




Logística Urbana para Entrega de Mercadorias

André Sousa – up202005277

Pedro Fonseca – up202008307

Vitor Cavaleiro – up202004724



Descrição do problema

Uma empresa de transporte de mercadorias pretende melhorar a sua produtividade de forma a ser mais competitiva. A empresa transporta encomendas normais e expresso, sendo que estas últimas são entregues diretamente ao cliente enquanto que as normais são atribuídas a estafetas para serem entregues. A empresa considerou três cenários:

- Otimização do número de estafetas: usar o menor número de estafetas, sem comprometer a entrega das encomendas
- Otimização do lucro da empresa: entregar as encomendas que mais lucro dão, usando os estafetas que cobram menos
- Otimização das entregas expresso: minimizar o tempo médio das entregas por dia

Cenário 1: Formalização

Dados:

- $\text{couriers}_0, \dots, \text{couriers}_n$ – vetor de objetos Courier, com todos os estafetas registrados na empresa
- $\text{normalTransports}_0, \dots, \text{normalTransports}_m$ – vetor de estruturas NormalTransport, com todas as encomendas normais que têm de ser entregues

Objetivo:

- Maximizar a carga transportada por cada estafeta

Restrições:

- O somatório dos pesos e volumes dos pacotes transportados por cada estafeta não pode ultrapassar a capacidade máxima do estafeta
- A prioridade de cada pacote deve ser respeitada

Cenário 1: Descrição de algoritmos relevantes

Inicialmente, o vetor de estafetas é ordenado por ordem decrescente de peso máximo (em caso de empate, aplica-se a ordem decrescente de volumes).

Para além disso, o vetor de pacotes é ordenado por ordem decrescente de prioridade.

Seguidamente, enquanto existirem encomendas por associar a estafetas, o programa vai seleccionando os estafetas pela ordem do vetor e para cada estafeta atribui-lhe todas as encomendas que conseguir transportar, de forma a maximizar o peso e volume total transportado.

A atribuição é feita recorrendo a um algoritmo de *backtrack*: percorre o vetor de encomendas e se o estafeta tiver possibilidade de a transportar, a encomenda é associada. Quando o estafeta não conseguir transportar mais encomendas, o algoritmo retorna o nº de encomendas que lhe foram atribuídas.

Dando prioridade aos estafetas que transportam mais carga e maximizando a carga por eles transportada, é garantido que se usa o menor número de estafetas possível.

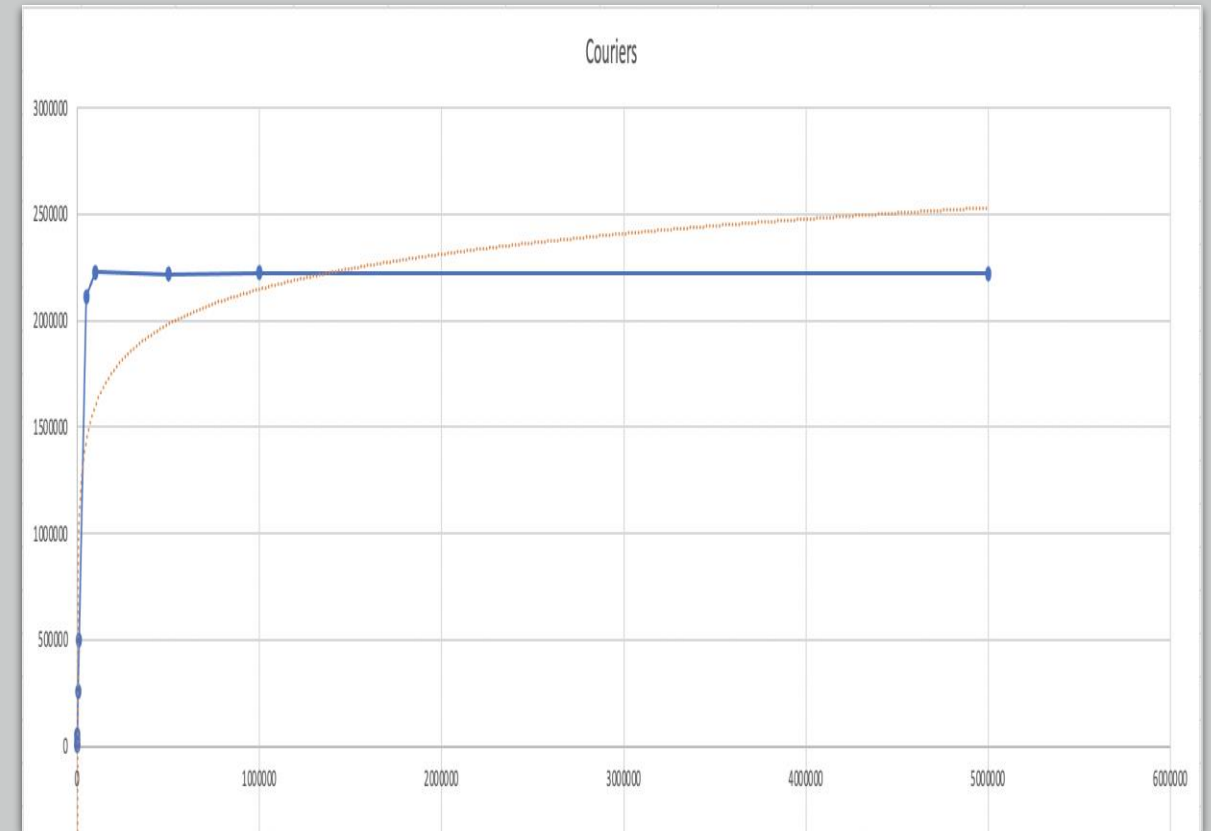
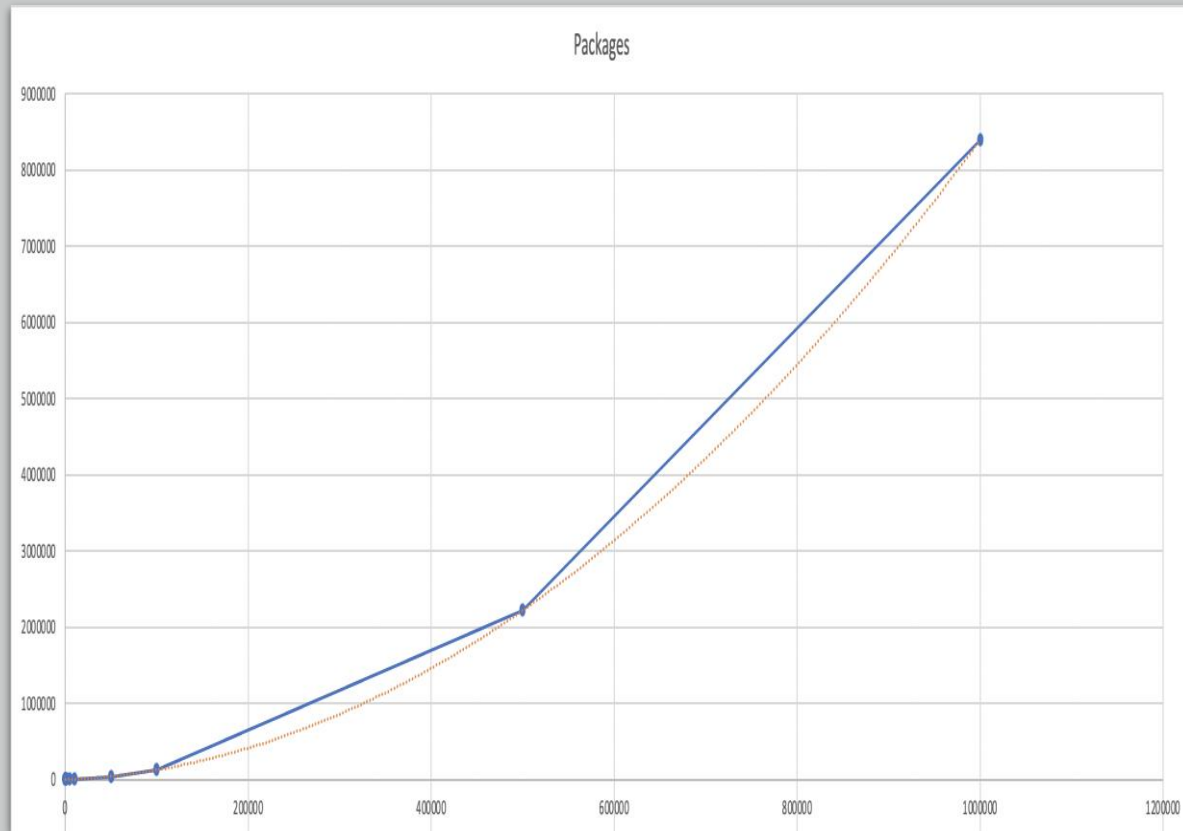


Cenário 1: Análise da complexidade

- Algoritmo de *backtrack* (*courierFiller*):
 - Complexidade temporal: n^2
 - Complexidade espacial: 1
- *optimizeNormalPackagesDistribution*:
 - Complexidade temporal: $n * \log(n) + m * \log(m) + m * \text{courierFiller}$
 - Complexidade espacial: $1 * \text{courierFiller}$
- Complexidade combinada:
 - Complexidade temporal: $n * \log(n) + m * n^2 \Rightarrow O(n, m) = m * n^2$
 - Complexidade espacial: 1

($m \Rightarrow n^\circ$ de estafetas, $n \Rightarrow n^\circ$ de encomendas normais)

Cenário 1: Avaliação empírica



Cenário 2: Formalização

Dados:

- $\text{couriers}_0, \dots, \text{couriers}_n$ – vetor de objetos Courier, com todos os estafetas registados na empresa
- $\text{normalTransports}_0, \dots, \text{normalTransports}_m$ – vetor de estruturas NormalTransport, com todas as encomendas normais que têm de ser entregues

Objetivo:

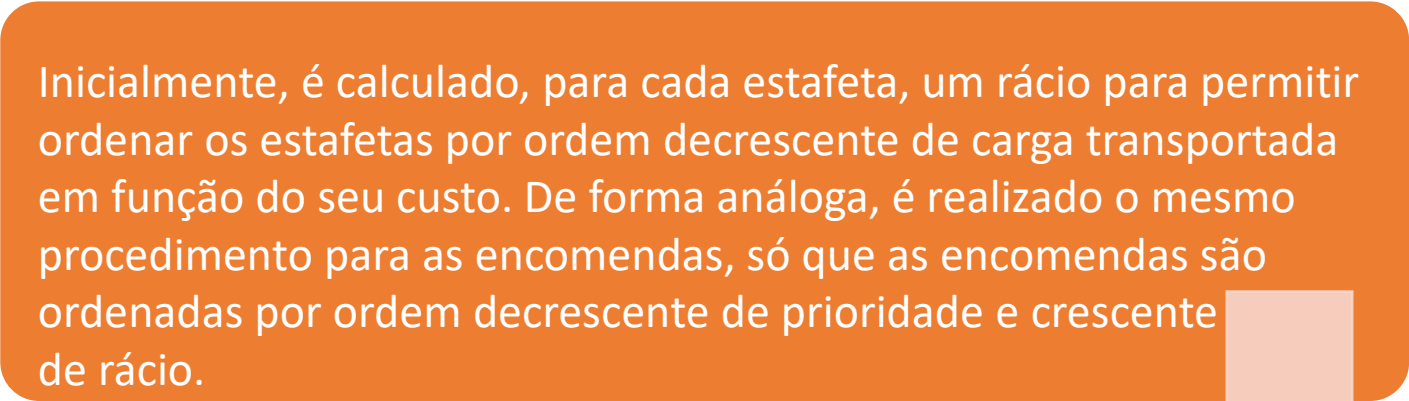
- Maximizar o lucro da empresa, reduzindo os gastos em estafetas e aumentando os ganhos provenientes da entrega de encomendas

Restrições:

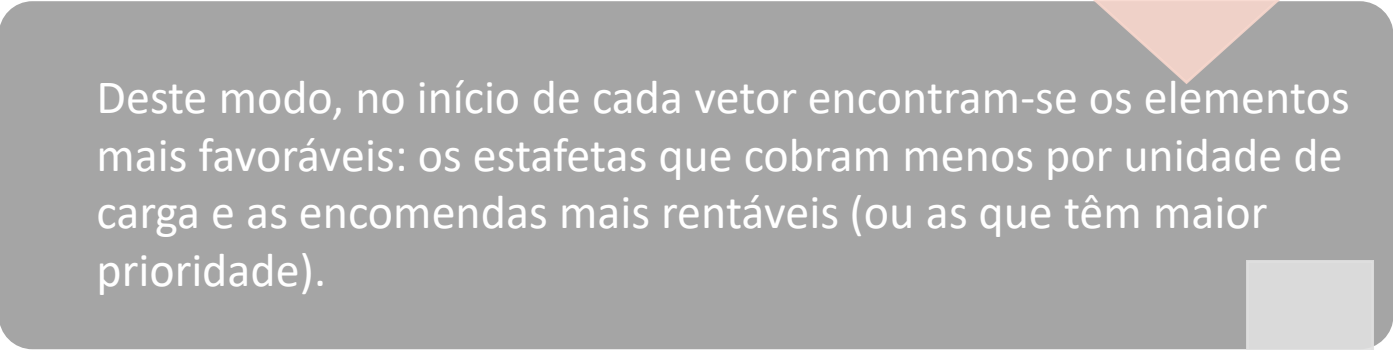
- O somatório dos pesos e volumes dos pacotes transportados por cada estafeta não pode ultrapassar a capacidade máxima do estafeta

Cenário 2: Descrição de algoritmos relevantes

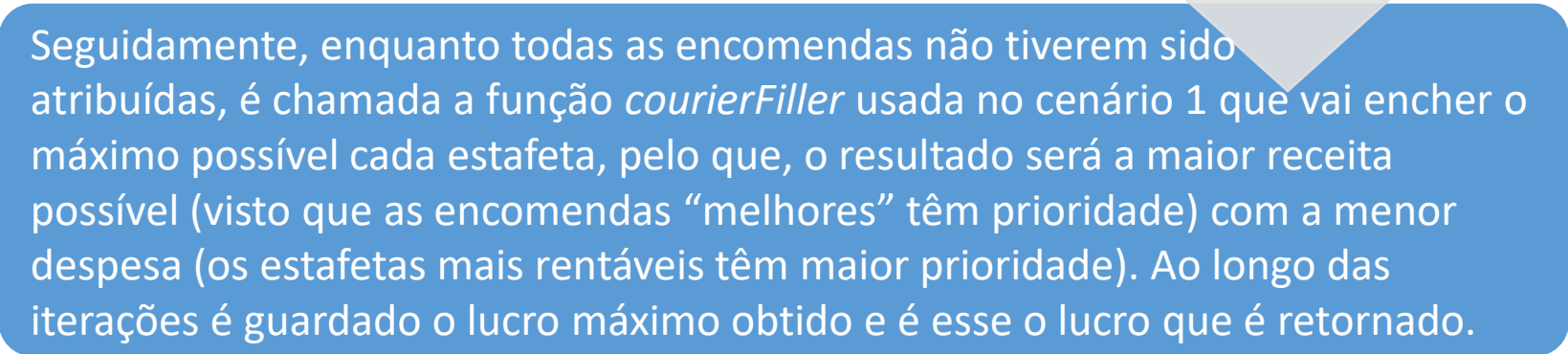
Inicialmente, é calculado, para cada estafeta, um rácio para permitir ordenar os estafetas por ordem decrescente de carga transportada em função do seu custo. De forma análoga, é realizado o mesmo procedimento para as encomendas, só que as encomendas são ordenadas por ordem decrescente de prioridade e crescente de rácio.



Deste modo, no início de cada vetor encontram-se os elementos mais favoráveis: os estafetas que cobram menos por unidade de carga e as encomendas mais rentáveis (ou as que têm maior prioridade).



Seguidamente, enquanto todas as encomendas não tiverem sido atribuídas, é chamada a função *courierFiller* usada no cenário 1 que vai encher o máximo possível cada estafeta, pelo que, o resultado será a maior receita possível (visto que as encomendas “melhores” têm prioridade) com a menor despesa (os estafetas mais rentáveis têm maior prioridade). Ao longo das iterações é guardado o lucro máximo obtido e é esse o lucro que é retornado.



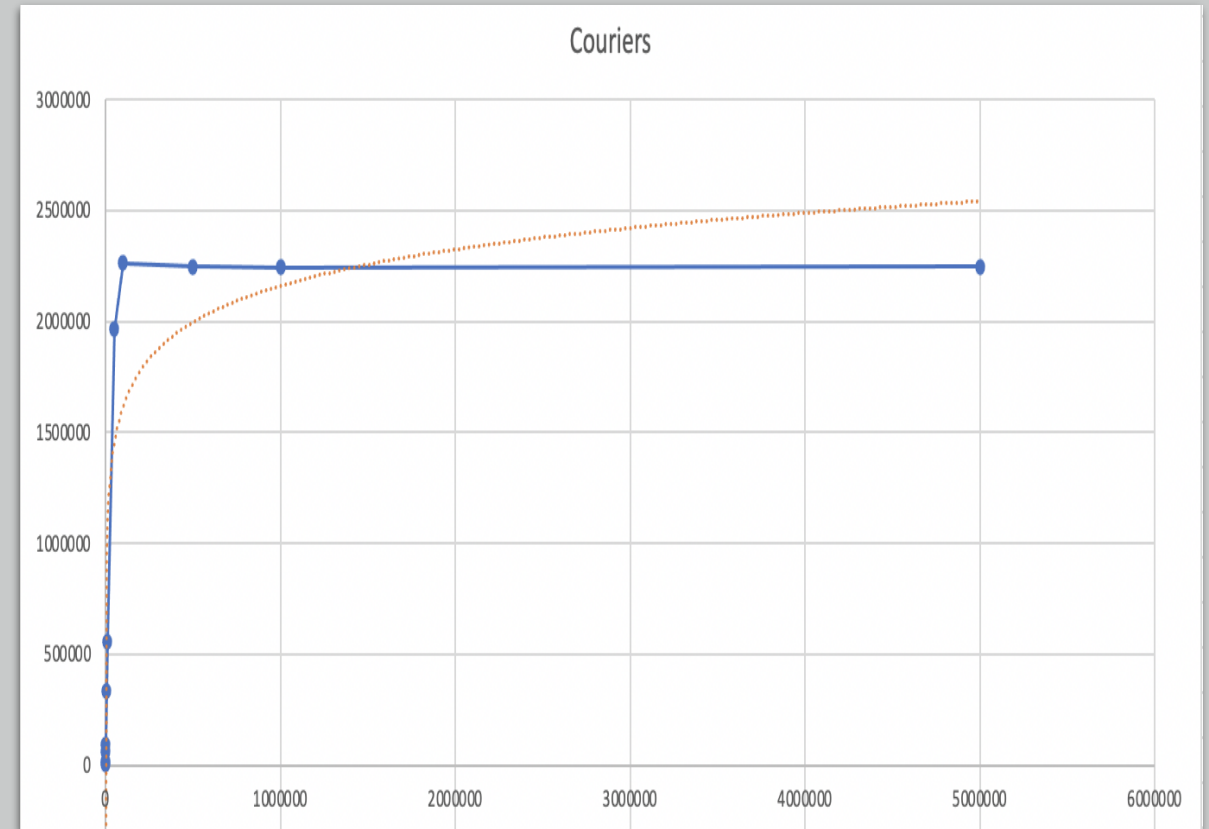
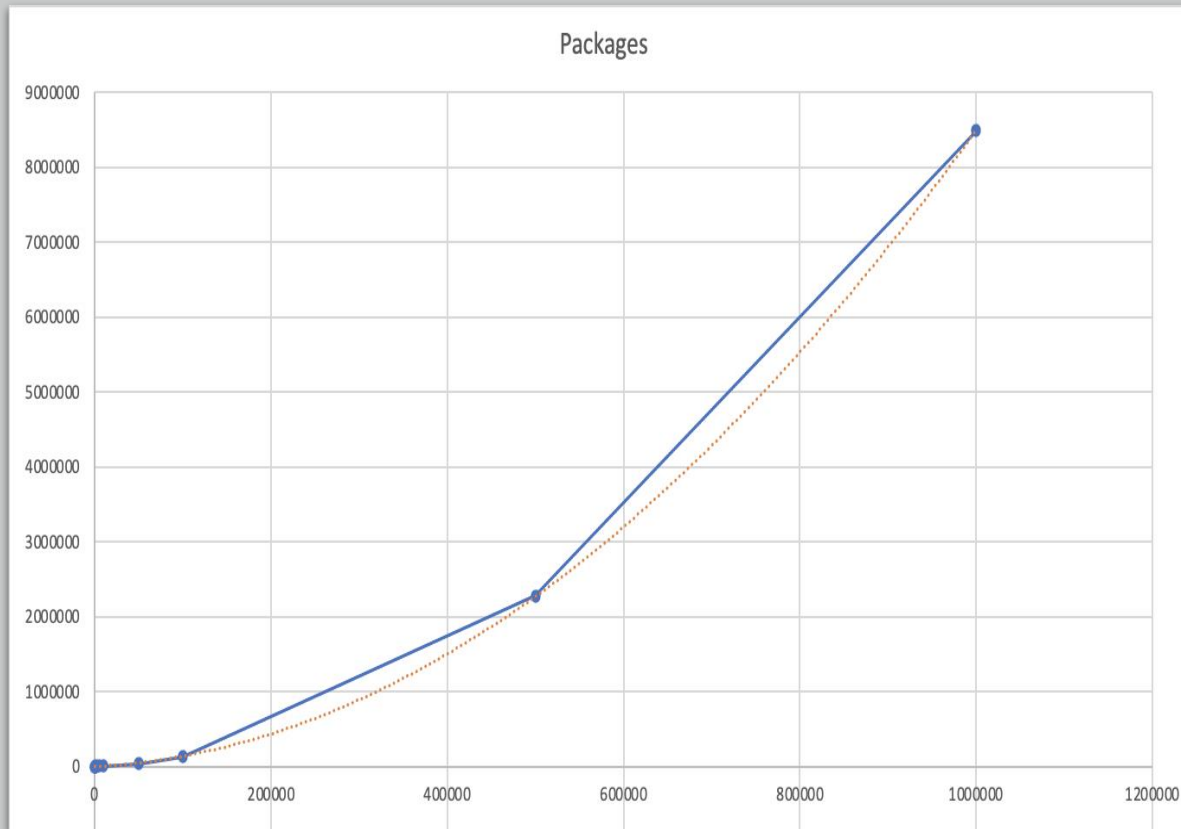


Cenário 2: Análise da complexidade

- Algoritmo de *backtrack* (*courierFiller*):
 - Complexidade temporal: n^2
 - Complexidade espacial: 1
- *optimizeProfit*:
 - Complexidade temporal: $2 * (n * \log(n)) + m * n + m * \text{courierFiller} + n$
 - Complexidade espacial: $n + \text{courierFiller}$
- Complexidade combinada:
 - Complexidade temporal: $2 * n * \log(n) + m * n^2 + n \Rightarrow O(n, m) = m * n^2$
 - Complexidade espacial: 1

($m \Rightarrow$ nº de estafetas, $n \Rightarrow$ nº de encomendas):

Cenário 2: Avaliação empírica



Cenário 3: Formalização

Dados:

- $\text{expressTransports}_0, \dots, \text{expressTransports}_m$ – vetor de estruturas `ExpressTransport`, com todas as encomendas normais que têm de ser entregues

Objetivo:


- Minimizar o tempo médio de entrega dos pedidos expresso

Restrições:

- A empresa só entrega uma encomenda por viagem
- As encomendas têm de ser entregues entre as 9:00 e as 17:00

Cenário 3: Descrição de algoritmos relevantes

Inicialmente, o vetor de pacotes expresso é ordenado por ordem crescente de duração da viagem (para ser usado um algoritmo Greedy)



Seguidamente, é calculado o tempo médio de cada entrega



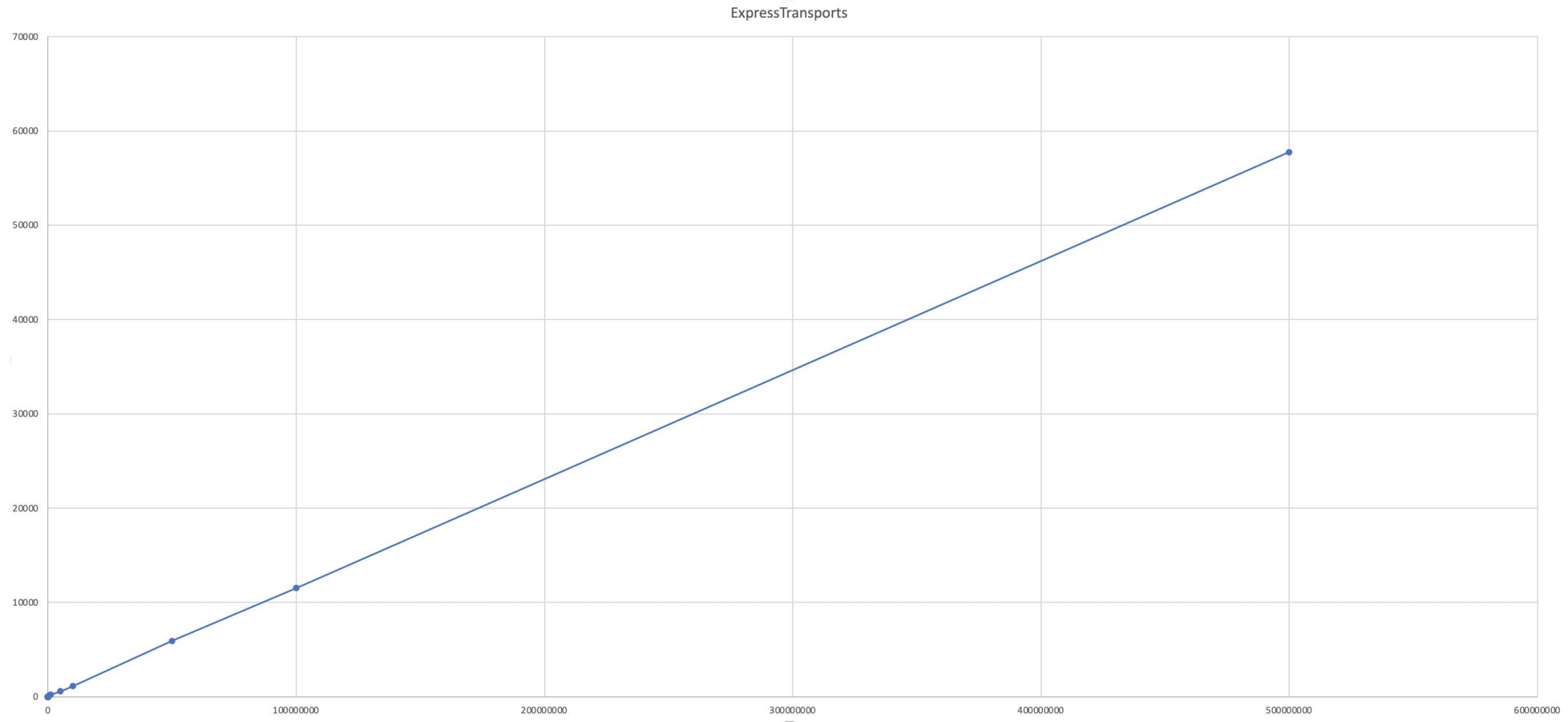
Cenário 3: Análise da complexidade

- *optimizeExpressTransports:*

- Complexidade temporal: $n * \log(n) + n \Rightarrow O(n) = n * \log(n)$
- Complexidade espacial: $O(1)$

($n \Rightarrow n^o$ de encomendas expresso)

Cenário 3: Avaliação empírica



Apesar de a complexidade temporal ser $n * \log(n)$, o resultado da análise empírica é linear.

Priorizar pedidos não entregues

A aplicação permite assinalar o final de um dia de trabalho através da função *endOfBusiness*, que remove do sistema as entregas normais já efetuadas e incrementa a prioridade das encomendas que não foram entregues nesse dia (inicialmente, todas as encomendas têm prioridade zero). Para além disso, o utilizador pode adicionar mais encomendas ao sistema, a qualquer momento (respeitando a prioridade).



Solução algorítmica a destacar: *courierFiller*

Dos três cenários que foram abordados, dois dependem fortemente do algoritmo auxiliar que faz uso da estratégia de *backtracking* para encher o máximo possível a carrinha de cada estafeta. Apesar de ser este o algoritmo que tem mais impacto na complexidade temporal dos cenários, é ele que permite a otimização do número de estafetas usados e do lucro da empresa.



Principais dificuldades e autoavaliação

Principais dificuldades:

- Implementar algoritmos eficientes para resolver os dois primeiros cenários.

Autoavaliação:

- André Sousa: 33%
- Pedro Fonseca: 33%
- Vitor Cavaleiro: 33%