1 Problemstellung

1.1 Ziele

- alle patienten sollen mit den benötigten medikamenten versorgt werden
- die meisten fahrer sollen wenn sie fahren eine maximale auslastung haben (i.e., wenig einzelfahrten)
- minimierung der anzahl der fahrer um exposition freiwilliger zu reduzieren

1.2 Out of Scope

- apotheken wollen möglicherweise nicht unbedingt ihr lager leer verkaufen
- es wird kein perfektes optimum angestrebt, dies würde das TSP lösen was NP vollständig wäre
- entscheidung sollte ein medikament nicht außreichend vorhanden sein
- minimale anzahl gleichzeitiger fahrten
- keine insg. kürzeste gefahrene distanz

1.3 Mögliche Verbesserungen

- fahrdistanz kann limitiert werden
- ladung kann limitiert werden

2 Heuristic

3 Input

$D = \{set\ of\ driver\}$
$d_d : D \rightarrow \mathbb{N}\ (max\ distance\ of\ driver)$
$A = \{set\ of\ pharmacies\}$
$P = \{set\ of\ patients\}$
$M = \{set\ of\ meds\}$
$d_{DA} : DxA \rightarrow \mathbb{N}\ (distance\ between\ driver\ and\ pharmacy)$
$T_{DA} = \{(d, a) \in D \times A \mid d_{DA}((d, a)) \leq d_d(d)\}$
$T_{DP} = \{(d, p) \in D \times P \mid d_{DP}((d, p)) \leq d_d(d)\}$
$N = \{set\ of\ needed\ meds\ by\ patients\} \subset PxM$
$S = \{set\ of\ stored\ meds\ by\ pharmacies\} \subset AxM$
$PD = \{(d, a, p, m) \in D \times A \times P \times P \times M|$
$(d, a) \in T_{DA} \wedge (d, p) \in T_{DP} \wedge (a, m) \in S \wedge (p, m) \in N\}$

4 Algorithmus

Zuerst müssen wir berechnen welche schritte ein fahrer abarbeiten soll, bevor wir die optimale route konzipieren

```
sort PD by driver, pharmacy, p, m
drives = {}
do until PD is empty
    tuple ← first(PD)
    drives ← tuple
    delete in PD where element (_, _, p, m)
    reduce m in stock of p
    if stock of m in p empty
        delete in PD where element (_, _, p, _)
return drives
```

Figure 1: Dieser Algorithmus gibt uns eine heuristisch optimierte zuteilung von fahrern zu routen von apotheken zu patienten mit medikament

Der vorherige algorithmus hat uns nun ein set gegeben bei dem alle patienten bedient werden und zwar von der minimalen anzahl an fahrern. Wir betrachten nun einen algorithmus der für ein set das alle tuple beinhaltet die einem einzelnden fahrer zugeordnet sind eine optimale route berechnet.

```
input set D ⊂ PD
drive_order = {}
sort D by frequency(a)
a = first(D)[1]
driver_order ← a
rem_steps = { p in D who were served by a and all remaining a in D }
// quadratic solution of TSP
do until rem_steps is empty
    closest ← smallest_distance(drive_order[−1], rem_steps)
    drive_order ← closest
    delete from rem_steps closest
    if closest is pharmacy
        rem_steps += {p in D who were served by closest)
```

Figure 2: Dieser Algorithmus gibt uns die 'optimale route' für einen fahrer