Montag, 11.07.2022, 15.00 - 17.00 Uhr:  
Präsentation der Gruppen Multiple Knapsack Problem (MKP) und Quadratic Assignment Problem (QAP)

1. Ich erkläre zuerst Das **QAP Problem**

Unter QAP Problem sollen wir insgesamt n Anlagen auf n identische Stationen verteilen, so dass die Gesamttransportleistung zwischen verschiedenen Anlagen minimiert wird. Die Gesamttransportleistung ist gleich der Summe aus der Entfernung multipliziert mit der Transportmenge.

Im Folgenden werde ich das QAP-Problem anhand einer konkreten Instanz ausführlich erläutern.

Hier ist ein Json-Datei mit 4 key value pairs, nämlich

* Size, die Anzahl der Anlagen sowie Stationen,
* A, die Transportmatrix zeigt die Transportmengen zwischen Anlage i und j,
* D, die Entfernungsmatrix zwischen Station k und l,
* M, die eingehaltene Mindestabstände zwischen Anlagen i und j

Die Lösung von QAP-Problem wird in zwei Fällen geteilt:

Im ersten Fall wird die m-Matrix in Ruhe gelassen, was ein relativ einfacher Fall ist, in dem wir nur die A- Matrix und D-Matrix verwenden.

Im anderen Fall muss jedoch die M-Matrix berücksichtigt werden. In diesem Fall müssen wir feststellen, ob der Abstand zwischen den Anlagen grüßergleich der von M-Matrix geforderten Mindestabstand ist. Wenn diese Bedingung erfüllt ist, dann haben wir eine zulässige Lösung. Eine unzulässige Lösung muss weggelassen werden.

Wie ich gerade erwähnt habe, besteht das QAP-Problem darin, n Anlagen an n Stationen zu platzieren. Aber in Verbindung mit den 3 Matrizen in diesem Fall, wenn ich die Anlagen (i, j) permutiere, muss ich die Reihenfolge der beiden Matrizen (A, M) anpassen. Aber ich würde nur eine Matrix verändern. Deswegen begann ich also mit der Absicht, die D-Matrize direkt zu verändern, d. h. **ich ordnete die Standorte den Anlagen zu**. Ich habe den Prozess in der class **Solution** realisiert.

Wenn ich die Permutationen der Musterlösung in meine EL eingebe, habe ich dieselben Ergebnisse wie Musterlösung bekommen. D.h. meine Logik ist richtig.

1. ROS

Um eine Startlösung zu generieren habe ich die Heuristik ROS gewählt. Das heißt, ich filtere die Lösung mit dem kleinsten Gesamttransport durch mehrere zufällig generierte Permutationen. ROS wird auch als konstruktive Baustein in meinem Solver beinhaltet.

Das ROS-Verfahren ist in der Lage, sehr schnell eine optimierte Startlösung zu finden und eignet sich besonders für Situationen, in denen keine Mindestabstände berücksichtigt werden müssen. Das ist der Grund, warum ich mich für ROS entschieden habe.

ROS hat jedoch auch die Einschränkung, dass es keine zulässige Startlösung mit Berücksichtigung von Mindestabstand von 25 Anlagen generieren kann. Der Grund dafür ist, dass bei 25 anzuordnenden Standorten insgesamt 25 Fakultät Permutationen möglich sind. Dies ist eine sehr große Zahl, aber aus Gründen der Laufzeit habe ich die Anzahl der zufälligen Permutationen in ROS nicht erhöht.

1. ILS Heuristik

Meine Metaheuristik ist Iterated Local Search. Der Prozess der Intensivierung beinhaltet eine einzige lokale Suche in einer Nachbarschaft. Der Prozess der Diversifikation beinhaltet das Hinzufügen eine Perturbation zur aktuellen Permutation. Danach können wir eine neue Permutation erhalten, und eine lokale Suche in der Nachbarschaft dieser Permutation durchführen, um festzustellen, ob eine Lösung mit einem niedrigeren Gesamttransport gefunden werden kann.

Eine Voraussetzung von ILS ist Nachbarschaft zu erstellen. Ich habe versucht, Nachbarschaften durch zwei Methoden zu erstellen, nämlich Swapmove und Twoedgeexchange. Aber als ich beide Methoden ausführte, stellte ich fest, dass Swapmove immer eine bessere Lösung lieferte. Daher denke ich, dass Swapmove eher für die 4 von mir behandelten Instanzen geeignet ist.

In der class IteratedLocalSearch habe ich eine Methode Pertubation definiert, die dient dazu, dass ich zwei zufällige Elemente i und j aus der initialen Permutation herausziehe. Und dann tausche ich die Elemente der aktuellen Permutation mit Index i und Index j aus. Außerdem habe ich auch eine Methode P\_3 definiert, dabei habe ich deri Elemente ausgetauscht. Aber diese Methode kann immer eine schlechtere Lösung liefern. Der einzige Vorteil von P\_3 ist es, dass sie weniger Laufzeit braucht.

1. Ergebnisse bewerten und Verbesserungsidee

Zum Ende möchte ich die Ergebnisse meines Programms zeigen und auswerten, und auch einige Verbesserungsideen stellen.

Zum erstens sehen Sie hier die Ergebnisse ohne Berücksichtigung von Mindestabstand:

Die Spalte FCFS zeigt die jeweiligen Gesamttransporte ohne Verbesserung. Die Spalte ROS zeigt die Gesamttransporte der StartLösungen, während die Spalte ILS die Gesamttransporte nach dem ganzen Verbesserungsverfahren zeigt. Die Spalte OPT ist die Musterlösung. Es ist leicht zu erkennen, dass scr12 nach der Optimierung die tatsächlich optimale Lösung erreicht und tai25b nur um 0,3 % von der optimalen Lösung abweicht. Die übrigen beiden Instanzen weichen um etwa 3,7 % von der optimalen Lösung ab. Ich denke, das ist akzeptabel.

Zum Zweitens die Situation mit Berücksichtigung von Mindestabstand:

Wie aus dieser Tabelle ersichtlich ist, liegen die Startlösungen bei kleinen Datenmengen (12 und 15) sehr nahe an der durch ILS optimierten Lösungen. Wenn die Datenmenge jedoch größer ist, wird der Optimierungsgrad des metaheuristischen Algorithmus erheblich steigert.

Aufgrund der durch den Lehrstuhl begrenzten Rechenzeit konnte ich die Vorteile des ROS-Algorithmus nicht voll ausschöpfen, insbesondere bei der Anordnung von 25 und mehr Anlagen. Aber in der realen Produktion hat man genug Zeit, um mit Hilfe von ROS eine bessere StartLösung zu finden. Und dann diese Startlösung durch ILS zu verbessern

Das war s, vielen Dank für Ihre Aufmerksamkeit!