Nebenläufige Programmierung

1. Der Thread und der Knoten

Unsere Implementierung wird in Java erfolgen. Jede Spalte bekommt, wie vorausgesetzt, einen thread. Als erstes wird nur der Thread mit dem Anfangsknoten betrachtet. Dieser wird, wie alle anderen Knoten dieser Spalte, vom Thread in einer LinkedList gehalten. Ein Knoten ist hierbei eine Datenstruktur, die einen y-Wert (im Falle, dass es mehrere Nullknoten innerhalb einer Spalte gibt dient dieser zur Identifikation), jeweils eine Referenz auf den Vorgänger- und Nachfolgerknoten, vier Übergangsraten und einen Wert hat. In jedem Durchgang arbeitet nun der Thread jeden seiner Knoten ab.

2. Der Reihenaustausch

Abarbeiten ist dabei so zu verstehen: Am aktuellen Knoten werden zwei Update-Funktionen aufgerufen, die einmal den Flow zum Vorgänger und das andere Mal den Flow zum Nachfolger berechnen und propagieren. Wird der Wert eines Knoten dadurch erstmals größer Null, so wird er erstellt und als Rückgabewert der jeweiligen Update-Funktion an den Thread geliefert. Dieser ordnet ihm seine Übergangsraten zu und reiht ihn in die Liste seiner Knoten ein. So wird die Liste an Knoten bei jeder Iteration durchgegangen.

3. Der Spaltenaustausch

Nach einer durch einen Parameter bestimmbaren Anzahl an Schritten soll eine Spaltenpropagierung stattfinden. Dazu hat jeder Thread, also jede Spalte, einen Exchanger zu beiden Seiten. Vom Thread wird eine TDoubleArrayList erstellt, die so groß ist, wie es Reihen gibt. Der Thread berechnet nun den Outflow zu den jeweiligen linken und rechten Spalten per Iteration über die Knoten. Dem Exchanger für die jeweilige Spalte wird jetzt diese Liste gegeben. Da der Exchanger erst eine Rückgabe liefert, sobald der andere Thread auch eine Liste übergeben hat, wird sichergestellt, dass beide Threads die gleiche Anzahl an Iterationen haben sobald die Synchronisation erfolgt. Dabei erfolgt erst der Austausch nach links und dann der Austausch nach rechts, um Deadlocks zu vermeiden. Ist der Austausch zu beiden Seiten erfolgt werden die so erhaltenen Werte verrechnet und etwaige neue Knoten angelegt.

4. Die Konvergenz

Bei jedem Spaltenaustausch wird die Summe an Inflow mit der Summe an Outflow auf approximative(Abstand < Epsilon) Gleichheit verglichen. Nun gibt es globale booleans, die die mögliche Konvergenz eines jedes Threads speichern. Eine synchronisierte Methode verändert diese gegebenenfalls nach jedem Exchange. Ist nach einem kompletten Spaltenaustausch in mindestens einem Thread Konvergenz vermutet, so wird die globale Iterationsanzahl halbiert. Ist jedoch keiner der Threads konvergent, so wird die Iterationszahl verdoppelt, sollte sie kleiner als die Originalanzahl sein. Ein Thread holt sich die Iterationsanzahl vor dem nächsten Exchange erneut (die Halbierung erfolgt also einen Zyklus nach der Konvergenzfeststellung). Fällt die Iterationszahl auf ein wird auf Konvergenz innerhalb von Spalten geprüft und terminiert, falls auch alle Spalten in sich konvergieren.