Technische Universität München
Institut für Informatik
Prof. Dr. Angelika Steger

SS 1999
Merkblatt 6
1. Juni 1999

Praktikum Diskrete Optimierung

Tabu-Search

1 Beispiel

Anhand eines Beispiels soll dargestellt werden, wie *recency* basierte Tabu-Search bei dem Netzplanungsproblem aussehen könnte. Als Attribut merken wir uns, welcher Knoten wann in einen Pfad eingefügt wurde. Eine bestimmte Zeitspanne nach dem Einfügen, der *tenure*, ist es verboten diesen Knoten nochmals einzufügen.
Auf der folgenden Seite finden Sie eine Skizze der Tabu-Search in Pseudocode.
Eine weitere Möglichkeit wäre, die zwischen zwei Knoten erlaubten Pfade einzuschränken und sich für jeden Pfad zu merken, wann dieser eingefügt wurde.

```
recentNodes := array(n × n × n);      // Start-, End-, Zwischenknoten
step := 0;

func deletedNode(newPath, oldPath)
    return true, wenn newPath aus oldPath durch Löschen eines Knotens ent-
    steht;

func newNode(newPath, oldPath)
    assert not deletedNode(newPath, oldPath);
    return den eingefügten bzw. verschobenen Knoten, um den sich newPath
    und oldPath unterscheiden;

func isAllowed(newPath, oldPath) : bool
    if deletedNode(newPath, oldPath) then return true;
    v := newNode(newPath, oldPath);
    if recentNodes[source(newPath), target(newPath), v] < step−tenure then
        return true;
    else
        return false;
    fi

while not finished do
    best := ∞;
    forall_paths(path) do
        forall_neighbors(neigh, path) do
            if isAllowed(neigh, path) and deltaCost(neigh, path) < best then
                best := deltaCost(neigh, path);
                bestNeigh := neigh;
                oldPath := path;
            fi
        od
    od
    insertPath(bestNeigh, oldPath);
    if not deletedNode(bestNeigh, oldPath) then
        s := source(bestNeigh);
        t := target(bestNeigh);
        recentNodes[s, t, newNode(bestNeigh, oldPath)] := step;
    fi
    step := step + 1;
od
```