Pesquisa Operacional – Relatório 1

Willian de Souza Soares 2014.1.08.034

Foi utilizada um implementação básica de ranqueamento dos itens a serem carregados.

Comecei tentando utilizar `Lucro÷Peso` do item para ranquear os items, o que obviamente não serviria: alguns itens tem um Lucro alto e peso baixo, porém tem o Volume altíssimo, o que os colocariam em uma posição pior no ranking. Alterei a fórmula para `Lucro÷Peso÷Volume`, que fazia bem mais sentido. Assim, após ordenar esta lista de itens, temos um roteiro básico de quais itens compensam mais ser carregados em um container.

Apesar de acreditar que o ranqueamento está aceitável por hora, algumas coisas ficaram para trás, como aproveitar completamente o espaço de um container. Por diversas vezes, o algoritmo não completa os container por falta de uma lógica que os permita continuar procurando itens menos valiosos que talvez completem as lacunas de um container.

Certamente, dada uma outra oportunidade, mudaria algumas coisas no projeto: o ranqueamento precisa de algum toque, principalmente quando se trata de uma instancia que pode-se fracionar os itens; o sistema de preechimento dos containers é muito falho e faltam diversas lógicas para aproveitar os espaços vazios. O mal aproveitamento dos containeres chega a levar o lucro da instância 2 em apenas `317.85`, o que em comparação com outras soluções chega a ser uma falta de `100-200` de lucro.

Em seguida, trexos relevantes da classe `Logistica`, que contém o algoritmo de organização de items e *assignment* aos containers.

```
1 reference
public List<Container> OtimizaLucro()
{
    List<(int qtdItemFrete, Item item)> itemsOrganizados = OrdenaItems().Select(x => (0, x.Item2)).ToList();
    for (int i = 0; i < itemsOrganizados.Count; i++)
    {
        for (int c = 0; c < Frete.Containers.Count; c++)
        {
            if ((Frete.CargaMax >= Frete.Containers[c].Carga + itemsOrganizados[i].item.Peso) &&
                (Frete.VolumeMax >= Frete.Containers[c].Volume + itemsOrganizados[i].item.Volume))
            {
                if (itemsOrganizados[i].qtdItemFrete < Frete.QtdMaxItem)
                {
                    Frete.Containers[c].Items.Add(itemsOrganizados[i].item);
                    itemsOrganizados[i] = (itemsOrganizados[i].qtdItemFrete + 1, itemsOrganizados[i].item);
                }
            }
        }
    }
    return Frete.Containers;
}
```

```
1 reference
public List<(double, Item)> OrdenaItems()
{
    var items = new List<(double beneficio, Item item)>();
    foreach (var item in Frete.ItemsDisponiveis)
    {
        items.Add((item.Lucro / item.Peso / item.Volume, item));
    }
    items.Sort((x, y) => y.beneficio.CompareTo(x.beneficio));
    return items;
}
```