

Problema 2

- Variáveis do modelo

`construido` : array(depositos) of mpvar – essa variável binária indica se no local `L` foi construído um depósito. Quando no local `L` foi construído um depósito, `construido(L) = 1`, e `construido(L) = 0` caso contrário.

`setor_x_deposito` : array(setores, depositos) of mpvar – matriz que indica se um setor `S` deposita seu lixo no depósito `D`. Essa variável também é binária. Quando o depósito `D` recebe o lixo do setor `S` `setor_x_deposito(S,D) = 1`, e `setor_x_deposito(S,D) = 0` caso contrário.

- Restrições

As restrições de integralidade foram aplicadas as variáveis do modelo, sendo as duas variáveis binárias.

A segunda restrição desse problema, diz respeito a que um setor deve depositar seu lixo em apenas um depósito. Como a variável `setor_x_deposito` controla em qual depósito cada setor deposita o lixo, colocamos a restrição de que para todo `S` em setores, e `D` variando de 1 a número de locais, a somatória `setor_x_deposito(S,L)` deve ser igual a 1.

A terceira restrição, serve para limitar a quantidade de lixo que um depósito recebe. Ela não deve ultrapassar a capacidade do depósito, definida pela variável `capacidade` : array (depositos) of integer. Para todo depósito `D`, a soma da quantidade de lixo dos setores alocados a ele, não deve ultrapassar a sua capacidade. A quantidade de lixo total dos setores alocados a ele é calculada através de `sum(i in setores) (req(i) * setor_x_deposito(i,j))`.

- Função objetivo

O objetivo desse problema é minimizar os custos de instalação de depósitos, e os custos de transporte de lixo entre os setores e seus respectivos depósitos. Os custos de transporte são definidos por `CustoSetorDeposito := sum(i in setores, j in depositos) 1000 * req(i) * distancia(i,j) * custo_km * setor_x_deposito(i,j)`, isto é, a produção de cada setor, multiplicada pela distância do setor ao depósito, multiplicada pelo custo/km.

O custo fixo, é calculado através de `CustoFixoInstalacaoDepositos := sum(j in depositos) custo_fixo(j) * construido(j)`, isto é, a soma dos custos de instalação de cada depósito construído.

Sendo assim, a função objetivo é dada por `minimize(CustoSetorDeposito + CustoFixoInstalacaoDepositos)`.

Problema 3

- Variáveis do modelo

`local_x_cadeia` : array(locais, cadeias) of mpvar – essa variável binária indica se no local L foi construído um restaurante da cadeia C . Quando no local L foi construído um restaurante da cadeia C , então $\text{local_x_cadeia}(L,C) = 1$, e $\text{local_x_cadeia}(L,C) = 0$ caso contrário.

- Restrições

A restrição de integralidade foi aplicada a variável, sendo ela binária.

Como em cada local só pode ser construído um restaurante de uma determinada cadeia, foi criada uma restrição para atender esse requisito. $\text{forall } (l \text{ in locais}) \text{sum}(c \text{ in cadeias}) \text{local_x_cadeia}(l, c) \leq 1$, isto é, para todo local L , a soma das cadeias instaladas deve ser no máximo.

A terceira restrição, diz que se um restaurante está localizado a menos de 5 km de um outro restaurante, eles não podem ser da mesma cadeia. Para todos locais I e J , e cadeias C , se I e J estão próximos, então $\text{local_x_cadeia}(i, c) + \text{local_x_cadeia}(j, c) \leq 1$.

No problema b, existe uma quarta restrição. Ela diz que se for instalado um restaurante de uma determinada cadeia, próximo a ele deve existir um restaurante da outra cadeia. Portanto, a somatória de $\text{local_x_cadeia}(j, c1)$ para qualquer local J próximo do local I , deve ser maior que $\text{local_x_cadeia}(i, c2)$. Se o local I possuir um restaurante da cadeia $c2$, então próximo a I devem existir um ou mais restaurantes da outra cadeia $c1$.

- Função objetivo

O objetivo desse problema é maximizar o lucro aferido pelo grupo. Para isso, para cada par local/cadeia instalado, é calculado seu lucro. O lucro final, é a somatória do lucro de cada restaurante construído.

Problema 4

- Variáveis do modelo

`tanque_x_combustivel` : array(tanques, combustiveis) of mpvar – essa variável binária indica se o tanque T contém o combustível C. Se o tanque T contém o combustível C, então `tanque_x_combustivel(T,C) = 1`, e `tanque_x_combustivel(T,C) = 0` caso contrário.

`qtde_tanque_combustivel` : array(tanques, combustiveis) of mpvar – matriz que indica a quantidade de um combustível C num tanque T.

- Restrições

As restrições de integralidade foram aplicadas as variáveis, sendo `tanque_x_combustivel` binária, e `qtde_tanque_combustivel` inteira.

Para todo tanque T, o número de tipos de combustíveis contidos deve ser 1. Essa restrição foi formulada com base na variável `tanque_x_combustivel` já que ela controla qual combustível está em um determinado tanque.

```
forall (t in tanques)
    (sum(c in combustiveis) tanque_x_combustivel(t,c)) = 1
```

O máximo que é transportado de um combustível é a sua demanda. Essa restrição foi formulada com base na variável do modelo `qtde_tanque_combustivel` e na variável `demanda(combustiveis)` of integer, que diz qual é a demanda de um combustível C.

```
forall (c in combustiveis)
    (sum(t in tanques) qtde_tanque_combustivel(t,c)) <= demanda(c)
```

A última restrição, diz respeito à capacidade do tanque. Obviamente, um tanque só pode transportar a sua capacidade, dada por `capacidade` : array (tanques) of integer. A partir da variável `qtde_tanque_combustivel` nós sabemos qual a quantidade no tanque T do combustível C. Sendo assim, temos:

```
forall (t in tanques, c in combustiveis)
    qtde_tanque_combustivel(t,c) <=
        capacidade(t) * tanque_x_combustivel(t,c)
```

- Função objetivo

O objetivo desse problema é maximizar a quantidade de combustível levado, ou seja, minimizar a quantidade de combustível não transportada. A quantidade transportada foi obtida através da soma da quantidade de combustível levada por cada tanque.