Dokumentation der Simulated Annealing Implementierung

18. Juli 2025

Dokumentation der main.c

Die Datei `main.c` enthält die vollständige Implementierung der Metaheuristik Simulated Annealing zur Optimierung der Reihenfolge von Fräsarbeitsgängen. Im Folgenden werden alle Funktionen und deren Zweck kurz beschrieben.

`graphml_to_adjacency_matrix` Liest eine GraphML-Datei ein, zählt die Anzahl der Knoten und Kanten, speichert die Knotennamen via name_to_idx und füllt die Adjazenzmatrix mit den Kantengewichten. Initialisiert außerdem die globale Knotengewichtsumme um sie nicht ständig neu zu berechnen, da dieser Wert für die Kostenberechnung konstant ist.

`name_to_idx` Wandelt einen Knotennamen (`char*`) in den zugehörigen Index im Knotenarray um. Wird für die Zuordnung von Kantennamen und internen Kantenindexen verwendet.

`calculate_cost` Berechnet die Kosten eines Pfades, indem die Kantengewichte aufsummiert werden. Ungültige Übergänge werden mit einer konstanten hohen Strafe versehen (dieser Ansatz versagt jedoch wenn die kantengewichte zu hoch werden, in diesem fall müsste dieser penalty überarbeitet werden). Die Knotengewichtssumme werden zu den Gesamtkosten hinzugefügt. Der Letzte übergang (letzter Knoten zu erstem) wird nicht mitberechnet, da lediglich ein Pfad gesucht wird.

`random_swap` Vertauscht zufällig zwei Elemente im Pfad. Dient der Erzeugung neuer Nachbarschaftslösungen im Simulated Annealing in dem Löcher im Pfad generiert werden, welche durch maximise_path 'behoben' werden.

`maximise_path` Versucht, einen Pfad möglichst gültig zu machen, indem an jedem ungültigen Übergang (Kante mit Gewicht 0) ein Knoten eingefügt wird, der ein gültiges Ziel wäre und selbst im aktuellen Pfad Ziel einer ungültigen Verbindung ist. Dies verbessert die Pfadqualität und reduziert die Kosten. Dieser Ansatz lässt intakte Pfadabschnitte unberührt, verliert jedoch signifikant an Performance bei schwach vernetzten Graphen.

`get_valid_transition_count` Gibt die Anzahl gültiger Übergänge (Kanten mit Gewicht ungleich Null) im Pfad zurück. Wird zur Bewertung der Pfadqualität genutzt. Hier werden alle Übergänge, inklusieve des Übergangs vom letzten Knoten zum ersten, gezählt.

`simulated_annealing` Implementiert den Simulated Annealing Algorithmus. Mutiert den Pfad iterativ durch zufällige Vertauschungen und akzeptiert neue Lösungen nach dem Metropolis-Kriterium. Gibt regelmäßig Zwischenstände und eine ETA aus.

`generate_random_path` Erzeugt eine zufällige Permutation der Knoten als Startpfad für die Optimierung.

`write_best_path_txt` Schreibt den besten gefundenen Pfad als Semikolon-separierte Liste in eine Textdatei. Die Datei wird unter `bestpaths/` abgelegt.

`main` Hauptfunktion. Liest die Eingabedatei, initialisiert die Datenstrukturen, gibt die Adjazenzmatrix aus, führt mehrere Simulated Annealing Durchläufe aus um tatsächlich gebrauch von den erhöhten Temperaturen zu machen und verschiedene Minima zu entdecken, wählt den besten Pfad dieser Läufe und gibt diesen sowohl auf der Konsole als auch in einer Textdatei aus. Am Ende werden alle Ressourcen freigegeben.

Nutzungsanleitung

Das Programm kann wie folgt kompiliert und ausgeführt werden:

```
gcc -o3 -o main main.c
./main path/to/graphfile.graphml
```

Eingabeparameter: Der Pfad zur GraphML-Datei wird als einziger Kommandozeilenparameter übergeben.

Dateien: Der optimierte Pfad wird in einer Textdatei im Verzeichnis `bestpaths/` gespeichert.