

Tabu-Search

[Übersicht/ Einführung in eine moderne Meta-Heuristik]

Clemens Heppner
Universität Hamburg Fachbereich Informatik
Vogt-Kölln-Straße 30
Hamburg, Germany
5heppner@informatik.uni-hamburg.de

ABSTRACT

Tabu-Search ist ein flexibles Verfahren, welches zum Lösen von schwierigen Optimierungsproblemen verwendet werden kann. Erstmals 1987 von Glover beschrieben, ist es heute ein intensiv untersuchtes Gebiet und ist bei sehr vielen unterschiedlichen Problemen angewandt worden.

Ich versuche Tabu-Search von seinen grundlegenden bis zu seinen erweiterten Konzepten zu beleuchten und einen groben Überblick zu verschaffen, der dazu ermuntern kann, Tabu-Search einmal selber auszuprobieren.

General Terms

Tabu-Search, Optimization

Keywords

Optimierung, Tabu-Search, Aspiration, Diversification, Intensification

1. EINLEITUNG

In Wissenschaft und Wirtschaft ist das Lösen von schwierigen Optimierungsproblemen ein häufiges und wichtiges Problem. Da es bei solchen Problemen keine effiziente Lösung gibt, sind heuristische Verfahren in diesem Bereich besonders wichtig.

Tabu-Search zeigte sich in der Vergangenheit als vielversprechende Methode um gute Lösungen zu finden (z.B. [8]). In diesem Paper soll der grundsätzliche Ansatz gezeigt und seine erweiterte Form kurz dargelegt werden. Das erste umfassende Lehrbuch über Tabu-Search von Glover, Laguna [5] behandelt die Thematik entsprechend ausführlicher.

Erstmals wurde Tabu-Search in seiner jetzigen Form von Fred Glover [3] beschrieben. Heutzutage zeigen die vielfältigen Anwendungen dieser Technik ihre Flexibilität, auch wenn es bislang keine exakten mathematischen Beweise dafür gibt.

2. SUCHVERFAHREN

Die meisten Optimierungsprobleme lassen sich als Maximierungsproblem auf einer Bewertungsfunktion auffassen. (Hier wird der Einfachheit halber nur dieser Fall betrachtet. Für eine genauere Definition siehe [1].) Im Folgenden ist $x \in X$ eine Lösung im Lösungsraum, $N(x)$ die Menge aller Nachbarlösungen von x , $f(x)$ die Bewertungsfunktion, m_a ein Zug von x nach x' , so dass gilt $x \xrightarrow{m_a} x'$ und $M(x)$ die Menge aller Züge, die von x aus möglich sind.

Zum Lösen von solchen Optimierungsproblemen gibt es neben Tabu-Search noch mehrere andere heuristische Verfahren, deren bekannteste Vertreter Simulated Annealing[9] und Genetische Algorithmen[7] sein dürften.

Tabusearch ist ein iteratives Verfahren, welches von einer Startlösung des Problems aus versucht, diese schrittweise zu verbessern. Der simpelste Vertreter eines solchen *Gradientenaufstiegsverfahrens* ist das Hillclimbing oder Local Search.

Algorithmus 1: Simple Hillclimbing

- (1) $s^* \leftarrow$ Startlösung aus X
- (2) **while** $\exists s \in N(s^*) : f(s) > f(s^*)$
- (3) $s^* \leftarrow s$

Dieser Algorithmus verbessert in jedem Schritt die Bewertung der Lösung s^* , bis ein Optimum gefunden wird. Hillclimbing verfügt jedoch über keinen Mechanismus um aus lokalen Optima zu entkommen und kann deshalb häufig nicht eingesetzt werden.

Tabusearch ist im einfachsten Sinne eine Modifikation dieses Algorithmus.

2.1 Tabu-Search

Die grundlegende Idee hinter Tabu-Search ist es, die Effizienz der Suche zu steigern, indem nicht nur die aktuelle Nachbarschaft einer Lösung betrachtet wird, sondern auch der bisherige Verlauf der Suche. In einem oder mehreren *Speichern* oder *Tabulisten* werden Informationen über den bisherigen Suchverlauf gespeichert. Diese Informationen werden später eingesetzt, um die Nachbarschaft einer Lösung einzugrenzen.

Die simpelste Version einer Tabuliste T_1 speichert einfach nur die vorhergegangenen Lösungen und verbietet sie für alle folgenden Suchschritte. Dadurch kann die Suche nicht im Kreis laufen oder ewig in einem lokalen Optima hängen bleiben, da irgendwann der gesamte Bereich des Optimums *tabu* ist und die Suche neue Bereiche des Lösungsraumes erkunden muss. Tabu-Search verfügt damit über eine einfache Methode, um aus lokalen Optima zu entkommen.

Das eigentlich wichtige in Algorithmus 2 passiert in Zeile 4: Hier wird die jeweils beste Lösung aus allen Nachbarlösungen gewählt, es sei denn sie ist *tabu*. Die Tabuliste enthält alle vollständigen Lösungen des bisherigen Suchverlaufs. Das bedarf nach vielen Durchläufen erheblichen Speicherplatz und erfordert beim Vergleichen viel Rechenkapazität.

Statt vollständiger Lösungen können aber auch nur die Züge, die zu ihnen geführt haben, *tabu* gesetzt werden. Falls

Algorithmus 2: Very Simple Tabu-Search

- (1) $s \leftarrow$ Startlösung aus X
- (2) $s^* \leftarrow s$
- (3) **while** Abbruchbedingung nicht erfüllt
- (4) $s \leftarrow \max(s' \in N(s) \setminus T_1)$
- (5) **if** $f(s) > f(s^*)$
- (6) $s^* \leftarrow s$
- (7) $T_1 \leftarrow T_1 \cup \{s\}$

jedoch nur ein Zug tabu ist, dann verbietet das alle Lösungen, die durch diesen Zug erreicht werden könnten. Um diese Problematik zu umgehen, aber trotzdem Zyklen und lokale Optima zu verbieten, kann die Länge der Tabuliste beschränkt werden. Das bedeutet, dass nur die letzten k Züge gespeichert werden. Dies verringert den Speicher- sowie den Rechenbedarf des Algorithmus, führt aber auch dazu, dass nur Zyklen bis zur Länge k verhindert werden können.

Algorithmus 3: Simple Tabu-Search

- (1) $s \leftarrow$ Startlösung aus X
- (2) $s^* \leftarrow s$
- (3) **while** Abbruchbedingung nicht erfüllt
- (4) $m \leftarrow m \in M(s) \setminus T_1 : \max(s' \xrightarrow{m} s)$
- (5) $s \leftarrow s'$
- (6) **if** $f(s) > f(s^*)$
- (7) $s^* \leftarrow s$
- (8) $T_1 \leftarrow T_1 \cup \{m\}$

Für den Fall, dass eine Lösung zwar tabu, aber trotzdem sinnvoll ist, weil sie z.B. besser ist als s^* , gibt es außerdem so genannte Aspirationskriterien. Der Umgang damit ist relativ einfach: Wenn ein Zug die Aspirationskriterien erfüllt, dann ist er *doch nicht* tabu und kann normal verwendet werden. Hierauf wird in 2.3.4 genauer eingegangen.

2.2 Die Speicher

Glover und Laguna [4][6] beschrieben Recency, Frequency, Quality und Influence als im Speicher zu speichernde Parameter des Suchverlaufs.

Recency ist das **Kurzzeitgedächtnis**, das schon im vorherigen Punkt als Tabuliste beschrieben wurde.

Frequency ist eher ein **Langzeitgedächtnis**. Dort wird gespeichert, wie oft welche Züge bisher ausgeführt wurden.

Quality verweist auf die **Qualität** der bisherigen Lösungen. Es können besonders gute Lösungen gespeichert werden, um später an ihnen weiter zu rechnen oder sie untereinander zu kombinieren.

Influence bedeutet, dass die individuellen **Verbesserungen** der jeweiligen Züge gespeichert werden.

Wenn Tabu-Search ernsthaft auf ein Problem angewandt werden soll, dann sollten alle diese Punkte beachtet werden, denn der Umgang mit den während des Suchverlaufs gesammelten Daten ist vom jeweiligen Problem abhängig.

2.3 Modellierung

Tabu-Search ist sehr flexibel und lässt entsprechend viel Spielraum für individuelle Problemmodellierungen. Es müssen insbesondere die Bewertungsfunktion, die Nachbarschaftsbeziehungen, die Aspirationskriterien sowie die Spe-

icherparameter entworfen werden. Die Begriffe Streuung (Diversification) und Intensivierung (Intensification) sind dabei von großer Bedeutung.

2.3.1 Streuung und Intensivierung

Streuung und Intensivierung sind zwei unterschiedliche Strategien.

Bei einer Streuungsstrategie wird versucht, einen groben Eindruck vom gesamten Suchraum zu bekommen. Aus diesem Grund werden möglichst unterschiedliche Lösungen generiert.

Bei einer Intensivierungsstrategie wird genau das Gegenteil versucht, nämlich das intensive Verbessern von vielversprechenden Lösungen.

Beide Strategien spielen eine große Rolle im Suchprozess und arbeiten zusammen. Während einer gut modellierten Suche sollten sich Streuung und Intensivierung optimal abwechseln, damit lokale Optima überwunden und das globale Optimum optimal ausgeschöpft wird.

2.3.2 Bewertungsfunktion

Die Bewertungsfunktion einer Lösung bestimmt die Güte der Lösungen. Es wurde bereits beim Hillclimbing erwähnt, dass lokale Optima für iterative Suchen mehr oder weniger große Probleme darstellen. Ein weiteres Problem ist auch noch die Existenz von Plateaus – also großen Bereichen im Lösungsraum, in denen keine direkte Verbesserung der Lösung möglich ist. Das Design der Bewertungsfunktion sollte darum möglichst so ausfallen, dass es so wenig Plateaus und lokale Optima gibt, wie möglich. Außerdem kann es sinnvoll sein die Bewertungsfunktion während der Suche systematisch anzupassen, um stärker zu streuen oder die Suche zu intensivieren.

2.3.3 Nachbarschaftsbeziehungen

Die Nachbarschaftsbeziehungen legen fest in welche Richtungen eine Suche sich fortbewegen kann. Das wichtigste Kriterium an die Nachbarschaftsdefinition ist, dass jede Lösung auch wirklich aus der Nachbarschaft erreicht werden kann. So trivial diese Feststellung auch klingen mag – es gibt Probleme bei denen die Definition einer geeigneten Nachbarschaftsbeziehung nicht so einfach ist. Bei einem Problem sollte die Nachbarschaft so spezifisch wie möglich sein. Das bedeutet, dass bei einer total zufälligen Fehleroberfläche wahrscheinlich keine Methode besser funktioniert, als eine zufällige Nachbarschaft. Aber wenn es aber um andere Probleme geht, sollten die Nachbarschaftsbeziehung einen *Sinn* haben.

2.3.4 Aspirationskriterien

Die Aspirationskriterien können sehr einfach sein und einfach nur Züge durchlassen, die die aktuell beste Lösung verbessern. Es sind aber auch Kriterien möglich, die andere Eigenschaften der Züge einbeziehen, beispielsweise die Existenz einer besonders guten Teillösung.

2.3.5 Speicherparameter

Die Speicherparameter spielen eine besonders große Rolle beim Charakter der Suchstrategie.

Das Kurzzeitgedächtnis verhindert hauptsächlich Zyklen und in gewissem Maße lokale Optima. Je kürzer das Kurzzeitgedächtnis ist, desto wahrscheinlicher wird die Suche in einem kleinen Bereich stecken bleiben und ist damit eher

Intensivierend. Wenn das Kurzzeitgedächtnis jedoch größer ist, dann hat es einen gewissen Einfluss auf die Streuung, da der Bereich der Suche bald komplett tabu wird. Es ist also fast immer sinnvoll, die Größe der Tabuliste während der Suche zu verändern, was auch vielfach experimentell bestätigt wurde [2].

Das Langzeitgedächtnis spielt tendenziell eher bei der Streuung eine Rolle. Eine Möglichkeit das Langzeitgedächtnis in die Suche mit einzubeziehen, ist die folgende: Wenn eine Lösung durch einen Zug m_s zustande kommt, so wird die Anzahl der bisherigen Aufrufe von m_s von der Bewertungsfunktion abgezogen. Dadurch wird der Wert einer Lösung kleiner (schlechter), je öfter der sie bedingende Zug bisher ausgewählt wurde. Das führt dann dazu, dass die Züge gleichmäßiger genutzt werden und weniger nach ihrer ursprünglichen Bewertung. Eine andere Strategie besteht darin, den am seltesten getanen Zug zu wählen, falls es in der Nachbarschaft keine sinnvollen Alternativen gibt.

Der Qualityspeicher speichert Lösungen, die besonders gut waren. Dies wird meistens durch einen maximalen Abstand von der bisher besten Lösung realisiert. Jede Lösung die in diesen Bereich gehört, wird gespeichert. In der Intensivierungsphase ist es die Aufgabe, diese Lösungen so weit wie möglich zu verbessern. Während der Streuung können solche Lösungen aus verschiedenen Bereichen des Suchraumes gesammelt werden.

Auch Intensifikation speichert besondere Lösungen, und zwar diejenigen, die besonders viel besser waren als ihre Vorgängertlösungen. Diese Lösungen können als Meilensteine verstanden werden und später dazu dienen, von einem vorherigen Zustand aus in eine andere Richtung weiter zu suchen.

3. AUSBLICK & FAZIT

Tabu-Search ist von seiner grundsätzlichen Idee ein sehr flexibler Ansatz, der sowohl simpel als auch sehr kompliziert und spezifisch implementiert werden kann. Das Verfahren eignet sich damit für viele Bereiche und ist einer der aussichtsreichsten Kandidaten, wenn Lösungen für Optimierungsprobleme gesucht werden, für die es keine effizienten Lösungsverfahren gibt. Mit der genannten Flexibilität kommt aber auch eine Vielzahl von Parametern, die es optimal einzustellen gilt.

Der grundlegende Unterschied von TabuSearch zu vielen anderen Methoden ist aber, dass die Entscheidungen während der Suche deterministisch sind und dem menschlichen, systematischen Denken damit eher entsprechen. Simulated Annealing und Genetische Algorithmen basieren hingegen auf stochastischen Konzepten und sind dadurch teilweise schwer an deterministische Probleme anzupassen.

4. REFERENZEN

- [1] E. T. Alain Hertz and D. de Werra. A tutorial on tabu search. Technical report, Département de Mathématique, MA-Ecublens, 1997.
- [2] R. Battiti and G. Tecchiolli. The reactive tabu search. In *ORSA Journal on Computing* 2, pages 126–140, 1994.
- [3] F. Glover. Tabu search methods in artificial intelligence and operations research. In *ORSA Artificial Intelligence*, volume 1. Kluwer Academic Publishers, 1987.
- [4] F. Glover. Tabu search and adaptive memory programming - advances, applications and challenges. In H. Barr and Kennington, editors, *Interfaces in Computer Science and Operations Research*, pages 1–75, Los Alamitos, CA, 1996. Kluwer Academic Publishers.
- [5] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Boston, 1997.
- [6] F. W. Glover and M. Laguna. Tabu search. In P. M. Pardalos and D. Du, editors, *Handbook of Combinatorial Optimization*, volume 3, pages 621–757. Kluwer Academic Publishers, 1998.
- [7] P. Graham and B. Nelson. Genetic algorithms in software and in hardware - A performance analysis of workstations and custom computing machine implementations. In K. L. Pocek and J. Arnold, editors, *IEEE Symposium on FPGAs for Custom Computing Machines*, pages 216–225, Los Alamitos, CA, 1996. IEEE Computer Society Press.
- [8] S. Hurley, S. U. Thiel, and D. H. Smith. A comparison of local search algorithms for radio link frequency assignment problems. In *SAC '96: Proceedings of the 1996 ACM symposium on Applied Computing*, pages 251–257, New York, NY, USA, 1996. ACM Press.
- [9] P. J. M. van Laarhoven and E. H. L. Aarts. *Simulated Annealing: Theory and Applications*. Kluwer Academic Publishers, Norwell, MA, USA, 1987.