

# Trabalho Computacional - Teoria da Decisão - UFMG

Pedro Loes

27/02/2021

## Resumo

- *Este artigo apresenta uma solução para um problema aplicado do curso de Teoria da Decisão do departamento de **Engenharia de Sistemas** da **UFMG** ministrado pelo professor Lucas Batista. O objetivo deste trabalho foi aplicar as técnicas de **otimização** e **teoria da decisão** lecionadas ao longo do semestre. O desenvolvimento compreende a exploração de **mínimos locais** e **mínimos globais**. A conclusão visou **otimizar** o número de **pontos de acesso** para suprir os **consumos de banda** demandados pelos clientes de uma conferência, atendendo à **restrições pré-estabelecidas**. Finalmente na etapa **otimizador em produção** foi criado um aplicativo para generalizar o uso do algoritmo.*

## Introdução

- O problema foi modelado no software **R** com implementação da função objetivo em **C++**.
- Diferentes espaços de busca foram considerados com o intuito de descobrir a partir de qual densidade de espaço de busca **discretizado** o algoritmo convergiria para a solução de **mínimo global ótima**.

## Especificações do Problema

- Deseja-se instalar uma rede **WLAN** do tipo **N 2D** para atendimento de um centro de convenções com **800 × 800 metros**. Para planejamento dessa rede foram estimados **500** pontos de demanda, com suas respectivas **posições** geográficas e **consumos** de largura de **banda**. O arquivo **clientes.csv** contém:
  - Coordenada **x** do cliente em metros.
  - Coordenada **y** do cliente em metros.
  - Consumo de banda do cliente em **Mbps**.
- Amostra das **10** primeiras observações:

Id	x	y	Mbps
1	168.99	220.62	0.47347
2	218.11	211.16	0.41026
3	207.48	157.38	1.37570
4	213.25	189.93	0.93025
5	199.19	140.71	0.15790
6	177.74	188.51	0.94899
7	197.44	176.30	0.30420
8	161.48	197.12	1.16990
9	189.84	211.32	1.29950
10	269.22	224.90	0.51370

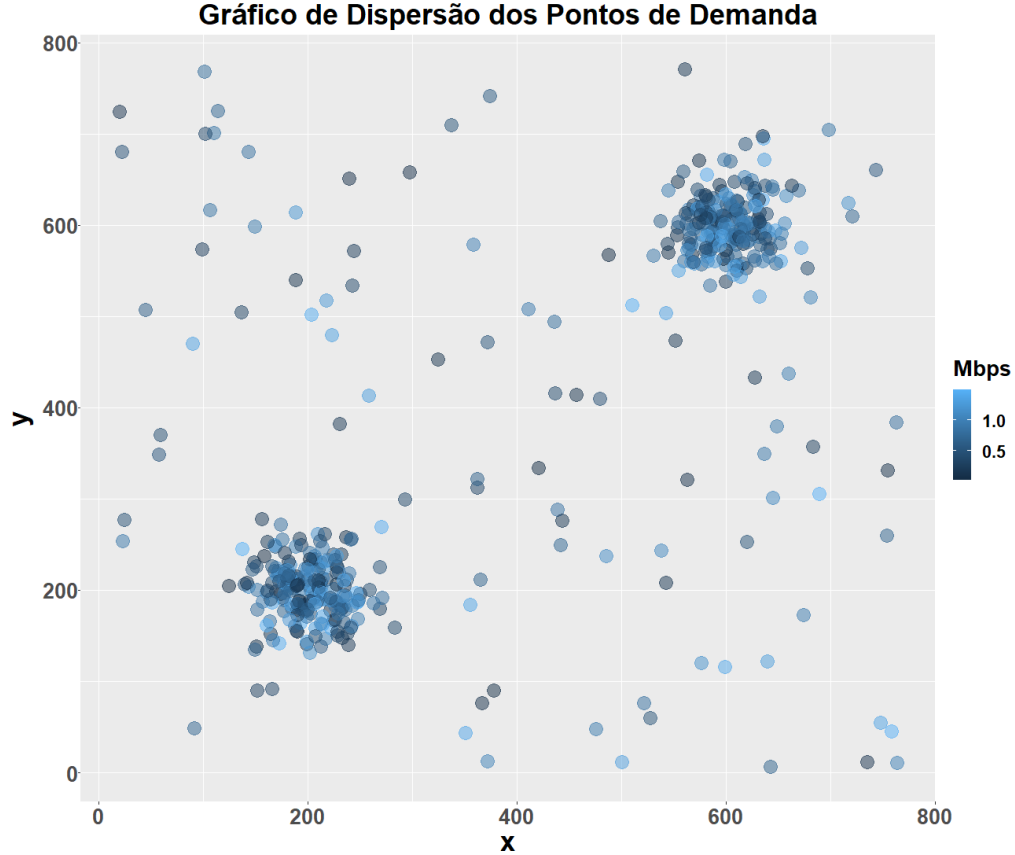
- Variáveis de Decisão:
  - Coordenadas dos pontos a serem instalados na área de **800 x 800 metros**.
  - Ponto de Acesso que será responsável pelo atendimento de cada cliente.
- Restrições:
  - Ao menos **95%** dos pontos de demanda devem ter suas demandas integralmente atendidas.
  - Cada ponto de acesso a ser instalado tem capacidade de **150Mbps**, que não pode ser excedida.
  - Um cliente pode ser atendido por um **Pa** se a distância entre ambos é inferior a **85 metros**.
  - Cada cliente só pode ser atendido por um único **Pa**.
  - Devido a restrições orçamentárias, podem ser instalados no máximo **100 Pa's**.
- Simplificações:
  - Os pontos de demanda e seus **consumos** de banda são **estáticos**.
  - O efeito de **obstáculos** no ambiente foram **desprezados**.
  - Um ponto de acesso **não** causa **interferência** em outros.

## Modelagem

- Definição de Variáveis:
  - $Pd_i \leftarrow$  Pontos de Demanda  $i = \{1, 2, 3, \dots, 500\}$ ,  $Pd_i \in R\{x, y\}$ ,  $x, y = \{1, \dots, 800\}$
  - $Pa_j \leftarrow$  Pontos de Acesso  $j = \{1, 2, \dots, 100\}$ ,  $Pa_j \in R\{x, y\}$ ,  $x, y = \{1, \dots, 800\}$
  - $\beta_i \leftarrow$  Consumo de banda em Mbps dos Pd's,  $\forall i \in Pd_i$
  - $d_{i,j} \leftarrow$  Distância Euclidiana entre  $Pd_i$  e  $Pa_j$ ,  $\forall i \in Pd_i, \forall j \in Pa_j$
  - $\alpha_j \leftarrow$  Ativação do  $Pa_j$ ,  $\alpha_j \in \{0, 1\} \forall j \in Pa_j$
  - $\eta_{i,j} \leftarrow$  Atendimento do  $Pd_i$  pelo  $Pa_j$ ,  $\eta_{i,j} \in \{0, 1\} \forall i \in Pd_i, \forall j \in Pa_j$
- Definição da Função:
  - $\min (\sum_{j=1}^{|Pa|} \alpha_j)$
- Definição das Restrições:
  - $\sum_{i=1}^{|Pd|} \sum_{j=1}^{|Pa|} \eta_{i,j} \geq 475$
  - $\sum_{i=1}^{|Pd|} \eta_{i,j} \times \beta_i \leq 150 \quad \forall j \in \alpha_j = 1$
  - $\sum_{j=1}^{|Pa|} \eta_{i,j} \leq 1 \quad \forall i \in Pd$
  - $\sum_{j=1}^{|Pa|} \alpha_j \leq 100$
  - $\exists \eta_{i,j} \forall d_{i,j} \leq 85$

## Análise Exploratória

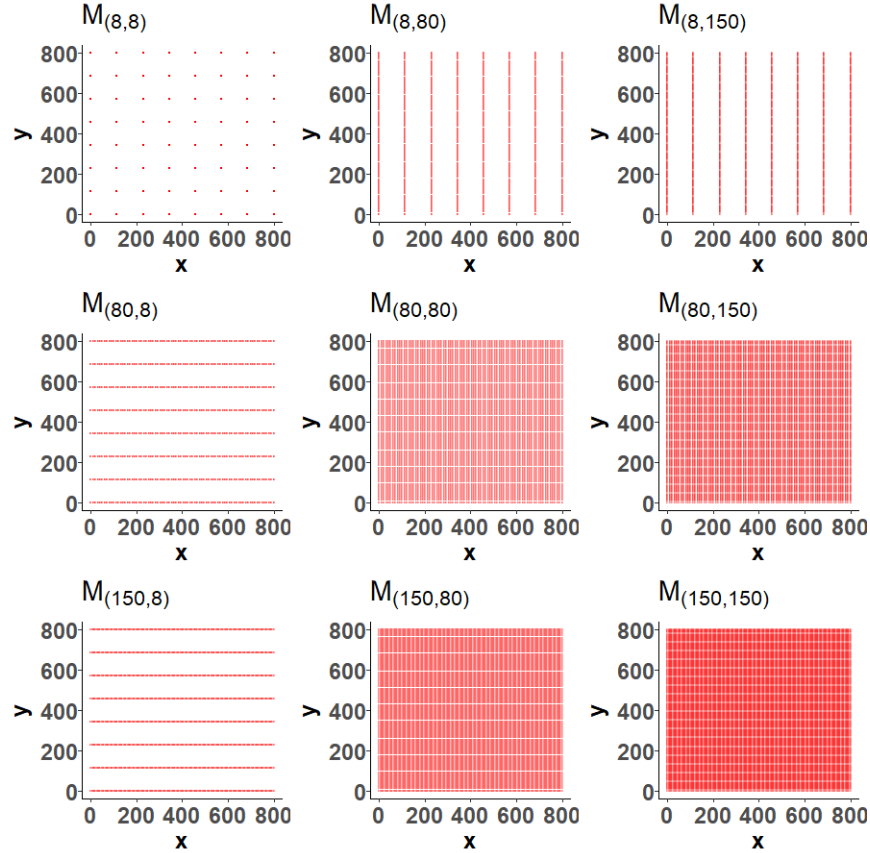
- Para ilustrar uma visão inicial da distribuição dos clientes no espaço de 800 x 800 foi elaborado um gráfico de dispersão 2d com os dados da localização (**x, y**) dos pontos de demanda.
- O parâmetro **alpha** de transparência foi alterado para **0.5** com o intuito de eliminar o problema da sobreposição de **Pd's** e facilitar a compreensão da densidade destes pontos.
- A dimensão de cor foi adicionada considerando o gradiente das cores preta até azul para o consumo de largura de banda requisitada em **Mbps** por cada cliente.



- Existem **2** grandes clusters na dispersão da localização dos clientes em torno dos pontos **(200, 200)** e **(600, 600)**. Os demais pontos apresentam distribuição esparsa na área de **800 x 800** metros.
- Na diagonal que vai de **(0,0)** até **(800,800)** os pontos de demanda são menos espalhados mas na diagonal **(0,800)** até **(800,0)** os pontos estão mais dispersos.
- Parece não existir padrão visual para a distribuição das demandas de banda em **Mbps**.

## Espaços de Busca

- Se considerada uma escala contínua, a área de **800 x 800** metros possui infinitas coordenadas de possíveis localizações para os pontos de acesso.
- A construção de espaços de busca considerou espaços discretizados na forma de matrizes do tipo  $M_{(i,j)}$  com  $i = \{8, 9, \dots, 150\}$  e  $j = \{8, 9, \dots, 150\}$ . A posição **i** representa a quantidade de posições igualmente espaçadas no eixo **x** e da mesma forma, a posição **j** no eixo **y**.



- O 1º gráfico no canto **superior esquerdo** ilustra a área discreta de **800 x 800** em **8** posições igualmente espaçadas para o eixo **x** e **y** totalizando **64** possíveis localizações para os pontos de acesso.
- Os gráficos **fora da diagonal principal** ilustram alguns dos espaços discretos irregulares arbitrariamente escolhidos, ou seja, com um número de possíveis posições diferente para os eixos **x** e **y**.
- Os gráficos **na diagonal principal** ilustram alguns dos espaços discretos regulares, ou seja, com a mesma escala de números discretos para os eixos **x** e **y**. Por serem igualmente espaçados, apresentam o mesmo número de possíveis posições para os eixos **x** e **y**.
- O **último gráfico** no canto inferior direito ilustra a área discreta de **800 x 800** em **150** posições igualmente espaçadas para o eixo **x** e **y** totalizando **22500** possíveis localizações para os pontos de acesso.

## Prototipagem do Algoritmo

- Testes exploratórios:
  - Espaços discretos menos granulares que  $M_{(8,8)}$  não apresentam solução.
  - Espaços discretos mais granulares que  $M_{(150,150)}$  apresentam convergência estável.
  - O custo computacional para executar o algoritmo em **R** foi na unidade de **dias**.
- Os laços para inspecionar o melhor posicionamento dos pontos de acesso em cada espaços de busca foram prototipados em **C++**.
- A importação dos dados, a declaração de objetos e a construção dos espaços de busca foram prototipados no ambiente **R**.

## Prototipagem do modulo Rastreador em C++

```
// Função para calcular o quadrado de um número
double enquadra(double x){
    double quadrado = x * x;
    return quadrado;
}

// Função para contar o nº de Pd's cobertos pelo raio de um Pa
std::vector<double> conta_pontos(x, y, wlanx, wlany, indice) {
    int n = wlany.size(); double soma;
    double raio = square(85); indices(0);
    for(int i = 0; i < n; i++) {
        soma = enquadra(wlanx[i] - x) + square(wlany[i] - y);
        if( soma <= raio) {
            indices.push_back(indice[i]);}
    }
    return indices;
}

// Função para inspecionar espaços de busca discretos
List rastreia(int intervalo_x, int intervalo_y, DoubleVector wlanx,
              DoubleVector wlany, DoubleVector centro_x, DoubleVector centro_y,
              NumericVector indice, int mais_populoso, std::vector<int> viola) {
    std::vector<double> cobertura(0);
    List ret;
    int intervalos = intervalo_x * intervalo_y;
    for (int j = 2; j <= intervalos; j++) {
        if( (conta_pontos(centro_x[j],centro_y[j],
                        wlanx, wlany, indice).size() >=
                        conta_pontos(centro_x[mais_populoso], centro_y[mais_populoso],
                        wlanx, wlany, indice).size() ) &&
            (! std::count(viola.begin(), viola.end(), mais_populoso) ) ){
            mais_populoso = j;
            cobertura = conta_pontos(centro_x[mais_populoso],
                                    centro_y[mais_populoso],wlanx,wlany,indice);}
        ret["cobertura"] = cobertura;
        ret["mais_populoso"] = mais_populoso;}
    return ret;}
}
```

- A inspeção das matrizes de busca de espaços discretos quadrados e não quadrados entre  $M_{(8,8)}$  e  $M_{(150,150)}$ , bem como a contagem de pontos de demanda dentro do raio de cobertura de um ponto de acesso foram prototipadas com o modulo rastreador no ambiente C++. Esse modulo contém as seguintes funções:
  - **enquadra** eleva um número ao quadrado.
  - **conta\_pontos** conta o número de pontos dentro do raio de cobertura de um ponto de acesso.
  - **rastreia** inspeciona as melhores coordenadas para cada ponto de acesso dado um espaço discreto.

## Prototipagem dos Espaços de Busca em R

```
# Declara objetos e carrega modulo rastreador.cpp
wlan_completa <- read_csv("clientes.csv", col_names = c("x", "y", "Mbps"))
wlan_id <- wlan_completa %>% mutate(indice = 1:nrow(wlan_completa))
performance <- tibble(intervalos = 8:50, PA = rep(0,143), tempo = rep(0,143))
sourceCpp('rastreador.cpp')

# Laços para combinação de espaços discretos
for(intervalo_x in seq(8, 150, by = 1)){
  for(intervalo_y in seq(8, 150, by = 1)){
    centro_x <- rep(seq(0, 800, length.out = intervalo_x), each = intervalo_y)
    centro_y <- rep(seq(0, 800, length.out = intervalo_y), times = intervalo_x)
    wlan <- wlan_id; vencedores <- NULL; contador <- 0; Mbps <- NULL; viola <- numeric(1)
    inicio <- Sys.time()
    while(contador < 475){
      cobertura <- NULL; mais_populoso <- 1
      rastreado <- rastreia(intervalo_x = intervalo_x, intervalo_y = intervalo_y,
                           wlanx = wlan$x, wlany = wlan$y,
                           centro_x = centro_x, centro_y = centro_y,
                           indice = wlan$indice, mais_populoso = mais_populoso,
                           viola = viola)

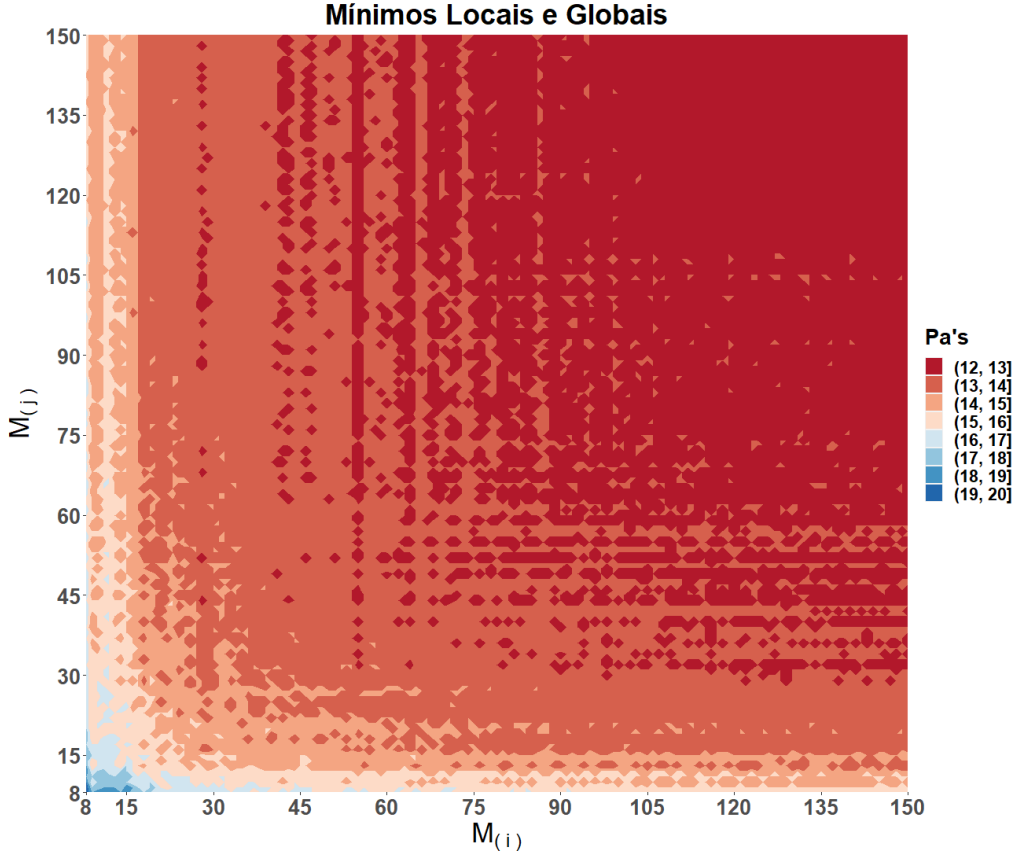
      megas <- wlan %>%
        filter(indice %in% as.integer(rastreado[[1]])) %>%
        summarise(total = sum(consumo)) %>%
        pull(total)
      if(megas < 150){
        Mbps <- c( Mbps, megas)
        vencedores <- c(vencedores, rastreado[[2]])
        wlan <- wlan %>% filter(!indice %in% as.integer(rastreado[[1]]))
        contador <- contador + length(rastreado[[1]])
        print(paste0("Indice: ", intervalo_x, " - ", intervalo_y, " - ",
                     "Contador: ", contador))} else { viola <- c(viola, rastreado[[2]])}
    }
    fim <- Sys.time()
    performance$tempo[indice_perfor] <- fim - inicio
    performance$PA[indice_perfor] <- length(vencedores)
    indice_perfor <- indice_perfor + 1}}
}
```

- Para leitura dos dados e laço para combinação de espaços foram utilizadas as facilidades do ambiente **R** em operar com data frames e gerar vetores de seqüências.
- A função **rastreia** foi utilizada no ambiente **R** por meio do pacote **Rcpp** que carrega o modulo **rastreador.cpp** e constrói a conexão entre os dois ambientes.

## Convergência

- O algoritmo completo utilizado para convergência considerou as combinações de espaços de busca de  $M_{(8,8)}$  até  $M_{(150,150)}$  contabilizando o mínimo de **64** o máximo de **22500** coordenadas para possíveis localizações dos pontos de acesso.
- O mapa de superfície tridimensional foi construído para ilustrar as regiões de inspeção do algoritmo. O gradiente das cores **azul(máximo)** e **vermelho(mínimo)** foi utilizado como a terceira dimensão do

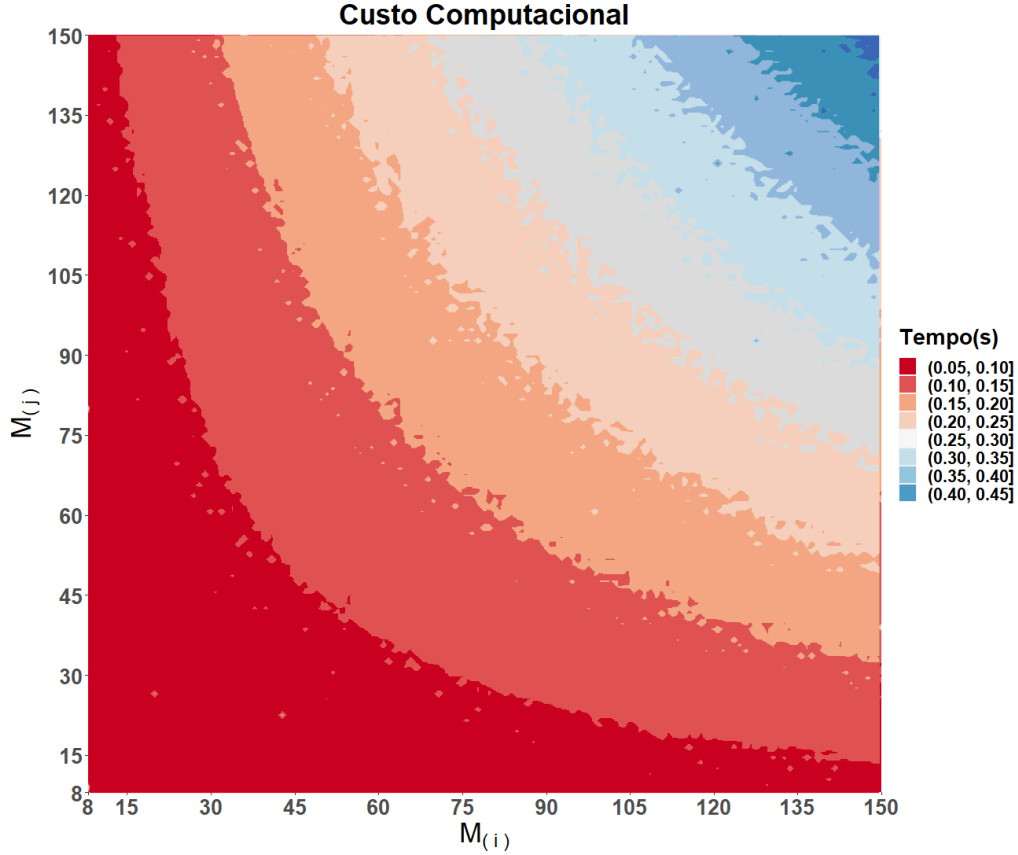
gráfico para ilustrar o número de pontos de acesso necessários para atingir um mínimo local ou global em cada combinação de espaços discretos.



- O ponto na coordenada  $i = 8$  do eixo  $M_{(i)}$  e  $j = 8$  do eixo  $M_{(j)}$ , corresponde a inspeção do espaço de busca  $M_{(8,8)}$  que atinge a pior solução ótima de mínimo local gastando **20 Pa's**.
- Entre  $M_{(8,8)}$  e  $M_{(30,30)}$  o algoritmo passa pelos mínimos locais de **20** até **13 Pa's**, em  $M_{(27,43)}$  atinge pela primeira vez o mínimo global de **12 Pa's** e em  $M_{(90,75)}$  o algoritmo começa a estabilizar em **12 Pa's** com a maioria dos espaços convergindo para **12 Pa's**.
- O algoritmo estabiliza em **12 Pa's** para espaços discretos mais granulares que  $M_{(110,110)}$ . Tal fato indicou convergência porque até  $M_{(150,150)}$  o número de **Pa's** permanece em **12**.

## Custo Computacional

- As performances de custo computacional do algoritmo nas combinações de espaços de busca foram comparadas utilizando a função **Sys.time** do **R**. O hardware utilizado foi um processador **Intel i7** com **16 GB RAM**.
- O mapa de superfície tridimensional foi construído para ilustrar os tempos computacionais necessários para atingir um mínimo local ou global em cada combinação de espaços discretos.

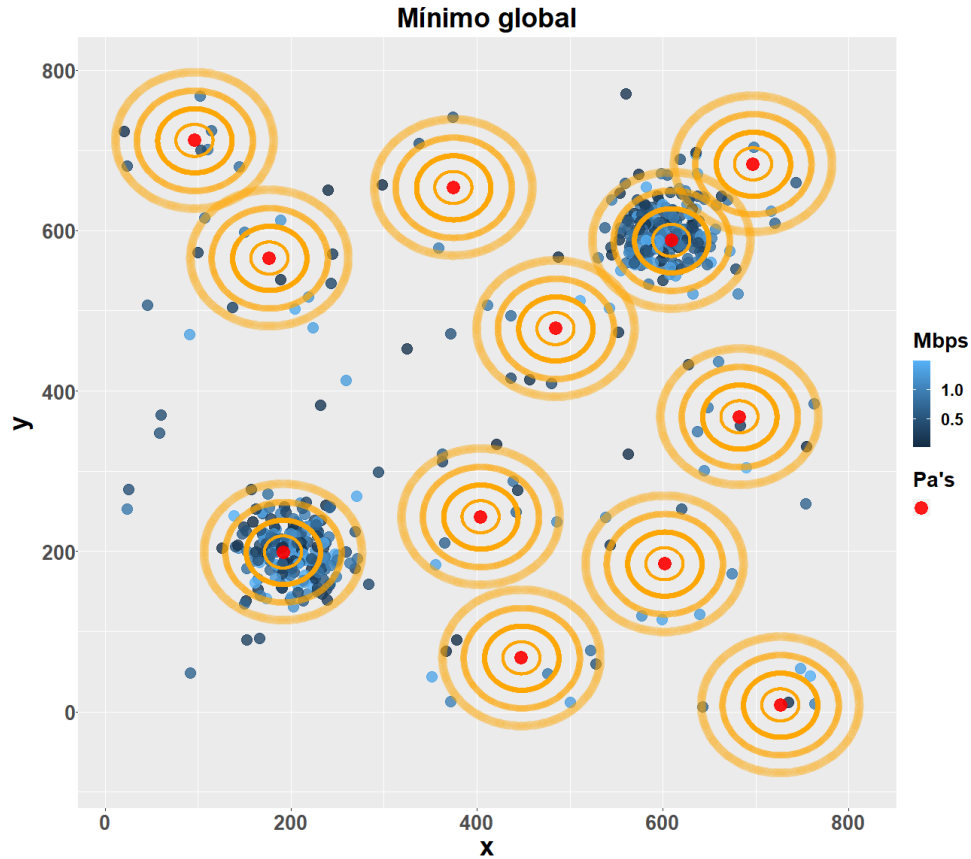


- O ponto na coordenada  $i = 8$  do eixo  $M_{(i)}$  e  $j = 8$  do eixo  $M_{(j)}$ , corresponde a inspeção do espaço de busca  $M_{(8,8)}$  que atinge a pior solução ótima de mínimo local com custo computacional de **0.02 segundos**.
- É possível observar **9** intervalos bem distintos com alguns poucos pontos discrepantes dentro de cada curva de nível. Pode-se verificar que a curva vai se tornando mais **linear** a medida que se aproxima dos maiores tempos. Os custos variaram entre **0.02** e **0.48 segundos**.
- O custo computacional total gasto para inspecionar todos os espaços discretos foi de aproximadamente de **41 minutos**.

## Mínimo Global

- A solução ótima de mínimo global considerou o espaço de busca  $M_{(110,110)}$  totalizando **12100** coordenadas de possíveis localizações para os pontos de acesso. Essa solução foi escolhida porque é um ponto de transição da convergência para estabilização do mínimo global.
- O algoritmo inspecionou o espaço de busca de **12100** possíveis localizações para os pontos de acesso e recuperou as localizações de **12** coordenadas que minimizam o número de pontos de acesso e satisfazem as condições do problema.





- O algoritmo gastou **0.35 segundos** para atingir o mínimo local de **12 Pa's**. É possível observar que ocorre somente uma sobreposição de áreas de cobertura dos pontos de acesso no quadrante superior direito do gráfico.
- Os **2 Pa's** posicionados nas coordenadas **(192, 199)** e **(609, 587)** estão cobrindo um grande número de **Pd's** podendo apresentar consumos de banda próximos da saturação.

## Tabela de Resultados

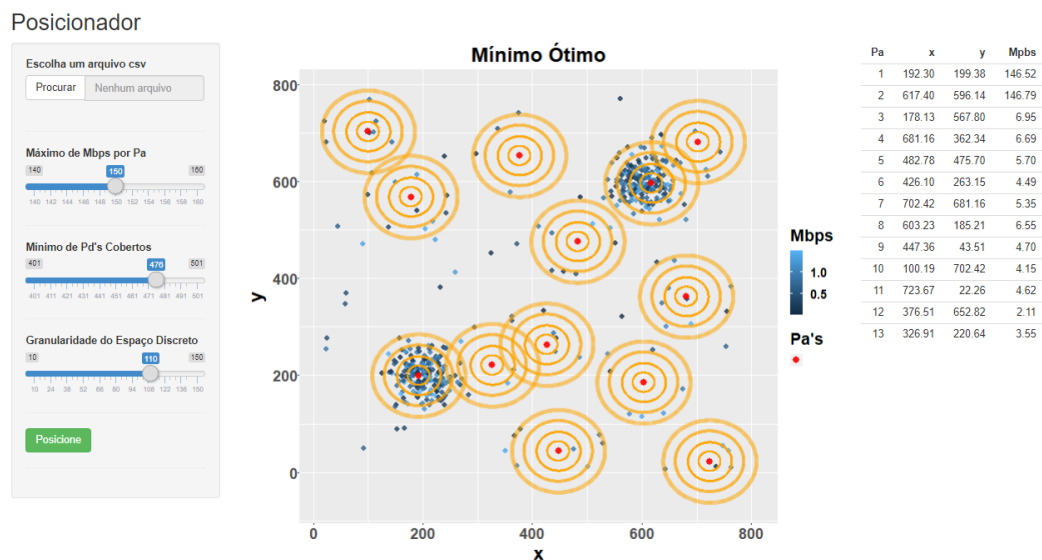
- Tabela de localização dos **12 Pa's** na área de **800 x 800** com os respectivos consumos totais em **Mbps**:

Pa	x	y	Consumo (Mbps)
1	192	199	147.2
2	609	587	148.1
3	177	565	6.9
4	683	368	6.7
5	485	477	5.7
6	404	243	5.8
7	697	683	5.4
8	602	184	6.6
9	96	712	4.2
10	448	67	3.7
11	727	8	4.6
12	375	653	2.1

- É possível observar que os dois clusters principais com um total de **147** e **148** Mbps de consumo estão muito próximos da saturação e talvez fosse adequado a colocação de mais **2 Pa's** nessas regiões para garantir a integridade da entrega do sinal onde há maior concentração de **pontos de demanda**.
- Outra possível consideração seria a de que a diminuição da cobertura de cada **Pa** de **95%** ou **475 Pd's** para **90%** ou **450 Pd's** reduziria o custo total de **Pa's** em **33%** ou seja de **12** para **8 Pa's**.

## Otimizador em Produção

- Para atender a demanda da unidade de decisão a nível de produção foi desenvolvido o aplicativo **Posicionador**.
- Este **programa iterativo** é composto de uma barra lateral para carregar um arquivo **.csv** com 3 colunas. A primeira e a segunda são referentes às coordenadas **(x,y)** dos pontos de demanda e a terceira ao consumo de banda de cada ponto de demanda em **Mbps**. O painel é composto de um gráfico que mostra o posicionamento dos **Pa's** e uma tabela que mostra a localização exata de cada **Pa**, bem como a largura de banda que será demandada.
- Esta ferramenta irá permitir que a **unidade de decisão** analise os cenários em tempo real otimizando o **tempo humano** gasto pela unidade de decisão para escolher o **melhor cenário**. Outra vantagem é que a unidade decisão poderá fazer **ajustes finos** no otimizador ao invés de escolher entre opções pré-configuradas.



## Referências

- [Projeto WLAN github](#)