Reichelt, Dirk ; Rothlauf, Franz :

CURE: Eine Reparaturheuristik für die Planung ökonomischer und zuverlässiger Kommunikationsnetzwerke mit Hilfe von heuristischen Otimierungsverfahren

Zuerst erschienen in:
Kommunikation in verteilten Systemen (KiVS). - Berlin [u.a.] : Springer, 2005, S. 283-294

CURE: Eine Reparaturheuristik für die Planung ökonomischer und zuverlässiger Kommunikationsnetzwerke mit Hilfe von heuristischen Optimierungsverfahren

Dirk Reichelt[1] und Franz Rothlauf[2]

[1] Institut für Wirtschaftsinformatik, Technische Universität Ilmenau
Dirk.Reichelt@tu-ilmenau.de
[2] Lehrstuhl für ABWL und Wirtschaftsinformatik, Universität Mannheim
rothlauf@uni-mannheim.de

Zusammenfassung. Dieser Beitrag beschäftigt sich mit dem Aufbau kostengünstiger Kommunikationsnetzwerke unter Zuverlässigkeitsrestriktionen. Für den Aufbau des Kommunikationsnetzes stehen je Verbindung verschiedene Leitungstypen mit unterschiedlichen Zuverlässigkeiten und Kosten zur Verfügung. Im Rahmen der Planung ist das Netzwerk so aufzubauen, dass das resultierende Gesamtnetz kostenminimal ist und eine geforderte minimale Gesamtzuverlässigkeit garantiert werden kann. Aufgrund der hohen Komplexität des Problems (NP-vollständig) werden üblicherweise heuristische Optimierungsverfahren zur Lösung eingesetzt. Um sicherzustellen, dass die dadurch ermittelten Lösungen die geforderte Zuverlässigkeit aufweisen, werden in den meisten Ansätzen unzulässige Lösungen, welche die geforderte Zuverlässigkeit nicht erfüllen, durch die Verwendung von Straftermen schlechter bewertet. Der vorliegende Beitrag ersetzt diese Strafterme durch eine Reparaturheuristik (CURE). CURE stellt sicher, dass heuristische Optimierungsverfahren nur zulässige Lösungen erzeugen und keine Strafterme für invalide Lösungen mehr notwendig sind. Experimentelle Untersuchungen der Leistungsfähigkeit von heuristischen Optimierungsverfahren am Beispiel eines genetischen Algorithmus zeigen, dass durch CURE im Vergleich zu Ansätzen mit Straftermen deutlich bessere Lösungen mit geringerem Aufwand gefunden werden können.

1 Einleitung

Bei der Planung von Netzwerktopologien zum Aufbau verteilter Systeme müssen in der Regel mehrere (oft auch konfligierende) Kriterien wie Kosten des Netzwerks und Ausfallsicherheit beachtet werden. Für den Planer stellt sich die Aufgabe, eine Netzwerkstruktur zu finden, welche möglichst kostengünstig ist, trotzdem aber eine vorgegebene Gesamtzuverlässigkeit erfüllt. Für den Aufbau eines derartigen Kommunikationsnetzes stehen üblicherweise Leitungstypen mit unterschiedlichen Zuverlässigkeiten (und entsprechenden Kosten) zur Verfügung.

In der Regel nehmen die Kosten einer Leitung mit der zugesicherten Ausfallsicherheit (Zuverlässigkeit) der Verbindung zu. Ein gebräuchliches Maß für die Messung der Gesamtzuverlässigkeit eines Kommunikationsnetzwerkes ist die *All-Terminal Zuverlässigkeit*. Dieses Maß berechnet sich aus den vorgegebenen Zuverlässigkeiten der einzelnen Leitungen, welche im Netzwerk eingesetzt werden, und gibt die Wahrscheinlichkeit dafür an, dass sämtliche Knoten des Netzwerks miteinander kommunizieren können. Das Problem des Findens einer kostenminimalen Netzwerkstruktur bei vorgegebener Gesamtzuverlässigkeit zählt zu den NP-harten Problemen [1, S.207]. Aufgrund der hohen Komplexität des Problems wurden in der Vergangenheit überwiegend heuristische Optimierungsverfahren wie z.B. genetische Algorithmen zur Lösung des Problems eingesetzt [3-7,9]. Der vorliegende Beitrag stellt ein Verfahren zur Ermittlung einer kostenminimalen Netzwerkstruktur bei Berücksichtigung einer vorgegebenen Gesamtzuverlässigkeit des Netzwerkes vor. Als Verfahren zur Ermittlung kostenminimaler Netzwerkstrukturen wird ein genetischer Algorithmus eingesetzt. Der Hauptunterschied zu bisherigen Ansätzen liegt in der Art und Weise, in der die geforderte minimale Gesamtzuverlässigkeit des Netzwerkes berücksichtigt wird. Im Gegensatz zu den meisten bisherigen Arbeiten, bei welchen Netzwerkstrukturen, welche die geforderte Zuverlässigkeit nicht erfüllen, durch die Einführung von Strafttermen (Penalties) schlechter bewertet werden [3,7], wird im vorliegenden Beitrag eine Reparaturheuristik (CURE) eingesetzt. Durch CURE wird sichergestellt, dass sämtliche Lösungen, welche durch das heuristische Optimierungsverfahren generiert werden, zulässig sind und die geforderte Gesamtzuverlässigkeit für das Netzwerk aufweisen. Zusätzliche Strafterme sind damit nicht mehr notwendig. Die Leistungsfähigkeit von CURE wird anhand von drei Benchmarkproblemen aus der Literatur untersucht. Die Ergebnisse zeigen, dass durch die Verwendung von CURE deutlich bessere Lösungen gefunden werden können, als mit herkömmlichen Straftermansätzen.

2 Problembeschreibung

2.1 Ökonomische Netzwerkplanung unter Zuverlässigkeitsrestriktionen

Ziel des Planungsprozesses ist es, eine Netzwerkstruktur mit minimalen Kosten zu finden, welche einer vorgegebenen Zuverlässigkeitsanforderung genügt. Ein Netzwerk wird hierbei als ein ungerichteter Graph $G(K, E)$ modelliert. K ist die Menge der Knoten und E die Menge der Kanten im Graphen. Durch jede Kante bzw. Knoten wird eine Verbindung bzw. ein Knoten des zugehörigen Netzwerkes repräsentiert. Es wird davon ausgegangen, dass die Position der Knoten fest vorgegeben ist und deren Installationskosten für die Planung nicht relevant sind. Für jede Kante $e_{ij} \in E$ zwischen Knoten i und j besteht die Auswahlmöglichkeit zwischen verschiedenen Zuverlässigkeitsoptionen l_k $(k = 1 \dots n)$, welche sich hinsichtlich Kosten $c(l_k(e_{ij}))$ und Ausfallsicherheit $r(l_k(e_{ij}))$ unterscheiden. Dabei entspricht $l_k(e_{ij})$ der aktuell für die Kante e_{ij} gewählten Option k. Weiterhin gelten die Annahmen, dass die Knoten zuverlässig arbeiten, jede Kante sich entweder im Zustand $s_{e_{ij}}$ „operational" ($s_{e_{ij}} = 1$) oder „failed" ($s_{e_{ij}} = 0$) befindet, die Ausfallwahrscheinlichkeiten der einzelnen Leitungen unabhängig voneinander sind, die Verbindungen bidirektional sind und Reparaturen ausgefallener Verbindungen nicht berücksichtigt werden. Die Lösung des Optimierungsproblems wird durch den Subgraph $G_N(K, E_N \subset E)$ dargestellt. Die Zielfunktion für das Optimierungsproblem lautet:

$$\begin{aligned} C(G_N) = \sum_{e_{ij} \in E_N} c(l_k(e_{ij})) \rightarrow min \\ \text{mit: } R(G_N) \geq R_0 \end{aligned} \tag{1}$$

Dabei sind $C(G_N)$ die Gesamtkosten des Netzwerkes, welche sich aus den Kosten $c_k(l(e_{ij}))$ der aktuell gewählten Optionen für die Kanten e_{ij} zusammensetzen. $R(G_N)$ ist die Gesamtzuverlässigkeit des Netzwerks, die eine minimal geforderte Zuverlässigkeit R_0 nicht unterschreiten darf.

Lösungsansätze für dieses Optimierungsproblem wurden bereits in verschiedenen früheren Arbeiten entwickelt. In [8] wird ein Branch and Bound Algorithmus vorgestellt, der die Netzwerkkosten unter Beachtung einer Zuverlässigkeitsschranke minimiert. Smith et al. stellen in [3,7] genetische Algorithmen vor, bei denen die Zuverlässigkeitsnebenbedingung über einen Strafterm direkt in die Zielfunktion integriert wird. Die Autoren arbeiten in [7] lediglich mit einer Option pro Kante. In [3] wird der Ansatz auf Probleme mit unterschiedlichen Optionen pro Kante erweitert. Eine Erweiterung des Ansatzes aus [7] für größere Probleminstanzen erfolgt in [9] durch den Einsatz von parallelen genetischen Algorithmen. Eine parallele Betrachtung von Zuverlässigkeit und Kosten als multikriterielles Optimierungsproblem erfolgt in [5,6].

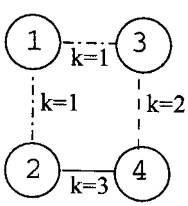
2.2 Zuverlässigkeit von Netzwerktopologien

Die Bewertung der Zuverlässigkeit der Kommunikation zwischen allen vorhandenen Knoten eines Netzwerks erfolgt durch die All-Terminal Zuverlässigkeit R_{All} [2-8]. Diese ist definiert als die Wahrscheinlichkeit, dass zwischen jedem Knotenpaar des Netzwerks ein funktionierender Kommunikationspfad existiert [10]. R_{All} kann als die Wahrscheinlichkeit interpretiert werden, dass in dem Netzwerk mindestens ein funktionierender Baum existiert, welcher sämtliche Knoten des Netzwerks miteinander verbindet [2, S. 3]. Die allgemeine Berechnungsvorschrift der All-Terminal Zuverlässigkeit für ein Netzwerk G_N lautet:

$$R_{All}(G_N) = \sum_{s_i \in S} \Phi(s_i) \cdot Pr(s_i) \tag{2}$$

Ein Zustand s_i repräsentiert dabei einen möglichen Zustand des Netzwerks G_N bei dem ein Teil der Kanten ausgefallen ist. Ein Zustand s_i gilt dabei als „operational" ($\Phi(s_i) = 1$), wenn der resultierende Graph immer noch verbunden ist. Falls der dem Zustand s_i entsprechende Graph nicht mehr verbunden ist, gilt $\Phi(s_i) = 0$. $Pr(s_i)$ entspricht der Eintrittswahrscheinlichkeit der einzelnen Zustände s_i. Ein Beispiel für die Berechnung von R_{All} wird in Abschnitt 2.3 gegeben. Die exakte Berechnung der All-Terminal Zuverlässigkeit zählt zu den NP-harten Problemen [2, S. 3]. Daher wurden für die Berechnung und Approximation von R_{All} in den vergangenen Jahren eine Reihe unterschiedlicher Methoden entwickelt. Ein einfaches Verfahren zur Bestimmung einer oberen Grenze für R_{All} wird in [11] beschrieben. Eine exakte Berechnung von R_{All} kann mit Hilfe von dem in [10] beschriebenen Dekompositionsansatz erfolgen. Bei zunehmender Netzwerkgröße ist allerdings der Einsatz von exakten Berechnungsverfahren auf Grund der langen Laufzeiten und dem damit verbundenen hohen Ressourcenbedarf nicht mehr praktikabel. An Stelle einer exakten Berechnung von R_{All} treten dann zunehmend Schätzverfahren, wie z.B. Monte-Carlo Simulationen [2,4-7]. Das Grundprinzip sämtlicher Monte-Carlo Techniken ist identisch. In mehreren unabhängig voneinander durchgeführten Stichproben generiert das Verfahren jeweils einen Zustand s_i des Netzwerks und untersucht, ob der entsprechende Graph verbunden ist. Aus einer Vielzahl von Stichproben wird anschließend eine Schätzung für die tatsächliche All-Terminal Zuverlässigkeit vorgenommen.

2.3 Ein Beispiel zur Bestimmung der Gesamtzuverlässigkeit und Kosten eines Kommunikationsnetzwerks



	$l_1(e_{ij})$		$l_2(e_{ij})$		$l_3(e_{ij})$	
	$r(l_1(e_{ij}))$	$c(l_1(e_{ij}))$	$r(l_2(e_{ij}))$	$c(l_2(e_{ij}))$	$r(l_3(e_{ij}))$	$c(l_3(e_{ij}))$
e_{12}	0,8	10	0,9	20	0,95	40
e_{24}	0,8	14	0,9	28	0,95	56
e_{34}	0,8	24	0,9	48	0,95	96
e_{13}	0,8	12	0,9	24	0,95	48

Abb. 1. Beispielnetzwerk zur Bestimmung von Netzwerkkosten und -zuverlässigkeit

An Hand des in Abb. 1 gezeigten Beispielnetzwerks wird die Berechnung der Netzwerkkosten sowie der All-Terminal Zuverlässigkeit demonstriert. Das abgebildete Netzwerk besteht aus 4 Knoten und 4 Kanten (e_{12}, e_{24}, e_{34} und e_{13}). Für jede der vier Kanten (Leitungen) können jeweils drei verschiedene Optionen (l_1 mit Zuverlässigkeit $r(e_{ij}) = 0,8$, l_2 mit Zuverlässigkeit $r(e_{ij}) = 0,9$ und l_3 mit Zuverlässigkeit $r(e_{ij}) = 0,95$) mit jeweils unterschiedlichen Kosten gewählt werden. Für die vier Leitungen wurden die folgenden Optionen ausgewählt: $l_1(e_{12})$, $l_3(e_{24})$, $l_2(e_{34})$ und $l_1(e_{13})$. Die Kosten für das Netzwerk werden nach (1) als $C(N) = c(l_1(e_{12})) + c(l_3(e_{24})) + c(l_2(e_{34})) + c(l_1(e_{14})) = 10 + 56 + 48 + 12 = 126$ berechnet. Die Berechnung der All-Terminal Zuverlässigkeit erfolgt mit Hilfe einer vollständigen Enumeration sämtlicher Zustände s_i nach Formel 2. Für die Berechnung sind nur die fünf Zustände von Interesse, für die das Netzwerk verbunden ist. Die möglichen Zustände hierfür sind 1) keine Kante ausgefallen 2) e_{12} ausgefallen 3) e_{13} ausgefallen 4) e_{34} ausgefallen und 5) e_{24} ausgefallen. Da bei einem gleichzeitigen Ausfall von mehr als zwei Kanten das Netzwerk nicht mehr verbunden ist, müssen die anderen Zustände für die Berechnung von R_{All} nicht mehr berücksichtigt werden. Damit berechnet sich R_{All} aus der Summe der Wahrscheinlichkeiten der Einzelzustände als $R_{All} = (0,8*0,95*0,9*0,8)+(0,2*0,95*0,9*0,8)+(0,8*0,05*0,9*0,8)+(0,8*0,95*0,1*0,8)+(0,8*0,95*0,9*0,2) = 0,9104$. Damit ist das Netzwerk mit einer Wahrscheinlichkeit von ca. 91% komplett verbunden.

3 Ein Ansatz zur Planung von Netzwerkstrukturen unter Kosten- und Zuverlässigkeitsaspekten

3.1 CURE – Eine Schnittbasierte Reparaturheuristik

Mit der Reparaturheuristik CURE (CUt based REpair heuristic) wird ein Verfahren vorgestellt, welches in der Lage ist, beim Entwurf ökonomischer und zuverlässiger Netzwerktopologien Kosten und Zuverlässigkeit gleicher Maßen zu berücksichtigen. Die Reparaturheuristik CURE verwendet Netzwerkstrukturen, welche durch ein heuristisches Optimierungsverfahren erzeugt wurden und verbessert diese solange bis die All-Terminal Zuverlässigkeit des Netzwerkes größer als eine minimal geforderte Schranke R_0 ist. CURE geht dabei prinzipiell so vor, dass in einem ersten Schritt versucht wird, die Zuverlässigkeit von einzelnen Kanten zu erhöhen. Die Zuverlässigkeit der einzelnen Kanten wird dadurch erhöht, dass für eine Kante e_{ij} eine Option l_k mit einer höheren Ausfallsicherheit $r(l_k(e_{ij}))$ gewählt wird. Falls für jede mögliche Kante die maximal mögliche Zuverlässigkeit $r(l_k(e_{ij}))$ gewählt ist und das geforderte R_0 noch nicht erreicht ist, werden in einem zweiten Schritt zusätzliche Verbindungen in das Netzwerk eingefügt. Durch diese Vorgehensweise stellt CURE sicher, dass stets eine Netzwerkstruktur erzeugt wird, welche die Zuverlässigkeitsnebenbedingung $R_{All} \geq R_0$ erfüllt. Der Einsatz einer solchen Prozedur in einem heuristischen Optimierungsverfahren wie z.B. einem genetischen Algorithmus (GA) ermöglicht es, ungültige Lösungen, welche im Laufe des Optimierungsprozesses entstehen, hinsichtlich der gestellten Zuverlässigkeitsnebenbedingung zu reparieren.

Für die Auswahl der Kanten, deren Zuverlässigkeit im ersten Schritt von CURE vergrößert wird, wird die Theorie der minimalen Schnitte in Graphen verwendet. Ein Schnitt $C \subset K$ in einem Graphen G ist eine nichtleere Teilmenge der Knoten K. Jeder Menge C an Knoten wird die Menge E_C an Kanten e_{ij} zugeordnet, für die gilt $\forall e_{ij} \in E_C : i \in C$ und $j \notin C$. Löscht man nun alle Kanten E_C aus G so zerfällt G in zwei Subgraphen, welche jeweils aus den Knotenmengen C und $K \setminus C$ bestehen. Das Gewicht eines Schnittes C ist die Summe der Gewichte der Kanten E_C. Ein minimaler Schnitt ist der Schnitt mit dem geringsten Gewicht. Beim Entwurf zuverlässiger Netzwerktopologien kann das Konzept der minimalen Schnitte zum Finden der Verbindungen in einem Netzwerk genutzt werden, deren Ausfall das Netzwerk trennen würde. Für die Anwendung der CURE Heuristik wird jede Kante des Graphen G_N mit den

Kosten $c(l_{k+1}(e_{ij}))$ der nächst zuverlässigeren Option $l_{k+1}(e_{ij})$ bewertet. Durch die Wahl von Kanten, die zu einem Schnitt gehören, wird sichergestellt, das die Elemente des Graphen verbessert werden, durch deren gemeinsamen Ausfall das Netzwerk nicht mehr verbunden wäre. Durch die Verwendung des minimalen Schnittes wird die Kantenmenge gefunden, bei deren Verbesserung der geringste Kostenzuwachs entsteht. Im Folgenden wird der Ablauf von CURE beschrieben:

Prozedur CURE
Input: $G_N(K, E_N)$, $G(K, E)$, R_0
Queue $Q = \emptyset$, Q.append(G_N)
while (!Q.empty) & ($R_{All}(G_N) < R_0$)) do begin
  G_{work} = Q.first()
  assign weights (costs) to $e_{ij} \in G_{work}$: $c(l_k(e_{ij})) = \begin{cases} c(l_{k+1}(e_{ij})) & \text{if } k < k_{max} \\ c(l_k(e_{ij})) & \text{if } k = k_{max} \end{cases}$
  C = MinCut(G_{work}) (using the weights $c(l_k(e_{ij}))$)
  increase reliability $\forall e_{ij} \in E_C : l_k(e_{ij}) = \begin{cases} l_{k+1}(e_{ij}) & \text{if } k < k_{max} \\ l_k(e_{ij}) & \text{if } k = k_{max} \end{cases}$
  calculate $R_{All}(G_N)$
  $G_{N_1} = G_{work} \setminus \{C\}, G_{N_2} = C$
  if number of nodes in $(G_{N_1}) > 1$
    Q.append(G_{N_1})
  if number of nodes in $(G_{N_2}) > 1$
    Q.append(G_{N_2})
  Q.remove(G_{work})
  if (Q.empty) & ($\exists e_{ij} \in E_N$: $k < k_{max}$)
    Q.append(G_N)
end
if ($R_{All}(G_N) < R_0$) begin
  add $e_{ij} \in E \setminus E_N$ to G_N, $\forall e_{ij} \in G_N : l_k(e_{ij}) = l_1(e_{ij})$
  call CURE
end

Im ersten Schritt werden sämtliche Kanten aus G_N mit den Kosten $c(l_{k+1}(e_{ij}))$ der nächst zuverlässigeren Option bewertet. k gibt hierbei die Nummer der aktuellen Option an. Ist für eine der Kanten bereits die Option mit der höchsten Zuverlässigkeit gewählt ($k = k_{max}$), so wird die Kante mit den Kosten der aktuellen Option bewertet. Die Bewertung für das in Abbildung 1 vorgestellte Netzwerk zeigt Abb. 2. Der zu verbessernde Graph wird anschließend in die Queue eingestellt.

Während des Verbesserungsprozesses wählt die Prozedur jeweils den Graphen G_{work} der Queue und ermittelt für diesen die Kanten des minimalen Schnittes (MinCut). Zum Finden des minimalen Schnittes C wird das Verfahren aus [12] genutzt. Anschließend wird für alle Kanten des minimalen Schnittes die Zuverlässigkeit auf die nächste Option erhöht und die Kanten E_C aus G_{work} entfernt, so dass zwei Subgraphen G_{N_1} und G_{N_2} entstehen. Jeder Subgraph mit $|K| > 1$ wird am Ende der Queue angefügt. Wird durch die erste Verbesserung die geforderte Zuverlässigkeit R_0 nicht erreicht, so werden auf diese Weise rekursiv die neu entstanden Subgraphen G_{N_1}, G_{N_2} ebenfalls mittels CURE verbessert. Wenn die Queue leer ist und noch nicht für sämtliche Kanten die höchste Zuverlässigkeitsoption ($k = k_{max}$) gewählt wurde, so wird der komplette Graph erneut in die Queue eingestellt und CURE erneut gestartet. Reichen die in G_N enthaltenen Kanten mit ihrer maximal möglichen Zuverlässigkeit nicht aus um die geforderte Zuverlässigkeit R_0 zu erfüllen, so wird eine neue Kante aus G in G_N eingefügt und CURE erneut gestartet. Details zur Auswahl einer geeigneten Kante sind in [4] beschrieben.

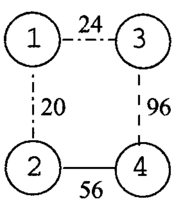


Abb. 2. Verbindungsbewertung für Netzwerk aus Abb. 1

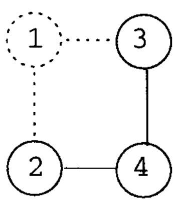


Abb. 3. Subgraph G_{N_1} für Netzwerk aus Abb. 1

Der Ablauf von CURE soll am Beispiel des Graphen G_N aus Abb. 2 erläutert werden. In Abb. 3 werden der Schnitt C sowie die Kanten E_C gepunktet dargestellt. Als minimaler Schnitt wird $C = \{1\}$ mit $E_C = \{e_{12}, e_{13}\}$ mit einem Gewicht von 44 ermittelt. Für die Kanten e_{12} und e_{13} wird die nächst bessere Option ($l_2(e_{12})$ und $l_2(e_{13})$) ausgewählt und diese anschließend aus G_N gelöscht. Nach dem Löschen entstehen die Subgraphen G_{N_1} mit den Knoten $\{2{,}3{,}4\}$ und G_{N_2} mit dem Knoten $\{1\}$. Da G_{N_2} nur einen Knoten besitzt, wird der Subgraph durch CURE nicht weiter betrachtet. Konnte durch die Verbesserung der Kanten e_{12} und e_{13} die geforderte Zuverlässigkeit R_0 nicht erreicht werden, so wird das Verfahren rekursiv auf G_{N_1} angewendet.

3.2 Entwurf eines genetischen Algorithmus unter Verwendung der CURE-Reparaturheuristik

Genetische Algorithmen [13] adaptieren die Prinzipien der Evolution (survival of the fittest) für die computergestützte Problemlösung. Dabei werden Suchoperatoren (Rekombination und Mutation) auf eine Menge (Population) von Problemlösungen über mehrere Iterationen (Generationen) angewendet. In jeder Generation wird die Qualität (Fitness) jedes Individuums mittels einer Zielfunktion (hier Formel (1)) bewertet. Über einen Selektionsoperator werden schlechte Lösungen aus der Population entfernt. GA verwenden als Hauptsuchoperator die Rekombination. Im vorliegenden Beitrag wird ein Uniform-Crossover Operator für die Rekombination eingesetzt. Durch den Mutationsoperator können zusätzlich kleine Veränderungen eines Individuums durchgeführt werden. Eine umfassende Erläuterung zu GA findet man in [13]. Durch die Operatoren des GA kann allerdings nicht sichergestellt werden, dass für sämtliche in der Population enthaltenen Lösungen gilt, dass $R_{All} \geq R_0$. Für das hier betrachtete Problem können durch die Suchoperatoren Lösungen erzeugt werden, die eine zu geringe

Gesamtzuverlässigkeit R_{All} aufweisen. Wird während des Suchprozesses ein solches Individuum erstellt, so wird dieses während der Bewertung der Fitness mit Hilfe der CURE Reparaturheuristik in eine valide (gültige) Lösung überführt.

4 Experimentelle Ergebnisse

4.1 Probleminstanzen

Die Leistungsfähigkeit des im vorherigen Abschnitts vorgestellten genetische Algorithmus soll an Hand von drei unterschiedlichen Testproblemen aus der Literatur untersucht werden:

Testproblem Jan5 - 5 Knoten Das einfachste hier untersuchte Netzwerk aus [3] besitzt 5 Knoten. Die optimalen Lösungen für $R_0 = \{0{,}85; 0{,}9; 0{,}93125; 0{,}95; 0{,}99; 0{,}995; 0{,}999\}$ sind aus [3] bekannt. Als „Proof of Concept" soll mit diesem Problem die Fähigkeit von CURE zum Finden optimaler Lösungen untersucht werden.

Testproblem Detter10 - 10 Knoten Das aus [3] entnommene Testproblem besitzt zehn Knoten, die zufällig auf einer 100x100 Fläche platziert wurden. Es werden optimale Lösungen für $R_0 = 0.95$ gesucht. Für das Problem sind $4^{10*9/2} = 1,237e27$ Lösungen möglich und eine optimale Lösung ist nicht bekannt. Als Kosten für die beste gefundene Lösung wird in [3] 5 881,42 angegeben.

Testproblem Türkei19 - 19 Knoten Dieses Problem ist das größte, welches in [3] untersucht wurde. Das Netzwerk muss mindestens eine All-Terminal Zuverlässigkeit R_0 von 0,99 aufweisen. Die Kosten der besten gefundenen Lösung werden in [6] mit 1 755 474 angegeben.

4.2 Experimentierumgebung

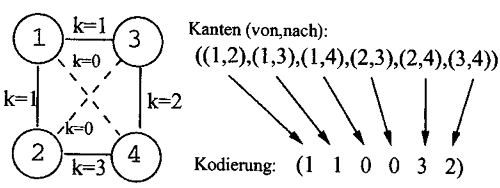


Abb. 4. Kodierung des Netzwerks aus Abb. 1 als Genom

Für die Experimente wurde ein genetischer Algorithmus mit überlappenden Populationen unter Verwendung der in Abschnitt 3.1 beschriebenen Reparaturheuristik CURE implementiert. Eine mögliche Lösung (Netzwerk) wird im GA als ein Vektor ganzer Zahlen der Länge $|K| * |K-1|/2$ kodiert. Hierbei gibt jede Zahl des Vektors an, welche Option $l_k(e_{ij})$ mit $0 < k \leq k_{max}$ für die jeweilige Kante e_{ij} verwendet wird. Eine Null gibt an, dass keine Verbindung zwischen Knoten i und j existiert. Abb. 4 zeigt die Kodierung des Netzwerks aus Abbildung 1. Nicht vorhandene Verbindungen sind gestrichelt dargestellt. Alle Experimente wurden mit einer Populationsgröße $M = 200$ durchgeführt. Die Initialisierung der Netzwerke für die erste Generation erfolgt zufallsbasiert. Das Initialisierungsverfahren erstellt mit 40% Wahrscheinlichkeit eine Verbindung zwischen zwei Knoten. Wurde eine Verbindung erstellt, wird gleichverteilt eine der zur Verfügung stehenden Optionen für die Zuverlässigkeit der Kante gewählt. Genügt ein durch das Initialisierungsverfahren erstelltes Netzwerk nicht den Zuverlässigkeitsanforderungen, so wird dieses mittels CURE repariert. Der GA arbeitet mit Uniform-Crossover, einem Flip-Mutator und einer Mutationswahrscheinlichkeit 0,01. Ein GA-Lauf wird nach maximal 1000 Generationen oder wenn in den letzten 20 Generationen keine bessere Lösung gefunden wurde abgebrochen. Die Bewertung der All-Terminal Zuverlässigkeit erfolgt für die Probleme Jan5 und Deeter10 mit Hilfe des exakten Verfahrens aus [10]. Für das Problem Türkei19 ist dieses Verfahren auf Grund der Netzwerkgrößen nicht mehr einsetzbar. Stattdessen wird hier eine einfache Monte-Carlo Simulation genutzt. Für jedes Problem wurden jeweils zehn voneinander unabhängige GA-Läufe durchgeführt. Für den Vergleich der mit CURE gewonnenen Ergebnisse mit bisherigen Ansätzen wurde der in [3] vorgestellte Strafterm-Ansatz implementiert.

4.3 Auswertung

Tabelle 1 fasst die Ergebnisse für das Problem Jan5 für unterschiedliche R_0 zusammen. Die Tabelle vergleicht die Reparaturheuristik CURE mit dem Strafwertansatz von [3] bezüglich den durchschnittlichen Kosten der jeweils gefundenen besten Lösung (min. Kosten), der Wahrscheinlichkeit P_{succ}, dass die optimale Lösung gefunden wird, und der durchschnittlichen Anzahl (⊘ Eval) der hierfür notwendigen Bewertungen von Lösungen. Für $R_0 = 0,85$ und $R_0 = 0,90$ wird in allen zehn Läufen die optimale Lösung bereits bei der Initialisierung der ersten Population gefunden. Darüber hinaus findet mit Ausnahme von $R_0 = 0,995$ der GA die optimale Lösung in mindestens drei von zehn Läufen. Für $R_0 = 0,995$ wird in allen zehn Läufen lediglich eine suboptimale Lösung mit den minimalen Kosten 4 382 gefunden. Ein Vergleich der notwendigen Fitnessbewertungen zwischen CURE und dem Strafwertansatz aus [3] zeigt, dass ein GA mit CURE deutlich weniger Fitnessbewertungen bei gleicher Lösungsqualität benötigt. Dies ist besonders positiv hervorzuheben, da die Bewertung von Lösungen (Berechnung von R_{All} sowie Ermittlung der Kosten) für größere Netzwerke sehr rechenaufwendig ist und im Vergleich dazu der Aufwand für CURE vernachlässigt werden kann. Zusammenfassend lässt sich für das einfache Problem Jan5 feststellen, dass ein GA mit der Reparaturheuristik CURE in der Lage ist, optimale bzw. nahezu optimale Lösungen zu finden. Tabelle 2 vergleicht die Leistungsfähigkeit des GAs für die Probleme Deeter10 und Türkei19. Es sind der Mittelwert der Kosten (⊘ Kosten) der besten gefundenen Lösungen über alle 10 Läufe, die Kosten des besten gefundenen Netzwerks (min. Kosten), die durchschnittliche Anzahl der pro Lauf durchgeführten Fitnessbewertungen (⊘ Eval.) und die durchschnittlich benötige Laufzeit (t_{conv}) jeweils für CURE und dem Strafwertansatz aus [3] angegeben. Als beste Lösung wurde unter Verwendung von CURE für das Problem Deeter10 ein Netzwerk mit den Kosten von 4 385,99 gefunden. Die mittleren minimalen Kosten gemittelt über alle 10 Läufe betragen 4 439,40. Beide Ergebnisse liegen deutlich unter den mit Hilfe eines Strafterms erzielten Lösungen.

Tabelle 1. Ergebnisse für Testproblem Jan5

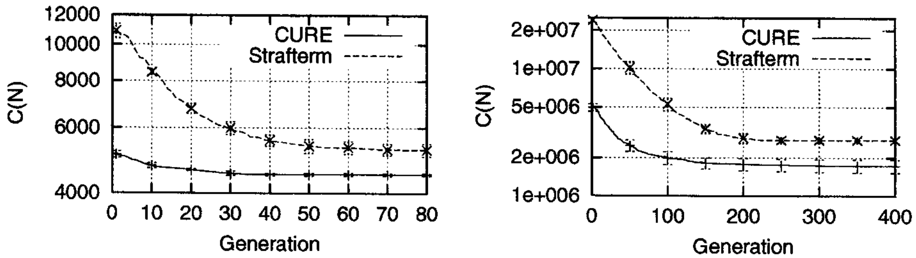
R_0	Kosten opt. Lsg.	CURE			Ansatz von [3]		
		min. Kosten	P_{succ}	⊘ Eval	min. Kosten	P_{succ}	⊘ Eval
0,999	5 522	5 522	0,3	360	5 522	0,8	40 560
0,995	4 352	4 382	0	-	4 352	0,4	23 200
0,99	3 754	3 754	1	1 560	3 754	1	31 400
0,95	2 634	2 634	1	270	2 634	1	118 880
0,93 125	2 416	2 416	1	1 170	2 416	1	25 560
0,9	2 184	2 184	1	200	2 184	1	26 160
0,85	1 904	1 904	1	200	1 904	1	4 640

In den Abbildungen 5(a) und 5(b) werden für die Probleme Deeter10 und Türkei19 die durchschnittlichen Kosten der jeweils gefundenen besten Lösung über die Anzahl der Generationen des GA dargestellt. Die Kurven beschreiben somit, wie sich die Qualität der besten gefundenen Lösungen während der GA-Läufe verändert. Die Ergebnisse zeigen, dass durch den Einsatz von CURE deutlich bessere Startlösungen ermittelt werden können und auch im weiteren Verlauf der Optimierung die Kosten der besten gefundenen Netzwerke beim Einsatz von CURE stets kleiner sind als beim Einsatz von Straftermen. Eine Analyse des Fitnessverlaufs zeigt, dass ein GA bei der Verwendung von CURE deutlich schneller konvergiert und somit zum Finden besserer Lösungen weniger Fitnessbewertungen als der straftermbasierte Ansatz benötigt. Es ist festzuhalten, dass der Strafterm-GA zwar teilweise weniger Bewertungen benötigt und stets eine kürzere Laufzeit hat, die Qualität der gefundenen Lösungen jedoch deutlich schlechter ist. Analysiert man die in Tabelle 2 für das Problem präsentierten Ergebnisse, so erfolgt durch den CURE-GA eine deutliche Verbesserung der bisher gefundenen besten Lösungen. Im Vergleich zu [6], wo die Kosten des besten gefundenen Netzwerkes für das Problem Türkei19 mit 1 755 474 angegeben wurde, konnte durch den Einsatz von CURE eine Netzwerkstruktur mit Kosten 1 577 755 ermittelt werden (eine Verringerung der Kosten um mehr als 10% bei gleichem R_0). Gegenüber der besten in [3] veröffentlichten Lösung (7 694 708) konnte sogar eine Verbesserung um zirka 80% erreicht werden.

	Deeter10		Türkei19	
	CURE	Strafterm	CURE	Strafterm
min. Kosten	4 385,99	4 948,79	1 577 755	2 499 080
⊘ Kosten	4 439,40	5 239,90	1 720 994	2 763 670
⊘ Eval.	4 830	6 190	33 330	20 550
t_{conv}	1 508	773	25 200	8 100

Tabelle 2. Vergleich der Ergebnisse für Testproblem Deeter10 und Türkei 19

Beim Problem Deeter10 konnten die Kosten der optimalen Lösung von 5881 auf 4386 bei gleichem R_0 verringert werden (eine Verbesserung von ca. 25%). Da das hier betrachtete Planungsproblem keinen Echtzeitanforderungen unterliegt, ist der zusätzliche Aufwand, der durch den Einsatz der Reparaturheursitik entsteht, in Anbetracht der deutlich höheren Lösungsqualität vertretbar.



(a) Deeter10 (b) Türkei19

Abb. 5. Kosten $C(N)$ der besten gefundenen Lösungen in Abhängigkeit von der Anzahl der Generationen des eingesetzten GAs. Ein GA findet bei der Verwendung von CURE deutlich bessere Lösungen als beim Einsatz des Strafkostenansatzes.

5 Zusammenfassung

Dieser Beitrag beschäftigt sich mit der Planung von Netzwerkstrukturen unter Kosten- und Zuverlässigkeitsaspekten. Hierbei werden zum Ermitteln von kostenoptimalen Netzwerkstrukturen, welche eine vorgegebene minimale Gesamtzuverlässigkeit R_0 aufweisen, heuristische Optimierungsverfahren eingesetzt. Im Gegensatz zu bisherigen Forschungsarbeiten, bei welchen Netzwerke, welche die geforderte Zuverlässigkeit R_0 nicht erfüllen, durch die Einführung von Straftermen schlechter bewertet werden, wird im vorliegenden Beitrag eine Reparaturheuristik CURE entwickelt und eingesetzt. CURE bestimmt mit Hilfe der minimalen Schnitte für ein Netzwerk die für eine Reparatur relevanten Kanten und liefert eine Netzwerkstruktur, welche die vorgegebene Zuverlässigkeitsrestriktion erfüllt. Die Reparaturheuristik CURE wurde in einem genetischen Algorithmus implementiert und deren Leistungsfähigkeit im Rahmen einer experimentellen Studie mit dem bisher üblichen Straftermansatz verglichen.

Die Leistungsfähigkeit von CURE wurde für drei Probleminstanzen aus der Literatur mit unterschiedlicher Komplexität getestet. Die Ergebnisse zeigen, dass CURE mit geringem Aufwand durchweg bessere Lösungen findet als der bisherige Straftermansatz. Bei den zwei praktisch relevanten Problemstellungen konnten durch den Einsatz von CURE Netzwerkstrukturen gefunden werden, welche um bis zu 25% geringere Kosten bei gleicher Gesamtzuverlässigkeit aufweisen.

Literatur

1. Garey M. R., Johnson D. S.: Computers and Intractibility: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, San Fransisco, 1979.
2. Colbourn, C.: The Combinatorics of Network Reliability. Oxford University Press, Oxford, 1987.
3. Deeter D., Smith A.: Economic Design of Reliable Networks. IIE Transcations, Special Issue on Economics of Reliable Engineering 30:1161–1174, 1998.

4. Reichelt D., Rothlauf, F., Gmilkowsky P.: Designing Reliable Communication Networks with a Genetic Algorithm using a Repair Heuristic. Evolutionary Computation in Combinatorical Optimization:177–187, Springer, Berlin, 2004.
5. Duarte S., Barán B.: Multiobjective Network Design Optimisation Using Parallel Evolutionary Algorithms. In XXVII Conferencia Latinoamericana de Informática CLEI'2001, Merida, Venezuela, 2001.
6. Barán B.,Duarte S.,Benítez D.: Telecommunication Network Design with Parallel Multi-objective Evolutionary Algorithms. In IFIP/ACM Latin America Networking Conference, La Paz, Bolivia, 2003.
7. Dengiz B., Altiparmak F., Smith A.E.: Local search genetic algorithm for optimal design of reliable networks. IEEE Transactions on Evolutionary Computation 1(3):179–188, 1997.
8. Jan R. H., Hwang F.J., Cheng S.T.: Topological optimization problem of communication networks subject to a reliability constraint. IEEE Transactions on Reliability 42:63–70, 1993.
9. Baran B.,Laufer F.: Topological optimization of reliable networks using a-teams. In World Multiconference on Systemics, Cybernetics and Informatics - SCI 99 and ISAS 99, Vol 5, 1999.
10. Yunbin C., Jiandong L., Jiamo C.: A new algorithm for network probabilistic connectivity. In IEEE military communication conference, 1999.
11. Konak A., Smith. A.: An improved general upperbound for all-terminal network reliability. Technical report, University of Pittsburgh, 1998.
12. Stoer M., WagnerF.: A Simple Min Cut Algorithm. Algorithms. In Algorithms - ESA '94 Second Annual European Symposium:141–147, Springer, Berlin, 1994.
13. Goldberg D. E.: Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading, 1989.