Simulated Annealing und Tabu Search

Marko Tosic

1. April 2003

Inhaltsverzeichnis

1 Einleitung **3**

2 Simulated Annealing **5**
2.1 Statistische Mechanik 5
2.2 Simuliertes Abkühlen 6
2.3 Der Algorithmus 6
2.4 Vor- und Nachteil 7
2.5 Anwendungsbeispiel: Das Traveling Salesman Problem 7

3 Tabu Search **9**
3.1 Grundlegende Idee 9
3.1.1 Intensivierung 10
3.1.2 Diversifikation 10
3.2 Der Algorithmus 10
3.3 Vor- und Nachteile 10
3.4 Anwendungsbeispiel: Grafen Färben 11

4 Anhang **13**

1 Einleitung

In vielen Bereichen der Industrie, Wirtschaft und Wissenschaft spielen Probleme einer Rolle, bei denen es darum geht ein komplexes System mit vielen Variablen auf ein oder mehrere Kriterien hin zu optimieren. Wie wir aus der theoretischen Informatik wissen, sind viele dieser Optimierungsprobleme, bzw. bereits ihre Entscheidungsvarianten NP-Vollständig, so dass wir nicht damit rechnen können, einen effiziente Algorithmus zu finden, der uns in jedem Fall die optimale Lösung in akzeptabler Zeit berechnet. Um diese Probleme dennoch behandeln zu können, weichen wir auf heuristische Methoden aus und geben uns mit möglichst guten Ergebnissen zufrieden.

Wir wollen zunächst von den konkreten Optimierungsaufgaben abstrahieren und die Problemstellung formalisieren, bevor wir uns mit einigen Lösungsmöglichkeiten, hier insbesondere Simulated Annealing und Tabu Search, beschäftigen.

Unser Problem besteht darin für eine bestimmte Funktion $f : S \to R$, genannt die *Kostenfunktion*, das globale Optimum zu finden. Je nach Problemstellung kann es sich dabei um das globale Maximum oder das globale Minimum handeln. Wir wollen und hier auf globale Minima beschränken, eine Übertragung auf globale Maxima ist nicht schwer. Die Definitionsmenge der Kostenfunktion ist der sogenannte *Suchraum* S, seine Elemente, die möglichen Lösungen des zu optimierenden Systems, sind meistens d-dimensionale Vektoren, die den Parametern des zu optimierenden Systems entsprechen. Aus diesem Grund spricht man bei einer bei einer Belegung einer Variablen $x \in S$ auch von einer *Parameterkonfiguration* bzw. von einer *Lösung*.

Ein sehr einfaches System wäre beispielsweise das mit einem Parameter und der Kostenfunktion, die in Abbildung 1 dargestellt ist.

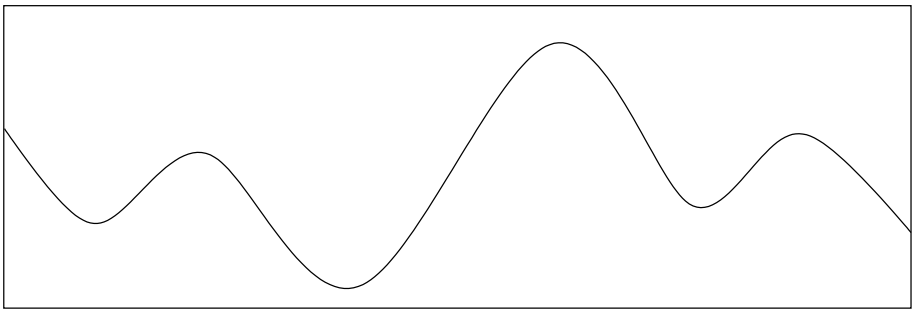

Abbildung 1: Einfache Kostenfunktion

Um nun nach dem Optimum zu suchen, in unserem Fall also nach dem Minimum, kann man an einer zufällig gewählten Stelle x mit der Suche beginnen. Man betrachtet nun die *Nachbarschaft* $N(x)$ von dieser Stelle, das sind alle Stellen, die höchstens einen anzugebenden Abstand δ von x haben: $N(x) := \{x_i \big| |x - x_i| < \delta\}$. In dieser Nachbarschaft wählen wir nun zufällig einige Werte x_j aus und schauen nach, ob die Kosten an dieser Stelle kleiner sind, also ob

$f(x_j) < f(x)$. Ist dies der Fall, so ist x_j unser neues x und die Suche kann iteriert werden. Kann kein kostengünsigeres x_j gefunden werden, so sind wir fertig. Dieses Verfahren wird auch als *zufällige lokale Suche* bezeichnet. Das große Problem ist, das schon bei einem lokalen Minimum die Suche abbrechen kann. Der Ursache dafür liegt darin, dass die Folge von gewählten Punkte streng monoton ist, so dass aus einem lokalen Optimum unter Umständen nicht mehr entkommen werden kann. Diese Situation ist in Abbildung 1 dargestellt.

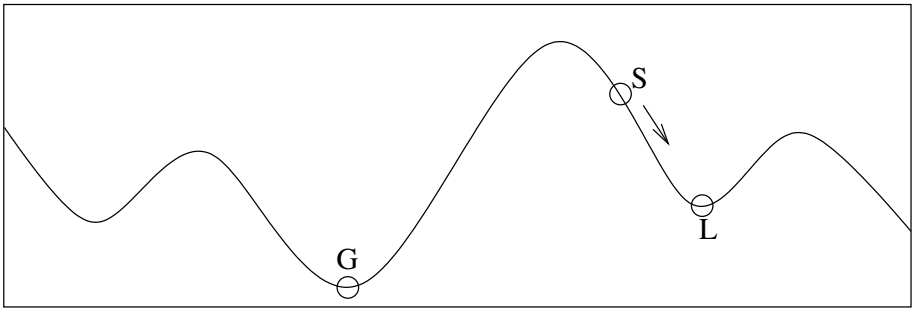

Abbildung 2: Lokale Suche
Die Suche hat bei S begonnen und ist im nächsten lokalen Minimum L steckengeblieben statt das globale Optimum bei G zu finden.

Algorithmus: Zufällige Lokale Suche

Schritt 1 Wähle eine beliebige Lösung i aus S.

Schritt 2 Wähle zufällig ein j aus der Nachbarschaft $N(i)$ aus. Falls $f(j) < f(i)$, so setze $i = j$.

Schritt 3 Falls Endbedingung eingetroffen ist stoppe, sonst mache weiter mit Schritt 2.

Um dieses Manko zu überwinden wurden verbesserte Verfahren entwickelt. Zwei davon werden wir hier näher beschreiben, namentlich *Simulated Annealing* und *Tabu Search*.

2 Simulated Annealing

Im Jahre 1982 stellten Kirpatrick, Gelatt und Vecchi in ihrem Forschungsbericht [Kirk82] einen Zusammenhang zwischen den Vorgängen in abkühlender Materie und allgemeinen Optimierungsproblemen her. Sie wandten ihre Erkenntnisse erfolgreich an, um schwierige Aufgaben im Rechnerentwurf und das Traveling Salesman Problem zu lösen. Das Simulated Annealing Verfahren hat seinen Ursprung somit in der Physik, genauer in der statistischen Mechanik, deren Inhalte wir kurz aufzeigen möchten.

2.1 Statistische Mechanik

Die Statistische Mechanik ist ein Teilgebiet der Festkörperphysik. Sie liefert Methoden zur Analyse der Aggregateigenschaften von Materialproben. Da solche Proben aus einer großen Anzahl Atomen bestehen und daher die Beobachtung aller einzelnen Atome nicht realisierbar ist, wird nur das wahrscheinlichste Verhalten des Gesamtsystems im thermischem Gleichgewicht bei einer bestimmten Temperatur in Experimenten untersucht. Um diese Verhalten zu characterisieren wird über ein sogenanntes *Ensemble* von identischen Systemen gemittelt.

In dem Ensemble wird nun jede *Konfiguration* r_i mit ihrer *Boltzmann Wahrscheinlichkeit* gewichtet. Die Boltzmann Wahrscheinlichkeit ist der Term

$$\exp^{\frac{-E(r_i)}{k_B T}}$$

Dabei ist $E(r_i)$ die Energie der Konfiguration, k_B ist die Boltzmann Konstante und T ist die Temperatur.

Ein Ziel ist es, möglichst reine Christallstrukturen herzustellen, also solche mit wenigen Fehlern. Dazu ist das Verhalten des zugrundeliegenden Stoffes bei seiner unteren Grenztemperatur entscheidend. Es hat sich gezeigt, das die Aggregateigenschaften eines Materials nicht ausschließlich von seiner Temperatur, sonder auch von der Art und Weise abhängt, wie diese Temperatur erreicht wird. So kann das Abkühlen des gleichen Materials in einem Fall zu einem Kristall führen, ein anderes Mal entsteht ein Glass. Um einen möglichst reinen Kristall zu erhalten wird wie folgt vorgegangen:

- Das Material wird auf die Ausgangstemperatur erhitzt.
- Folgende Schritte werden wiederholt, bis die Atome des Materials ihre Entgültige Position eingenommen haben:
  - Die Temperatur wird um eine Stufe herabgesetzt.
  - Es wird gewartet, bis alle Atome einen (für diese Temperatur) festen Platz eingenommen haben.
- Waren die Temperaturschritte richtig bemessen, so hat man nun einen reinen Kristall.

Metropolis et al. haben in [Metro53] einen einfachen Algorithmus entworfen, mit dem das Verhalten einer Ansammlung von Atomen in thermischen Gleichgewicht bei einer gegebenen Temperatur simuliert werden kann. In jedem Schritt dieses Algorithmus wird einem Atom ein wenig verschoben, was in einer Veränderung ΔE der Energie resultiert. Falls $\Delta E \leq 0$ und somit die Gesamtenergie des Systems abnimmt, so wird die Verschiebung akzeptiert. Falls $\Delta > 0$, so wird die neue Konfiguration mit der Boltzmann Wahrscheinlichtkeit $e^{\frac{\Delta E}{k_B T}}$ trotzdem akzeptiert. Werden diese Schritt sehr oft wiederholt, so simuliert man damit die thermische Bewegung der Atome bei konstant gehaltener Temperatur. Nach einer gewissen Zeit haben die Atome die Position eingenommen, die für die festgehaltene Temperatur die minimale Energie hat und es wird keine Atombewegung mehr stattfinden. Dies ist genau, was wir für das oben angegebenen Schema benötigen.

Die Temperaturschritte werden durch einen *Kühlplan* festgelegt. Dieser sollte so aufgebaut sein, das die Schrittweite mit sinkenden Temperaturen abnimmt. Der Grund dafür ist, dass sich die Atome bei hohen Temperaturen noch relativ frei bewegen können. Es entsteht als kein Schaden, wenn schneller abgekühlt wird. Bei niedrigen Temperaturen, insbesondere in der Nähe des Gefrierpunktes werden die Bewegungen der Atome immer träger, so dass sie sich bei zu schnellem Abkühlen nicht in der günstigsten Postion aufhalten, was zu Fehlern im Christall führt.

2.2 Simuliertes Abkühlen

Die bei der Erstellung von reinen Christallen zu bewältigende Aufgabe hat vieles gemeinsam mit den Problemen aus der kombinatorischen Optimierung. Allerdings gibt es bei diesen Problemen im allgemeinen keine Temperatur.

Wir führen daher eine effektive Temperatur für die Optimierung ein und simulieren den Abkühlungsprozess. Dabei ist diese Temperatur lediglich ein Kontrollparameter. Die Kostenfunktion übernimmt die Rolle der Energie.

Würde man beim Durchlaufen des Algorithmus nur kostenreduzierende Konfigurationen übernehmen, so käme das einem sehr schnellen Abkühlen gleich, was beim realen Abkühlungsprozess einen Christall mit vielen Defekten zu Folge hätte. Glücklicherweise ist dem nicht so, da mit einer gewissen Wahrscheinlichkeit ja auch energievergrößernde bzw. kostensteigernde Konfigurationen übernommen werden. Dies hat zur Folge, dass der Algorithmus auch mit dem Problem der lokalen Optima umgehen kann.

Der Kühlplan soll ebenso wie beim realen Abkühlprozess auch mit sinkenden Temperaturen kleinere Schritt durchführen. So werden zu Konfigurationsänderungen, die zu großen Veränderungen führen zu Beginn durchgeführt während zum Ende hin nur noch kleinere Veränderungen duchgeführt werden.

2.3 Der Algorithmus

Schritt 1 Wähle eine beliebige Lösung i aus S, eine Start Temperatur T aus dem Kühlplan. Merke die Anfangslösung als die bisher Beste $b = i$.

Schritt 2 Führe die Metropolis Simulation durch:

Schritt 2a Wähle zufällig eine Konfiguration j aus , wobei die die Wahrescheinlichkeit gewählt zu werden für „nahe“ Konfigurationen mit sinkender Temperatur steigen sollte.

Schritt 2b Falls $f(j) <= f(i)$, so setze $i = j$.

Schritt 2c Falls $f(j) <= f(b)$, so setze $b = j$.

Schritt 2d Falls $f(j) > f(i)$, so setze $i = j$ mit einer Wahrscheinlichkeit von $e^{\frac{f(j)-f(i)}{k_B * T}}$

Schritt 3 Falls kürzlich noch Verbesserung von $f(b)$ stattgefunden hat gehe zu Schritt 2.

Schritt 4 Falls $T < \epsilon$ gib b aus und stop.

Schritt 5 Führe den nächsten Temperaturschritt gemäß Kühlplan durch und gehe zu Schritt 2.

2.4 Vor- und Nachteil

\+ Lokale Optima sind kein großes Problem

\+ Der Algorithmus hat sich etabliert, viele erfolgreich Beispiele. sind verfügbar

\+ Das Verfahren ist leicht auf neue Probleme übertragbar.

\+ Der Algorithmus ist einfach zu implementieren.

\- Der gute Erfolg ist theoretisch nur schwer zu beweisen.

\- Ermitteln eines guten Kühlplans kann schwer sein.

2.5 Anwendungsbeispiel: Das Traveling Salesman Problem

Beim dem Traveling Salesman Problem geht es darum eine Rundreise durch N Städte zu finden, welche die Kosten der Reise minimiert und wieder am Ausgangsort endet. Um aus der Problemstellung einen Simulated Annealing Algorithmus ableiten zu können, brauchen wir Hilfsmittel. Einerseits, um die Tour zu repräsentieren und andererseits, um eine Tour zufällig umzuordnen. Kirkpatrick, Gelatt und Vecchi lösten das Problem wie folgt: Eine Tour wurde einfach dargestellt als eine Permutation der Zahlen 1 bis N, die angibt in welcher Reihenfolge die Städte besucht werden. Sie verwendeten die folgende Umordnungsstrategie: In jedem Zug wird die Richtung, in der ein zufällig gewähltes Segment der Tour duchlaufen wird umgekehrt. Dies ist in Abbildung 3 dargestellt.

Der Abkühlplan wurde empirisch bestimmt, wobei die Temperatur, bei der sich die Segemente frei bewegen konnten mit $N^{0.5}$ angesetzt wurde und Temperaturen unterhalb von 1 als kalt galten.

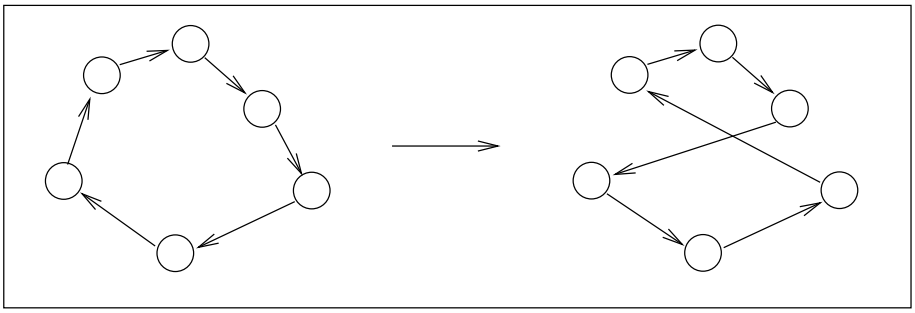

Abbildung 3: Bei den unteren drei Knoten wird die Richtung umgedreht

Mit diesem Ansatz wurde für das Traveling Salesman Problem mit 6000 Städte eine gute Route berechnet.

3 Tabu Search

Tabu Search zielt ebenfalls darauf ab, das Problem der lokalen Optima zu beseitigen. Es handelt sich um eine Metastragie, die auf andere, traditionelle Optimierungsmethoden angewandt wird. Tabu Search wie wir es heute kennen geht auf F. Glover ([Glov86])zurück.

3.1 Grundlegende Idee

Die einfachen Sucherverfahren verwenden für ihre Suche nur lokale Informationen, also beispielsweise den Wert der Kostenfunktion für Punkte aus der Nachbarschaft des Suchpunktes. Die Idee besteht nun darin, zusätzlich das vorhandene Wissen aus der bisherigen Suche zu verwenden. Dazu wird ein Gedächnis eingeführt, in dem Informationen über den bisherige Verlauf der Suche abgelegt werden. Mit dieser Information kann die Auswahl eine Wertes x' aus der Nachbarschaft von x dadurch beeinflusst werden, dass bestimmte Bereiche verboten werden – sie sind Tabu. Weiterhin kann sich dieses Gedächnis, ganz wie auch das Gedächnis eines Menschen, im Laufe der Zeit verändern und an die aktuelle Situation anpassen. Um aus lokalen Optima wieder entfliehen zu können müssen auch Züge erlaubt werden, die nicht-verbessernd sind. Sobald man diese jedoch zulässt, besteht die Gefahr, dass ein bereits gefundenes Optimum wiederentdeckt wird. Der Algorithmus wäre wieder in einem lokalen Optimum gefangen, diesmal in einer Schleife, bei der er das Optimum in einem Zug verlässt um es im nächsten wiederzufinden. Mit Hilfe des Gedächnisses könne wir dieses Szenario verhindern, indem kürzlich besuchte Optima zum Tabu erklärt werden. Es handelt sich also um eine Art Kurzeitgedächnis für schon besuchte Lösungen.

Man könnte diese Gedächnis einfach realisieren, indem man die letzten k besuchten Lösungen in einer Liste speichert. Allerdings wäre eine solche Liste in der Benutztung unpraktikabel, da der Speicherverbrauch enorm wäre. Aus diesem Grund wird die sogenannte *Tabu Liste* L als Liste der letzten Züge gespeichert. Wenn S der Suchraum ist, so bezeichnet $M(i), i \in S$ die Menge der aktuell möglichen Züge von i aus um einen potentiellen neuen Suchpunkt zu erreichen. Der Nachbarschaftsbegriff der lokalen Suche wird dadurch eingeschränkt auf die Punkte, die von i aus mit Zügen aus $M(i)$ erreicht werden können. Da nur eine begrenzte Anzahl der vorhergehenden Züge gespeichert wird werden Kreise immernoch nicht völlig ausgeschlossen, der kürzeste Kreis ist jedoch mindstens um Eins länger als die Liste Elemente speichern kann. Weiterhin kann es durch die Einschränkung bei der Tabu Liste dazu kommen, dass Lösungen tabuisiert werden, die noch gar nicht besucht wurden. Dem kann durch die Einführung von Aspirations- Bedingungen abgeholfen werden, die einen Zug, der in der Tabu Liste gespeichert ist, dennoch zulassen, sofern die Bedingung für ihn erfüllt ist. Die Aspirations-Bedingungen werden ebenfalls in einer Liste verwaltet. Um die Suche noch besser steuern zu können gibt es bei Tabu Search zwei gängige Methoden den Suchort zu beinflussen: *Intensivierung* und *Diversifikation.*

3.1.1 Intensivierung

Bisher haben wir das Kurzzeitgedächnis beschrieben, welches uns hilft das wiederholte Finden von Lösungen zu verhindern. Führt uns dies jedoch nicht mehr weiter, so setzte ein Mittelzeitgedächnis ein, welches die Suche an erfolgsversprechenden Stellen, die es sich gemerkt hat, intensiviert. Die Intensivierung kann folgendermassen von statten gehen: Für die Zeit der Intensivierung wird die Kostenfunktion um einen Term f_I erweitert. Dieser Term „bestraft“ Suchstellen, die weiter weg liegen von der Stelle die genauer untersucht werden soll durch einen schlechteren Wert. Dadurch werden weiter entfernte Stellen gemieden und die aktuelle Stelle genauer nach guten Lösungen untersucht.

3.1.2 Diversifikation

Wenn auch die Intensivierung keine Verbesserung mehr bewirkt, so liegt es nahe an einer ganz andere Stelle weiterzusuchen. Daher gibt es noch ein Langzeitgedächnis, welches die Suche dazu veranlasst, an anderer Stelle fortzufahren. Dies geschieht wiederum dadurch, dass die Kostenfunktion um einen weiteren Term f_D ergänzt wird, der nahegelegene Suchstellen duch einen schlechteren Wert bestraft. Die verwendete Kostenfunktion ist also $f + f_I + f_D$.

3.2 Der Algorithmus

Fasst man alles bisher gesagt zusammen, so kommt man, wendet man Tabu Search auf die Zufällig Lokale Suche an, zu dem folgenden Algorithmus:

Schritt 1 Wähle eine beliebige Lösung i aus S. Sei $i^* = i$ und $k = 0$.

Schritt 2 Setze $k = k + 1$ und erzeuge eine Teilmenge V^* der Lösungen aus $N(i, k)$, so dass ihre Elemente entweder keine Tabus verletzen oder eine Aspirations-Bedingung erfüllen.

Schritt 3 Wähle bezüglich $f + f_I + f_D$ den besten nächsten Zug j aus V^* aus und setze $i = j$

Schritt 4 Falls $f(i) < f(i^*)$, so setze $i^* = i$.

Schritt 5 Aktualisiere die Tabu Liste sowie die Aspirations-Liste.

Schritt 6 Falls eine Abbruchbedingung erfüllt ist stoppe, andernfalls gehe zurück zu Schritt 2.

3.3 Vor- und Nachteile

\+ Duch Intensivierung und Diversifikation ist es zumindest theoretisch immer möglich das globale Optimum zu finden.

\+ Es wird der Suche ermöglicht, aus lokalen Optima wieder zu entfliehen

+ Die Suche wird durch das durch Intensivierung und Diversifikation in gezielt in vielversprechende Bereiche geleitet.

+ Tabu Search kann auf viele verschiedene Suchalgorithmen aufgesetzt werden, was vielversprechende neue Lösungsansätze ermöglicht.

+ Tabu Search hat sich in der Praxis bereits als effizient bewährt.

+ Tabu Search ist leicht zu implementieren.

- Es ist nicht immer leicht und offensichtlich Tabu Search so zu formulieren, dass die Optimale Lösung überhaupt gefunden werden kann.

3.4 Anwendungsbeispiel: Grafen Färben

Es ist nicht immer so leicht wie es scheint, die Tabu Suche auch anzuwenden, wie hier gezeigt werden soll.

Gegeben Sei ein Graf $G = (V, E)$. Es solle eine Einfärbung der Knoten gefunden werden, bei der zwei adjazente Knoten jeweils unterschiedliche Farben haben. Dabei soll die Anzahl der benötigten Farben minimiert werden.

Zunächste ein Beispiel für einen Ansatz zur Tabu Suche, der nicht in jedem Fall zum Ziel führt. Dazu wählen wir als mögliche Lösungen S die Menge aller Einfärbungen des Grafen, die nicht mehr als eine bestimmte Anzahl an Farben benutzt. Diese Obergrenze kann beispielsweise die Anzahl der Farben der zu Anfang gewählten Lösung sein. Als Nachbarschaft $N(i)$ einer Lösung i wählen wir nun alle Lösungen aus, die man aus i erhält indem genau ein Knoten seine Farbe ändert. Auf den ersten Blick erscheint dies vielleicht ein guter Ansatz zu sein, allerdings kann mit ihm nicht in jedem Fall das Optimale Ergebnis erreicht werden, wie Abbildung 3.4 zeigt:

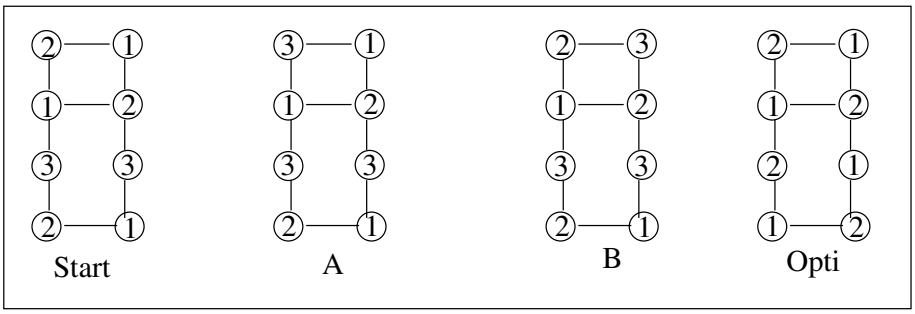

Abbildung 4: Nicht optimale Lösung des Grafen einfärbens

Alternativ stellen wir die Lösung des Problems mittels Tabu Search aus [Hert87] vor.

Eine Einfärbung der Knoten von G mit k Farben kann anders ausgedrückt werden als eine Partition der Knotenmenge V in k unterschiedliche Mengen

$V_1, ..., V_k$. Für die Definition der Menge S der möglichen Lösungen wird die Bedingung der Eindeutigkeit gelockert und Tabu Search wird verwendet um eine Färbung des Grafen mit k Farben zu ermitteln, sofern dies möglich ist. Eine mögliche Lösung ist dabei jede Partition $V_1, \ldots, V_k$ von V in k Mengen. Wir definieren $E(V_r)$ als die Menge der Kanten, deren Endknoten beide in V_r enthalten sind. Das Ziel ist es nun $\sum_{r=1}^{k} |E(V_r)|$ zu minimieren. Eine Lösung mit Wert 0 stellt eine gültige Einfärbung des Grafen mit k Farben dar.

Einige praktisch relevante Probleme können umformuliert werden zu einem Grafen-Einfärbungs Problem.

4 Anhang

Glossar

Annealing Annealing ist der Fachausdruck für ein ein Abkühlverfahren, bei dem das Ziel ist, christalline Strukturen in dem abkühlenden Material zu erzeugen.

Aspiration-Bedingung Bei Tabu Search erlaubt es eine erfüllte Aspirations-Bedingung, dass eine Lösung ausgewählt wird, obwohl Sie gerade Tabu ist.

Boltzmann-Konstante Eine Naturkonstante, benannt nach ihrem Finder.

Diversifikation An einer anderen Stelle weitersuchen.

Ensemble Ein Ensemble beschreibt in der Statistischen Mechanik eine Menge von identischen Systemen.

Intensivierung Die Suche an einer Stelle vertiefen.

Konfiguration Eine *Konfiguration*, auch *Parameterkonfiguration* oder *Lösung*, beschreibt den Zustand des zu optimierende Systems. Sie umfasst beispielsweise den Inhalt aller Variablen zu einem Zeitpunkt.

Kostenfunktion Die Kostenfunktion repräsentiert die Güte des zu optimierenden Systems.

Lösung Eine Lösung ist ein gültiger Zustand des Systems. Siehe auch *Konfiguration*

Nachbarschaft Unter einer *Nachbarschaft* versteht man alle Lösungen des Suchraums, die höchstens eine bestimmte Entfernung vom Suchpunkt haben, wobei anzugeben ist, was man unter „Entfernung“ versteht (euklidischer Abstand etc.).

Suchraum Der Suchraum besteht aus allen möglichen Lösungen (gute wie schlechte) des zu optimierenden Systems. In ihm soll die optimale Lösung gefunden werden.

Tabu Auch *Taboo*. Beschreibt in der Sprache der Ureinwohner Tongas etwas, dass heilig ist und deshalb nicht berührt werden darf. In Tabu Search werden Lösungen als Tabu bezeichnet, die momentan nicht verwendet werden sollen, da sie schon bekannt sind.

Thermisches Gleichgewicht Thermische Gleichgewicht ist vorhanden, wenn der betrachtete Gegenstand überall dieselbe Temperatur hat.

Literatur

[Kirk82] Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P. (1982). Optimization by Simulated Annealing. IBM.

[Kirk83] Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P. (1983). Optimization by Simulated Annealing. Science, 220:671-680.

[Metro53] N. Metropolis, A. Rosenbluth, A. Teller, and E. Teller, J. Chem. Phys 21, 1087 (1953).

[Glov86] Glover, F. (1986). Future Paths for Integer Programming and Links to Artificial Intelligence. Computers and Operations Research 13:533-549.

[Hert87] A. Hertz, D. de Werra. (1087). Using Tabu Search Techniques for Graph Coloring. Computing 39:345-351.