**Rapport: Optimering av förtjänst för Lindas Lustfyllda Bud och Åkeri**

**Introduktion:**

Jag fick i uppdrag att skapa ett program som hjälper Lindas Lustfyllda Bud och Åkeri att optimera sina leveranser. Huvudsyftet var att maximera förtjänsten varje dag genom att tilldela paket till budbilar på ett effektivt sätt. Samtidigt ville vi minimera eventuella straffavgifter för paket som inte levereras i tid. Projektet omfattar:

* Tilldelning av paket till budbilar baserat på vikt och förtjänst
* Beräkning av total förtjänst och straffavgifter.
* Skapande av statistik och visualiseringar för att analysera resultaten.
* Implementering av optimeringsspårning över flera iterationer.

**Lösningsmetod**

Programmet bygger på en genetisk algoritm, där lösningar iterativt förbättras genom mekanismer som mutation, korsning och urval. Algoritmen arbetar med en population av möjliga lösningar, där varje lösning representerar en specifik tilldelning av paket till bilar.

Fitnessvärden beräknas för varje lösning för att avgöra dess kvalitet baserat på förtjänst, straffavgifter och viktbegränsningar. Algoritmen avslutas när förbättringar upphör eller när ett fördefinierat antal generationer har körts

* Det bör noteras att arbetet med den genetiska algoritmen var en utmaning.

Initialt var det svårt att förstå hur algoritmen komponenter skulle samverka för att optimera resultaten.

**Datahantering**

Data läses in från en CSV-fil som innehåller information om paketens vikt, förtjänst och deadline. För varje paket beräknas en justerad förtjänst som tar hänsyn till eventuella straffavgifter. Paketen sorteras efter förtjänst per vikt för att säkerställa att de mest värdefulla paketen prioriteras.

**Algoritmens Huvudkomponenter**

* **Initialisering av Population:** En slumpmässig uppsättning lösningar skapas.
* **Fitnessberäkning:** Varje lösning utvärderas baserat på total förtjänst och straffavgifter.
* **Mutation och Korsning:** Nya lösningar genereras för att förbättra variationen och undvika lokala optima.
* **Stoppkriterium:** Algoritmen avslutas när förbättringar avtar eller efter ett fördefinierat antal generationer.

**Visualisering och Statistik**

Programmet genererar flera typer av visualiseringar och statistik:

* Histogram visar fördelningen av vikt och förtjänst bland paketen.
* Stapeldiagram visar vikt och förtjänst per bil för att analysera lastfördelningen.
* Diagram spårar förbättringen av fitnessvärden över generationer.

**Schemaläggning av körning:**

Eftersom programmet ska köras kl.05:00 varje dag för att optimera leveranserna, läste jag på om hur man ska använda schemaläggningsverktyg som cron på macOS eller Windows Task Scheduler. Dessa verktyg göt det möjligt att automatisera körningen utan att behöva lägga till manuell tidslogik i programmet.

Ex med cron:

Öppna crontab🡪 crontab -e

Sen lägg till följande rad för att schemalägga körning av programmet varje dag kl. 05:00

🡪 05 \*\*\* python /sökväg/till/filnamn.py

**Resultat**

Programmet lyckades optimera förtjänsten genom att tilldela paket till bilar på ett sätt som respekterar viktbegränsningarna och minimerar straffavgifter. Resultaten inkluderar:

* Total förtjänst från levererade paket.
* Total straffavgift från kvarvarande paket.
* En fördelning av kvarvarande paket efter vikt och förtjänst.

Visualiseringarna tydliggör optimeringsresultaten. Stapeldiagrammen visar en jämn fördelning av vikt mellan bilarna, medan histogrammen belyser prioriteringen av paket med hög förtjänst per vikt.

**Slutsats**

Lösningen uppfyller kraven genom att effektivt hantera data, utveckla en optimeringsalgoritm och presentera resultaten på ett tydligt sätt. Algoritmen är väl lämpad för detta problem, men ytterligare förbättringar, som parallellisering och dynamiska viktningar av olika faktorer, kan förbättra prestandan ytterligare. Programmet är redo att användas för att maximera intäkterna och förbättra effektiviteten i pakettilldelningen.