgegangen, repräsentiert der Vektor eines Parametersatzes ein einzelnes Individuum. Die Vitalität eines solchen Individuums wird durch die Fitneß-Funktion beschrieben.

2.1.1 (1+1)-ES

Die einfachste Variante dieser Strategie ist die von Rechenberg [3] vorgestellte zweigliedrige (1+1)-ES, die sich auf ein Mutations-Selektions-Prinzip gründet. Ein Individuum wird dupliziert und durch Mutation verändert. Durch Auswertung der Fitneß-Funktion vergleicht man das veränderte Kind- mit dem ursprünglichen Elter-Individuum. Das bessere überlebt und wird zum Elter der nächsten Generation. Die Mutation kann durch Addition eines normalverteilten Vektors erzielt werden. Der Erwartungswert der einzelnen Komponenten

dieses Vektors ist 0. Dieses einfache Verfahren kann als eine Art „stochastisches Gradientenverfahren“ angesehen werden. Für die Erzeugung im Intervall (0,1) gleichverteilter Zufallszahlen Z aus gleichverteilten Zufallszahlen Y wird von Schwefel [4] auf eine Transformationsregel von Box und Muller verwiesen:

$$Z'_1 = \sqrt{-2 \cdot \ln Y_1} \cdot \sin(2 \cdot \pi \cdot Y_2) \quad Z'_2 = \sqrt{-2 \cdot \ln Y_1} \cdot \cos(2 \cdot \pi \cdot Y_2) \quad (1)$$

Die Transformation auf $N(0, \sigma)$ erfolgt mit: $Z_i = \sigma_i \cdot Z'_i$ (2)

2.1.2 ($\mu+\lambda$) ES

Bei den mehrgliedrigen ES wird von μ Elter- und λ Kind-Individuen ausgegangen. Zur Erzeugung der Kinder wird vor dem Mutations- (*mut*) der Rekombinationsoperator (*rec*) verwendet. Grundsätzlich wird beim Selektionsoperator (*sel*) zwischen Plus- und Komma-Strategien unterschieden. Bei den Plusstrategien konkurrieren die Eltern mit den Kindern, bei der Kommastrategie hingegen sterben die Eltern sofort. Dies ist für das Überwinden lokaler Minima, sowie die Selbstadaption der Strategieparameter vorteilhaft. Im Gegensatz zur Plusstrategie ist bei der Kommastrategie auch eine Verschlechterung von einer zur nächsten Generation möglich. Im vorliegenden Beitrag wurde ein leicht modifizierter Weg eingeschlagen. Die Eltern werden nach jeder Generation durch eine Strafterm verschlechtert. Hierbei wurde die Addition eines konstanten Wertes auf die zu minimierende Fitness-Funktion gewählt. Dies simuliert den Alterungsprozeß der Eltern. Für den Wert 0 erhält man eine Plus-, für einen sehr großen Wert praktisch eine Kommastrategie. Das Vorgehen läßt sich mit folgender Formel beschreiben:

$$opt_{(\mu\#\lambda)-ES}(P^{(t)}) = sel_{\mu}^{\mu+\lambda} \left(\bigcup_{i=1}^{\lambda} \{ mut(rec(P^{(t)})) \} \cup P_{pen}^{(t)} \right) \quad (3)$$

Sämtliche Restriktionen des Problems wurden ebenfalls als Strafterme miteinbezogen. Konstante Terme sind hier unbrauchbar, da man keinen Gradienten erzeugt, der das Problem in den zulässigen Bereich zurückführt. Die Wahl fiel daher auf lineare Strafterme.

Für $v > v_{zul}$ wird eine festgelegte Konstante mit dem Faktor $\frac{v}{v_{zul}}$ multipliziert und zur Fitness-Funktion addiert.

Bei den Strategieparametern wurde ein alternatives Konzept getestet. Anstatt jedem Individuum die Standardabweichungen mitzugeben, bestimmt man nach jeder Generation die Standardabweichungen der Merkmale der Eltern, und falls eine untere Schranke nicht unterschritten wurde, verwendet man diese bei der normalverteilten Mutation der erzeugten Kinder. Für die Plusstrategie wurde für das vorgestellte Beispiel eine zufriedenstellende Konvergenz erreicht.

2.2 Parallelisierung

Die biologische Evolution findet massiv parallel statt. Auch die mehrgliedrige Evolutionsstrategie ist gut geeignet, parallel implementiert zu werden. Dazu gibt es mehrere Ansatzpunkte. Pro Generation wird die Fitneß-Funktion für alle Kinder ausgewertet. Diese Berechnungen sind völlig unabhängig voneinander. Da sich nur die Parameter ändern, ist der Kommunikationsaufwand relativ gering.

Weitere Ansätze einer parallelen Evolutionsstrategie sind mit Multipopulationen möglich. Beim Insel-Modell entwickeln sich auf den einzelnen Rechnern Populationen unabhängig voneinander. Werden Individuen zwischen diesen Inseln ausgetauscht, kann dies zu einer Netzwerk-Methode erweitert werden.

Die hier vorgestellte Implementierung einer ES geht von einem Master/Slave Konzept aus. Die Slave-Prozesse werden mit den konstanten Daten initialisiert und werten dann die Fitneß-Funktion der Individuen einer Generation parallel aus. Dabei wird die Grobkörnigkeit der Jobs maximal, wenn jedem Slave pro Generation nur ein Job vergeben wird. Auf diese Weise kann nur ein statischer Lastausgleich erzielt werden. Die Aufteilung in kleinere Einheiten erhöht den Kommunikationsaufwand, macht aber einen dynamischen Lastausgleich möglich. Hier muß in Abhängigkeit von der Netzwerk-Bandbreite ein Kompromiß gefunden werden.

2.3 Anwendung

Nachfolgend steht die diskrete Optimierung eines Fachwerks mit 10 Stäben im Vordergrund, welche bereits in einer Veröffentlichung von M. Galante [5] vorgestellt wurde.

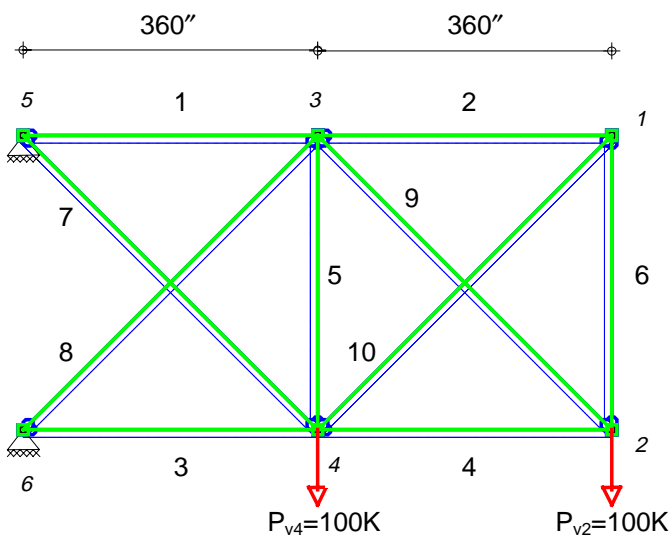


Abb. 3: zu optimierendes Fachwerk

E-Modul [Ksi]: 10^4
 Max. Spannung [Ksi]: 25
 Spez. Gewicht [lb/in³]: 0,1 (Aluminium)

Tabelle 1: Profil-Querschnitte [si]

1,62	1,8	1,99	2,13	2,38	2,62
2,63	2,88	2,93	3,09	3,13	3,38
3,47	3,55	3,63	3,84	3,87	3,88
4,18	4,22	4,49	4,59	4,8	4,97
5,12	5,74	7,22	7,97	11,5	13,5
13,9	14,2	15,5	16,0	16,9	18,8
19,9	22,0	22,9	26,5	30,0	33,5

Max. Durchbiegung [in]: 2

Dieses Problem wurde mit einer (15+100) Strategie und dualer diskreter Crossover-Rekombination angegangen. Die Querschnittswerte aller Fachwerkelemente sind für jedes

Individuum jeweils in einem Vektor $\vec{p} = [q_1 \quad q_{10}]$, $q_i \in \{0,1, .41\}$ codiert. Der Mutation lag eine generationsweise Berechnung der Standardabweichung jedes Merkmals \bar{p}_i aller Eltern zugrunde. Die Qualitätssteigerung des besten Individuums wird in Abb. 4 dargestellt. Das Optimum stellte sich mit einem Gewicht von 5490,7 lb nach 86 Generationen mit den Querschnitten (1-10)

33,5 1,62 22,9 14,2 1,62 1,62 7,97 22,9 22,0 1,62

ein. Dies entspricht dem Ergebnis, das in [5] unter Verwendung eines genetischen Algorithmusses errechnet wurde.

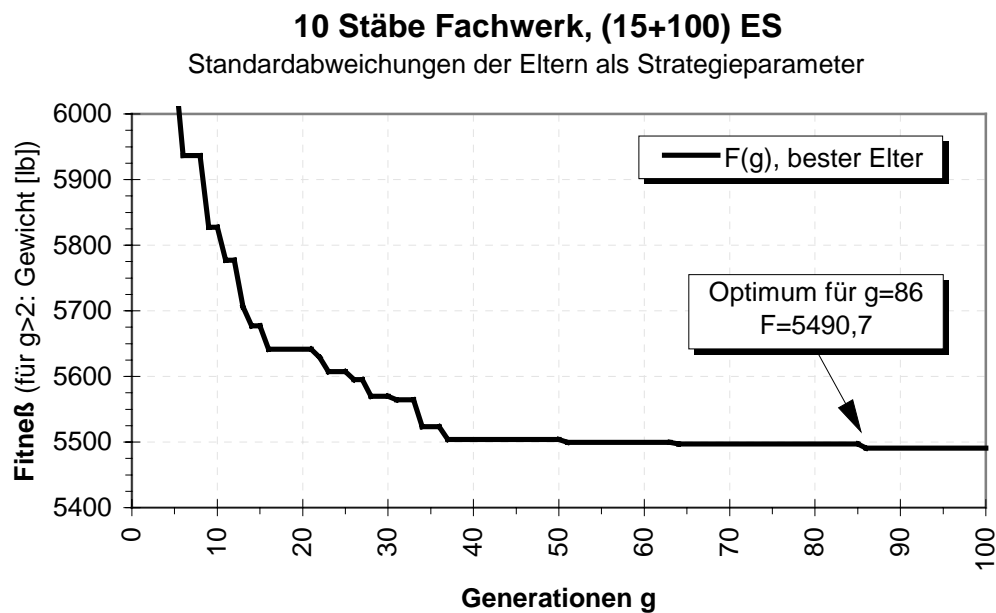


Abb. 4: Fitneß des besten Individuums der jeweiligen Generation

Literatur

- [1] Bräunl, Thomas: Parallele Programmierung. Braunschweig, Wiesbaden: Vieweg, 1993
- [2] Geist, A.; Beguelin, A.; Dongarra, J.; Jiang, W.; Manchek, R.; Sunderam, V.: PVM Parallel Virtual Machine - A User's Guide and Tutorial for Networked Parallel Computing. Cambridge: MIT Press, 1994
- [3] Rechenberg, Ingo: Evolutionsstrategie. Stuttgart: Frommann-Holzboog, 1973
- [4] Schwefel, Hans-Paul: Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie. Basel, Stuttgart: Birkhäuser, 1977
- [5] Galante, M.: Genetic Algorithms as an Approach to Optimize Real-world Trusses, Int. j. numer. methods eng., vol. 39, 361-382 (1996)