Licenciatura em Ciência de Dados - 2º ano

Trabalho de grupo

Otimização Heurística

3 de junho de 2024

Discentes: João Batista nº 111611 / João Dias nº 110305
António Teotónio nº 111283 / Diogo Aqueu nº 110705

2023/2024

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA

Índice

a) Descreva por palavras em que consiste uma solução admissível para o problema

Uma solução admissível para o problema da clínica envolve a **distribuição dos enfermeiros pelos procedimentos** médicos, **minimizando** a **duração total do dia** de trabalho e respeitando as restrições e critérios definidos:

1. Atribuição de Enfermeiros aos Procedimentos:
    a. Cada procedimento requer **exatamente 3 enfermeiros**.
    b. Os **procedimentos regulares** podem ser realizados por **qualquer enfermeiro**, enquanto que os **procedimentos complexos** só podem ser realizados por enfermeiros das categorias 2 e 3.
2. Limites de Participação dos Enfermeiros:
    a. Cada enfermeiro pode participar em no **máximo 5 procedimentos** ao longo do dia.
    b. O mesmo enfermeiro **não pode** participar em **ambos os procedimentos** do **mesmo período**.
3. Divisão do Trabalho:
    a. Há sempre **dois procedimentos** a decorrer em **simultâneo** em cada período e um total de 7 períodos num dia de trabalho.
    b. O próximo período só **começa** quando **ambos** os procedimentos do período **anterior** estiverem **terminados**.
    c. A **duração** de cada **procedimento** corresponde ao **tempo máximo** entre os **três enfermeiros** que lhe estão afetos.
    d. A **duração** de cada **período** é determinada pelo **procedimento mais demorado** dentro desse período.

Por outras palavras, o objetivo é **minimizar a soma das durações dos 7 períodos**, **sem violar** nenhuma das **restrições** acima definidas.

b) Desenvolva uma heurística para determinar uma afetação admissível dos enfermeiros aos procedimentos e apresente uma solução admissível para o problema da clínica

Tendo em conta o que foi feito no ponto a), para a definição da heurística vamos percorrer as seguintes etapas (note-se que poderiam ter sido definidas outras regras e outras heurísticas desde que os pontos descritos em a) fossem respondidos):

Passo 1: Descrição Detalhada da Alocação

1. **Listas e Dicionários:**
    a. Criar uma lista para cada enfermeiro que rastreie a quantidade de procedimentos em que eles já participaram e os períodos em que estão alocados.

b. Criar uma estrutura que permita armazenar a informação sobre a alocação dos procedimentos para cada período.

2. **Percorrer Períodos:**
    a. Para cada período (1 a 7):
        i. Obter a lista de procedimentos a serem realizados no período.
3. **Processar Cada Procedimento:**
    a. Para cada procedimento do período:
        i. Determinar as categorias permitidas:
            - Procedimentos regulares (P1, P2, P4, P5, P6, P9, P11, P13, P14): qualquer categoria (1, 2, 3).
            - Procedimentos complexos (P3, P7, P8, P10, P12): apenas categorias 2 e 3.
        ii. Criar uma lista temporária para armazenar os enfermeiros selecionados.
        iii. Ordenar a lista de enfermeiros por ordem crescente de tempo para o procedimento que está a ser processado.
        iv. Percorrer a lista de enfermeiros:
            - Verificar se o enfermeiro pertence a uma das categorias permitidas.
            - Verificar se o enfermeiro ainda não atingiu o limite de 5 procedimentos.
            - Verificar se o enfermeiro não está alocado a outro procedimento no mesmo período.
            - Se não houverem enfermeiros das categorias 2 e 3 para procedimentos complexos, fazer uma permutação com um enfermeiro da categoria 1 que não esteja alocado a esse período, e que não tenha excedido o máximo de 5 participações.
            - Se todas as condições forem satisfeitas, adicionar o enfermeiro à lista temporária de selecionados.
            - Continuar até selecionar 3 enfermeiros.
4. **Atualizar Informações:**
    a. Após selecionar os enfermeiros para um procedimento:
        i. Atualizar a contagem de procedimentos para cada enfermeiro selecionado.
        ii. Atualizar a lista de períodos alocados para cada enfermeiro selecionado.
        iii. Registar a alocação do procedimento e dos enfermeiros selecionados na estrutura de alocação.

Passo 2: Resultados

- Após percorrer todos os períodos e procedimentos, a estrutura de alocação conterá a distribuição de enfermeiros para cada procedimento em cada período.
- A estrutura final deve garantir que todas as restrições sejam atendidas e que a distribuição seja feita de maneira a minimizar possíveis conflitos de alocação.

Como chegar a uma solução admissível:

Primeiro criamos a lista dos enfermeiros e o dicionário dos períodos.

enfermeiros =

- 'E1': {'categoria': 1, 'procedimentos': 0, 'periodos': []},
- 'E2': {'categoria': 1, 'procedimentos': 0, 'periodos': []},
- 'E3': {'categoria': 1, 'procedimentos': 0, 'periodos': []},
- 'E4': {'categoria': 1, 'procedimentos': 0, 'periodos': []},
- 'E5': {'categoria': 2, 'procedimentos': 0, 'periodos': []},
- 'E6': {'categoria': 2, 'procedimentos': 0, 'periodos': []},
- 'E7': {'categoria': 2, 'procedimentos': 0, 'periodos': []},
- 'E8': {'categoria': 2, 'procedimentos': 0, 'periodos': []},
- 'E9': {'categoria': 3, 'procedimentos': 0, 'periodos': []},
- 'E10': {'categoria': 3, 'procedimentos': 0, 'periodos': []}

periodos =

- 1: ['P1', 'P2'],
- 2: ['P3', 'P4'],
- 3: ['P5', 'P6'],
- 4: ['P7', 'P8'],
- 5: ['P9', 'P10'],
- 6: ['P11', 'P12'],
- 7: ['P13', 'P14']

Depois definimos os procedimentos regulares e complexos.

1. Procedimentos **regulares (P1, P2, P4, P5, P6, P9, P11, P13, P14)**: Podem ser realizados por enfermeiros de qualquer categoria (1, 2 ou 3).
2. Procedimentos **complexos (P3, P7, P8, P10, P12)**: Podem ser realizados apenas por enfermeiros das categorias 2 e 3.

Podemos agora processar cada procedimento, utilizaremos uma lista temporária para armazenar os enfermeiros selecionados para cada período. A cada novo período, esta lista será esvaziada para garantir que novos enfermeiros sejam selecionados conforme as restrições.

Período 1:

1. **P1 (Regular)**:
   - Enfermeiros de qualquer categoria.
   - Cria uma lista temporária para os enfermeiros desse período.
   - Ordenamos os enfermeiros para o procedimento em questão por ordem crescente, ficando desta forma: [E3; E6; E7; E4; E9; E1; E2; E5; E8; E10]
   - Seleciona-se 3 enfermeiros que ainda não têm alocações no Período 1 e adiciona-se à lista temporária:

        - E3, E6, E7
    - Lista temporária: [E3, E6, E7]

Como dito na explicação da heurística, sempre que um procedimento acaba a lista dos enfermeiros e o dicionário das alocações devem ser atualizados. Iremos apenas mostrar essa atualização neste primeiro procedimento para a demonstração não ficar muito extensa mas o objetivo passa por estas serem sempre atualizadas.

enfermeiros =

- 'E1': {'categoria': 1, 'procedimentos': 0, 'periodos': []},
- 'E2': {'categoria': 1, 'procedimentos': 0, 'periodos': []},
- 'E3': {'categoria': 1, 'procedimentos': 1, 'periodos': [1]},
- 'E4': {'categoria': 1, 'procedimentos': 0, 'periodos': []},
- 'E5': {'categoria': 2, 'procedimentos': 0, 'periodos': []},
- 'E6': {'categoria': 2, 'procedimentos': 1, 'periodos': [1]},
- 'E7': {'categoria': 2, 'procedimentos': 1, 'periodos': [1]},
- 'E8': {'categoria': 2, 'procedimentos': 0, 'periodos': []},
- 'E9': {'categoria': 3, 'procedimentos': 0, 'periodos': []},
- 'E10': {'categoria': 3, 'procedimentos': 0, 'periodos': []}

periodos =

- 1:[('P1', [E3, E6, E7]), 'P2'],
- 2: ['P3', 'P4'],
- 3: ['P5', 'P6'],
- 4: ['P7', 'P8'],
- 5: ['P9', 'P10'],
- 6: ['P11', 'P12'],
- 7: ['P13', 'P14']

2. **P2 (Regular)**:
    - Enfermeiros de qualquer categoria.
    - Ordenamos os enfermeiros para o procedimento em questão por ordem crescente, ficando desta forma: [E8; E4; E10; E2; E6; E5; E3; E1; E7; E9]
    - Seleciona-se 3 enfermeiros que ainda não têm alocações no Período 1 e adiciona-se à lista temporária:
        - E8, E4, E10
    - Lista temporária: [E3, E6, E7, E8, E4, E10]

Período 2:

- Esvazia-se a lista temporária.

1. **P3 (Complexo)**:
    - Enfermeiros das categorias 2 ou 3.
    - Cria uma lista temporária para os enfermeiros desse período.
    - Ordenamos os enfermeiros para o procedimento em questão por ordem crescente, ficando desta forma: [E5, E8, E9, E10, E7, E6]
    - Seleciona-se 3 enfermeiros que ainda não têm alocações no Período 2 e adiciona-se à lista temporária:

        - E5, E8, E9
    - Lista temporária: [E5, E8, E9]
2. **P4 (Regular)**:
    - Enfermeiros de qualquer categoria.
    - Ordenamos os enfermeiros para o procedimento em questão por ordem crescente, ficando desta forma: [E1, E4, E5, E8, E7, E10, E2, E6, E3, E9]
    - Seleciona-se 3 enfermeiros que ainda não têm alocações no Período 2 e adiciona-se à lista temporária:
        - E1, E4, E7
    - Lista temporária: [E5, E8, E9, E1, E4, E7]

Período 3:

- Esvazia-se a lista temporária.

1. **P5 (Regular)**:
    - Enfermeiros de qualquer categoria.
    - Cria uma lista temporária para os enfermeiros desse período.
    - Ordenamos os enfermeiros para o procedimento em questão por ordem crescente, ficando desta forma: [E1, E3, E6, E2, E5, E7, E10, E4, E9, E8]
    - Seleciona-se 3 enfermeiros que ainda não têm alocações no Período 3 e adiciona-se à lista temporária:
        - E1, E3, E6
    - Lista temporária: [E1, E3, E6]
2. **P6 (Regular)**:
    - Enfermeiros de qualquer categoria.
    - Ordenamos os enfermeiros para o procedimento em questão por ordem crescente, ficando desta forma: [E10, E6, E5, E8, E9, E3, E2, E1, E7, E4]
    - Seleciona-se 3 enfermeiros que ainda não têm alocações no Período 3 e adiciona-se à lista temporária:
        - E10, E5, E8
    - Lista temporária: [E1, E3, E6, E10, E5, E8]

Período 4:

- Esvazia-se a lista temporária.

1. **P7 (Complexo)**:
    - Enfermeiros das categorias 2 ou 3.
    - Cria uma lista temporária para os enfermeiros desse período.
    - Ordenamos os enfermeiros para o procedimento em questão por ordem crescente, ficando desta forma: [E8, E7, E9, E6, E10, E5]
    - Seleciona-se 3 enfermeiros que ainda não têm alocações no Período 4 e adiciona-se à lista temporária:
        - E8, E7, E9
    - Lista temporária: [E8, E7, E9]
2. **P8 (Complexo)**:
    - Enfermeiros das categorias 2 ou 3.

- Ordenamos os enfermeiros para o procedimento em questão por ordem crescente, ficando desta forma: [E8, E10, E7, E6, E9, E5]
- Seleciona-se 3 enfermeiros que ainda não têm alocações no Período 4 e adiciona-se à lista temporária:
  - E10, E6, E5
- Lista temporária: [E8, E7, E9, E10, E6, E5]

Período 5:

- Esvazia-se a lista temporária.

1. **P9 (Regular)**:
   - Enfermeiros de qualquer categoria.
   - Cria uma lista temporária para os enfermeiros desse período.
   - Ordenamos os enfermeiros para o procedimento em questão por ordem crescente, ficando desta forma: [E7, E5, E10, E4, E3, E9, E6, E1, E2, E8]
   - Seleciona-se 3 enfermeiros que ainda não têm alocações no Período 5 e adiciona-se à lista temporária:
     - E7, E5, E10
   - Lista temporária: [E7, E5, E10]
2. **P10 (Complexo)**:
   - Enfermeiros das categorias 2 ou 3.
   - Ordenamos os enfermeiros para o procedimento em questão por ordem crescente, ficando desta forma: [E6, E5, E8, E9, E7, E10]
   - Seleciona-se 3 enfermeiros que ainda não têm alocações no Período 5 e adiciona-se à lista temporária:
     - E6, E8, E9
   - Lista temporária: [E7, E5, E10, E6, E8, E9]

Período 6:

- Esvazia-se a lista temporária.

1. **P11 (Regular)**:
   - Enfermeiros de qualquer categoria.
   - Cria uma lista temporária para os enfermeiros desse período.
   - Ordenamos os enfermeiros para o procedimento em questão por ordem crescente, ficando desta forma: [E8, E10, E5, E9, E4, E3, E1, E7, E2, E6]
   - Seleciona-se 3 enfermeiros que ainda não têm alocações no Período 6 e adiciona-se à lista temporária:
     - Como o E8 já tem 5 participações no seu dia de trabalho, saltamos este elemento.
     - [E10, E5, E9]
   - Lista temporária: [E10, E5, E9]
2. **P12 (Complexo)**:
   - Enfermeiros das categorias 2 ou 3.
   - Ordenamos os enfermeiros para o procedimento em questão por ordem crescente, ficando desta forma: [E7, E6, E5, E10, E9, E8]

- Seleciona-se 3 enfermeiros que ainda não têm alocações no Período 6 e adiciona-se à lista temporária:
  - Como não é possível atribuir três enfermeiros das categorias 2 ou 3 sem violar as restrições fazemos uma permutação do E8 com um enfermeiro aleatório da categoria 1 que não esteja neste período e que não tenha excedido as 5 participações diárias. Assim atribui-se E8 a este procedimento, e o E8 do procedimento 2, do período 1, passa a ser E2 (por exemplo):
    1. A lista de enfermeiros do período 1 passa a ser [E3, E6, E7, ~~E8~~ **E2**, E4, E10]
    2. A lista de enfermeiros do período 6 passa a ser [E10, E5, E9, E7, E6, E8]

Período 7:

- Esvazia-se a lista temporária.

1. **P13 (Regular)**:
   - Enfermeiros de qualquer categoria.
   - Cria uma lista temporária para os enfermeiros desse período.
   - Ordenamos os enfermeiros para o procedimento em questão por ordem crescente, ficando desta forma: [E1, E8, E2, E3, E10, E5, E7, E6, E4, E9]
   - Seleciona-se 3 enfermeiros que ainda não têm alocações no Período 7 e adiciona-se à lista temporária:
     - E8 não tem espaço porque já tem 5 participações.
     - Lista temporária: [E1, E2, E3]
2. **P14 (Regular)**:
   - Enfermeiros de qualquer categoria.
   - Ordenamos os enfermeiros para o procedimento em questão por ordem crescente, ficando desta forma: [E4, E3, E9, E5, E2, E6, E10, E1, E8, E7]
   - Seleciona-se 3 enfermeiros que ainda não têm alocações no Período 7 e adiciona-se à lista temporária:
     - Como só o E4 e E9 é que têm espaço para poder fazer o procedimento realizamos uma permutação:
       1. A lista de enfermeiros do período 1 passa a ser [E3, ~~E6~~ **E1**, E7, E2, E4, E10] e o E6 passa a ter espaço.
       2. A lista de enfermeiros do período 6 passa a ser [E1, E2, E3, E4, E9, E6].

Resultados

alocacao =

1: [('P1', [E3, E1, E7]), ('P2', [E2, E4, E10])],
2: [('P3', [E5, E8, E9]), ('P4', [E1, E4, E7])],
3: [('P5', [E1, E3, E6]), ('P6', [E10, E5, E8])],
4: [('P7', [E8, E7, E9]), ('P8', [E10, E6, E5])],
5: [('P9', [E7, E5, E10]), ('P10', [E6, E8, E9])],

6: [('P11', [E10, E5, E9]), ('P12', [E7, E6, E8])],
7: [('P13', [E1, E2, E3]), ('P14', [E4, E9, E6])]

enfermeiros =

'E1': {'categoria': 1, 'procedimentos': 4 , 'periodos': [2, 3, 7, 1]},
'E2': {'categoria': 1, 'procedimentos': 2, 'periodos': [1, 7]},
'E3': {'categoria': 1, 'procedimentos': 3 , 'periodos': [1, 3, 7]},
'E4': {'categoria': 1, 'procedimentos': 3 , 'periodos': [1, 2, 7]},
'E5': {'categoria': 2, 'procedimentos': 5 , 'periodos': [2, 3, 4, 5, 6]},
'E6': {'categoria': 2, 'procedimentos': 5, 'periodos': [3, 4, 5, 6, 7]},
'E7': {'categoria': 2, 'procedimentos': 5, 'periodos': [1, 2, 4, 5, 6]},
'E8': {'categoria': 2, 'procedimentos': 5, 'periodos': [2, 3, 4, 5, 6]},
'E9': {'categoria': 3, 'procedimentos': 5, 'periodos': [2, 4, 5, 6, 7]},
'E10': {'categoria': 3, 'procedimentos': 5, 'periodos': [1, 3, 4, 5, 6]}

Tempo por período, por ordem crescente em minutos: 79, 72, 68, 86, 56, 81, 61

Tempo total: 503 minutos

Com a solução admissível apresentada em cima temos uma alocação de enfermeiros que totaliza 503 minutos, respeitando todas as restrições destacadas na alínea a).

c) Defina um cromossoma que permita codificar uma afetação dos enfermeiros aos procedimentos

Comecemos por definir um cromossoma, que se trata de uma solução pertencente à população, no contexto de um problema específico. No âmbito do problema aqui apresentado, podemos definir um **cromossoma** como uma **sequência de genes**, onde cada **gene** representa **um enfermeiro**, e a **posição** em que se encontra traduz o procedimento em questão (note-se que os **alelos** representam o **tempo** que cada enfermeiro necessita para executar um determinado procedimento).

Como em cada período há necessariamente 6 enfermeiros (pelo facto de existirem 2 procedimentos por período, onde cada um requer 3 enfermeiros) e no total um dia de trabalho tem 7 períodos, vamos ter um **cromossoma de tamanho 42 (=6*7)** ordenado, onde de 3 em 3 genes se têm os enfermeiros alocados a um determinado procedimento (um por um) e de 6 em 6 genes se têm os enfermeiros alocados a um período específico (um por um). Esta representação permite facilmente aplicar operadores genéticos, como crossover e mutação, para explorar e encontrar soluções melhores durante a otimização.

d) Tendo em conta a codificação que sugeriu em c), proponha um operador de crossover e exemplifique-o no contexto do problema da clínica

O operador de **crossover**, inspirado no processo reprodutivo biológico, consiste em **combinar informações** de **dois cromossomas pais** para gerar **dois novos cromossomas**

filhos. A operação de crossover é aplicada em algoritmos genéticos com alta probabilidade, denominada probabilidade de crossover. Existem três tipos principais de crossover que podem ser utilizados (entre os mais utilizados):

- Crossover **a um Ponto**
- Crossover a **k=2 Pontos**
- Crossover **Uniforme**

Neste trabalho, optámos por aplicar o **Crossover a um Ponto**, pela sua simplicidade de implementação e eficiência. De seguida, e antes de passarmos ao exemplo prático, explicamos como este operador será aplicado no contexto da alocação dos enfermeiros aos procedimentos, no contexto do problema da clínica.

1. Definir **aleatoriamente** um **ponto de corte**:
    a. Selecionamos um **ponto de corte aleatório** (m), que poderá **variar** entre **1** e a **dimensão do cromossoma**. Por outras palavras, temos que 1 <= m <= 42, já que no nosso caso, o tamanho do cromossoma é 42, representando a alocação de 6 enfermeiros em cada um dos 7 períodos do dia.
2. **Dividir** os cromossomas dos **pais** em **duas partes**:
    a. Os **cromossomas pai**, aos quais chamaremos CP1 e CP2, são **divididos** em **duas caudas** com base no ponto m, definido no passo anterior, tendo-se assim:
        i. **Cauda esquerda**: Contém os **genes de 1 a m**.
        ii. **Cauda direita**: Contém os **genes m+1 a 42**.
3. Criar os **filhos** a partir das **caudas** dos **pais**:
    a. **Cada filho recebe uma das caudas dos cromossomas pai**, tendo-se assim:
        i. **Filho 1**: Combina a **cauda esquerda** de **CP1** com a **cauda direita** de **CP2**.
        ii. **Filho 2**: Combina a **cauda esquerda de CP2** com a **cauda direita de CP1**.

Vejamos na prática: considerando as restrições definidas na alínea a) e a definição de cromossoma que se considerou na alínea imediatamente anterior a esta, c), vamos definir os cromossomas pai, CP1 e CP2, de forma a que sejam soluções admissíveis. Assim, temos que:

CP1: E1 | E2 | E5 | E3 | E4 | E9 | E9 | E6 | E7 | E1 | E8 | E10 | E4 | E2 | E3 | E6 | E8 | E10 | E5 | E8 | E9 | E6 | E7 | E10 | E5 | E3 | E4 | E8 | E9 | E10 | E1 | E4 | E3 | E5 | E6 | E7 | E4 | E7 | E10 | E5 | E6 | E8

CP2: E5 | E6 | E8 | E3 | E4 | E9 | E9 | E6 | E7 | E1 | E8 | E10 | E4 | E2 | E3 | E6 | E8 | E10 | E5 | E8 | E9 | E6 | E7 | E10 | E5 | E3 | E4 | E8 | E9 | E10 | E1 | E4 | E3 | E5 | E6 | E7 | E4 | E7 | E10 | E1 | E2 | E5

Suponhamos que o ponto de corte (gerado de forma aleatória) é m = 18. Assim, vamos ter a seguinte divisão:

CP1: E1 | E2 | E5 | E3 | E4 | E9 | E9 | E6 | E7 | E1 | E8 | E10 | E4 | E2 | E3 | E6 | E8 | E10 | E5 | E8 | E9 | E6 | E7 | E10 | E5 | E3 | E4 | E8 | E9 | E10 | E1 | E4 | E3 | E5 | E6 | E7 | E4 | E7 | E10 | E5 | E6 | E8

CP2: E5 | E6 | E8 | E3 | E4 | E9 | E9 | E6 | E7 | E1 | E8 | E10 | E4 | E2 | E3 | E6 | E8 | E10 | E5 | E8 | E9 | E6 | E7 | E10 | E5 | E3 | E4 | E8 | E9 | E10 | E1 | E4 | E3 | E5 | E6 | E7 | E4 | E7 | E10 | E1 | E2 | E5

Passando à criação dos filhos, CF1 e CF2, temos que o primeiro filho (CF1) combina a cauda esquerda de CP1 com a cauda direita de CP2; e o segundo filho (CF2) combina a cauda esquerda de CP2 com a cauda direita de CP1. Os cromossomas filhos apresentam as seguintes configurações:

CF1: E1 | E2 | E5 | E3 | E4 | E9 | **E9 | E6 | E7** | E1 | E8 | E10 | E4 | E2 | E3 | E6 | E8 | E10 | **E5 | E8 | E9 | E6 | E7 | E10** | E5 | E3 | E4 | **E8 | E9 | E10** | E1 | E4 | E3 | **E5 | E6 | E7** | E4 | E7 | E10 | E1 | E2 | E5

CF2: E5 | E6 | E8 | E3 | E4 | E9 | **E9 | E6 | E7** | E1 | E8 | E10 | E4 | E2 | E3 | E6 | E8 | E10 | **E5 | E8 | E9 | E6 | E7 | E10** | E5 | E3 | E4 | **E8 | E9 | E10** | E1 | E4 | E3 | **E5 | E6 | E7** | E4 | E7 | E10 | E5 | E6 | E8

É importante reforçar que a aplicação do operador de crossover pode gerar soluções não admissíveis para um problema. Nesse sentido, é fundamental verificar se CF1 e CF2 obedecem às restrições do problema, estabelecidas na alínea a). Para a restrição associada aos procedimentos complexos necessitarem de enfermeiros das categorias 2 ou 3 (note-se que os procedimentos complexos estão assinalados a negrito na configuração de CF1 e CF2 acima) vemos que todos os enfermeiros são das categorias 2 ou 3. Sobre os limites de participação dos enfermeiros, podemos ver a informação sumarizada na Tabela 1.

Enfermeiros	Participações (CF1)	Participações (CF2)	Mesmo período? (CF1)	Mesmo período? (CF2)
E1	4	2	Não	Não
E2	3	1	Não	Não
E3	4	4	Não	Não
E4	5	5	Não	Não
E5	4	5	Não	Não
E6	4	6	Não	Não
E7	4	4	Não	Não
E8	4	6	Não	Não
E9	4	4	Não	Não

E10	5	5	Não	Não

Tabela 1 - Verificação das restrições associadas aos limites de participação dos enfermeiros

Podemos concluir assim que o CF1 é uma solução admissível já que respeita todas as restrições do problema, enquanto que o CF2 **não** é uma solução admissível por violar a restrição associada ao limite máximo de participações de um enfermeiros (para E6 e E8, conforme se pode ver na Tabela 1).

e) Tendo em conta a codificação que sugeriu em c), proponha um operador de mutação e exemplifique-o no contexto do problema da clínica

A **mutação** é uma pequena **alteração aleatória** no cromossoma que gera uma **nova solução**. O operador de mutação é essencial para **manter a diversidade** na população de soluções. Esta operação é aplicada com uma **baixa probabilidade**, conhecida como **probabilidade de mutação**. Se a probabilidade de mutação for muito **alta**, o algoritmo genético reduzir-se-ia a uma mera **pesquisa aleatória**.

Existem três tipos principais de operadores de mutação (entre os mais utilizados):

- Mutação **Bit Flip**
- Mutação por **Troca**
- Mutação por **Inversão**

No âmbito deste trabalho, optou-se por utilizar a **Mutação por Troca**, por ser adequada para algoritmos genéticos baseados em **permutações**, como é o exemplo da clínica. O procedimento para aplicar este operador é o seguinte:

1. Gerar um **número aleatório uniformemente distribuído entre 0 e 1** (denotado como u).
2. Se o número **u** for **menor** que a **probabilidade de mutação** (pm), selecionamos **aleatoriamente dois genes** e **trocamos** os **valores** dos seus **alelos**.

Aplicamos este operador aos cromossomas CF1 e CF2, gerados na alínea imediatamente anterior a esta, d).

Suponhamos que tenha sido obtido o valor aleatório u=0,03 (poderia ter sido qualquer valor desde que gerado aleatoriamente numa distribuição uniforme no intervalo fechado [0, 1]), e que pm=0,1. Os genes selecionados para troca (gerados aleatoriamente) são destacados abaixo para maior clareza:

CF1 inicial: E1 | E2 | **E5** | E3 | E4 | **E9** | E9 | E6 | E7 | E1 | E8 | E10 | E4 | E2 | E3 | E6 | E8 | E10 | E5 | E8 | E9 | E6 | E7 | E10 | E5 | E3 | E4 | E8 | E9 | E10 | E1 | E4 | E3 | E5 | E6 | E7 | E4 | E7 | E10 | E1 | E2 | E5

CF1 após a mutação por troca: E1 | E2 | **E9** | E3 | E4 | **E5** | E9 | E6 | E7 | E1 | E8 | E10 | E4 | E2 | E3 | E6 | E8 | E10 | E5 | E8 | E9 | E6 | E7 | E10 | E5 | E3 | E4 | E8 | E9 | E10 | E1 | E4 | E3 | E5 | E6 | E7 | E4 | E7 | E10 | E1 | E2 | E5

Na mutação do cromossoma CF1, os valores dos alelos correspondentes aos enfermeiros E5 e E9 são trocados, resultando no enfermeiro E9 passar a ocupar a posição 3 (e consequentemente o primeiro procedimento) e o enfermeiro E5 passar a ocupar a posição 6 (e consequentemente o segundo procedimento). O cromossoma CF1 com a Mutação por Troca continua a ser uma solução admissível, já que todas as restrições continuam a ser respeitadas: cada enfermeiro participa no máximo em 5 procedimentos, não há enfermeiros duplicados nos mesmos períodos, e os procedimentos complexos continuam a ser realizados por enfermeiros das categorias 2 ou 3. Registou-se apenas uma troca no primeiro período.

CF2 inicial: E5 | E6 | **E8** | E3 | E4 | **E9** | E9 | E6 | E7 | E1 | E8 | E10 | E4 | E2 | E3 | E6 | E8 | E10 | E5 | E8 | E9 | E6 | E7 | E10 | E5 | E3 | E4 | E8 | E9 | E10 | E1 | E4 | E3 | E5 | E6 | E7 | E4 | E7 | E10 | E5 | E6 | E8

CF2 após a mutação por troca: E5 | E6 | **E9** | E3 | E4 | **E8** | E9 | E6 | E7 | E1 | E8 | E10 | E4 | E2 | E3 | E6 | E8 | E10 | E5 | E8 | E9 | E6 | E7 | E10 | E5 | E3 | E4 | E8 | E9 | E10 | E1 | E4 | E3 | E5 | E6 | E7 | E4 | E7 | E10 | E5 | E6 | E8

Na mutação do cromossoma CF2, os valores dos alelos correspondentes aos enfermeiros E8 e E9 são trocados, resultando no enfermeiro E9 passar a ocupar a posição 3 (e consequentemente o primeiro procedimento) e o enfermeiro E8 passar a ocupar a posição 6 (e consequentemente o segundo procedimento). O cromossoma CF2 com a Mutação por Troca continua a ser uma solução **não** admissível, já que continuam a existir 6 participações por parte dos enfermeiros E6 e E8. Note-se que com esta mutação se registou apenas uma troca de enfermeiros no primeiro período.

f) Os operadores propostos em d) e e) garantem a obtenção de soluções admissíveis para o problema da clínica? Caso não garantam, justifique que tipos de inadmissibilidades os operadores indicados podem gerar e indique como pode ultrapassar esta situação na implementação do algoritmo genético

Para que uma solução seja **admissível** é necessário que **cumpra todos os requisitos** estabelecidos na alínea a). Os operadores propostos em d) e e) **podem gerar soluções admissíveis**, mas **não garantem** que tal aconteça **em todos os casos**.

O operador de **crossover** proposto em **d)**, **Crossover a um Ponto**, tem como objetivo combinar informações de dois cromossomas pais para gerar dois novos cromossomas filhos. No entanto, **não há garantia** de que **as soluções filhas respeitem todas as restrições** do problema. No nosso caso, inclusivé, **CF1** apresentou-se como uma **solução admissível**,

enquanto que **CF2** se apresentou como uma **solução não admissível** por violar a restrição relacionada com o número máximo de cinco participações de um enfermeiro (para os enfermeiros E6 e E8).

Quanto ao operador de **mutação** proposto em **e)**, **Mutação por Troca**, introduz pequenas alterações aleatórias nos cromossomas, para explorar diferentes regiões do espaço de pesquisa. No entanto, também **não há garantia** de que as soluções que daqui resultam sejam todas admissíveis. No exemplo apresentado em e) CF1 continuou a ser uma solução admissível após a mutação, mas CF2 continuou a ser uma solução não admissível por continuar a violar a restrição associada ao número máximo de participações de um enfermeiro (para os enfermeiros E6 e E8).

Desta forma, as inadmissibilidades podem ocorrer de várias maneiras:

1. **Violação dos limites de participação dos enfermeiros**: como observado no exemplo com CF2, os operadores podem gerar soluções onde um ou mais enfermeiros excedem o número máximo de participações em procedimentos permitidos no seu dia de trabalho.
2. **Duplicação de enfermeiros nos mesmos períodos**: Os operadores podem inadvertidamente alocar o mesmo enfermeiro para dois procedimentos no mesmo período, violando a restrição de que cada enfermeiro deve ser alocado a apenas um procedimento por período.
3. **Procedimentos complexos atribuídos a enfermeiros de categoria 1**: podem-se gerar soluções onde enfermeiros da categoria 1 são atribuídos a esses procedimentos (embora tal não tenha acontecido nos exemplos ilustrados nas alíneas anteriores).

Para ultrapassar as inadmissibilidades e garantir que as soluções geradas pelos operadores sejam admissíveis devem-se **adicionar mecanismos de verificação** e **utilizar procedimentos para reparar a não admissibilidade**, **trocando o valor** de **alguns genes**. Os pontos abaixo sumarizam as correções que são necessárias implementar para corrigir a não admissibilidade:

1. Verifica-se se algum **enfermeiro excedeu** o **número máximo de participações permitidas** em procedimentos (5). Se um enfermeiro exceder esse limite, aplica-se uma correção **trocando** esse enfermeiro **por outro aleatório** que ainda **não tenha atingido o limite máximo** de participações e que **não tenha sido alocado ao outro procedimento que decorre no mesmo período**.
2. Durante a aplicação do operador de crossover e mutação, incluem-se **mecanismos de verificação** para garantir que **nenhum enfermeiro** seja alocado a **mais de um procedimento** no **mesmo período**. Se essa duplicação ocorrer, **trocamos o enfermeiro** por **outro enfermeiro aleatório** que ainda **não** tenha sido **alocado** naquele **período** e que ainda **não** tenha **excedido** o **limite máximo** de participações.
3. Verifica-se se os **procedimentos complexos** foram atribuídos a **enfermeiros da categoria 1**. Se sim, substitui-se **aleatoriamente** os **enfermeiros de categoria 1** por **enfermeiros das categorias 2 ou 3** que ainda **não** tenham sido **alocados** ao **período** em questão e que ainda **não** tenham **excedido** o **limite máximo** de participações.

g) Na tentativa de determinar uma solução admissível de qualidade, a clínica irá definir e implementar um algoritmo genético. Tendo em conta as alíneas anteriores e sabendo que a clínica deseja uma afetação dos procedimentos aos enfermeiros que não ultrapasse as 8 horas, apresente o pseudocódigo para a aplicação do algoritmo

Para resolver o problema da alocação de enfermeiros aos procedimentos, a clínica optou por utilizar um **algoritmo genético**. A ideia passará por gerar uma **população inicial de cromossomas aleatórios**, representando **diferentes alocações** de enfermeiros aos procedimentos (a estrutura dos cromossomas é a que aparece descrita na alínea c)). Cada cromossoma é **avaliado** quanto à sua **adequação**, medindo a **soma das durações dos períodos de trabalho** dos enfermeiros. Se um cromossoma **não** for **admissível** de acordo com as restrições impostas na alínea a), aplica-se um **processo de correção** para corrigi-lo. Durante as iterações do algoritmo genético, selecionamos os **cromossomas mais adequados** para **reprodução**, realizamos **crossover** e **mutação** para gerar **novas soluções**, e avaliamos se houve **melhorias**. Se a solução **não melhorar** ao fim de um número especificado de **iterações**, **paramos** o processo. Este ciclo **continua** até que uma **solução satisfatória** seja encontrada (neste caso **inferior a 480 minutos** que correspondem a 8 horas) ou até que o **número máximo de iterações** seja atingido. No final, retornamos a **melhor solução encontrada**. Este processo visa encontrar uma **alocação de enfermeiros** que **minimize a duração total do dia de trabalho**, **respeitando** todas as **restrições** estabelecidas pela clínica. O pseudocódigo pode ver-se abaixo.

Pseudocódigo

```
1. Inicialização:
  a. Definir os parâmetros do algoritmo:
    - Tamanho da população (POP_SIZE)
    - Probabilidade de crossover (PC)
    - Probabilidade de mutação (PM)
    - Número máximo de gerações (MAX_GEN)
    - Limite de tempo máximo (MAX_TIME)
    - Número máximo de iterações (MAX_ITER)
    - Número máximo de iterações sem melhoria (MAX_ITER_NO_IMPROVEMENT)
  b. Gerar uma população inicial de cromossomas aleatórios.
    - Para cada cromossoma na população:
      - Gerar 7 períodos com 6 enfermeiros aleatórios cada.
      - Se o cromossoma não for admissível, corrigir.

2. Avaliação:
  Função `evaluate(chromosome)`:
    a. Calcular a aptidão de um cromossoma:
      - Calcular a soma das durações dos 7 períodos.
      - Quanto menor a soma das durações, melhor a aptidão.

3. Seleção:
  Função `selection(population, fitnesses)`:
    a. Selecionar cromossomas para reprodução usando seleção por torneio:
      - Realizar torneios entre pares de cromossomas e selecionar os vencedores.

4. Crossover:
  Função `crossover(parent1, parent2)`:
```

a. Realizar crossover entre dois cromossomas:
- Gerar um número aleatório u.
- Se u < PC, aplicar o crossover a um ponto para gerar dois filhos.
- Caso contrário, os filhos são cópias dos pais.
- Verificar se os filhos são admissíveis, caso contrário, corrigir.

5. Mutação:
Função `mutate(chromosome)`:
a. Aplicar mutação a um cromossoma:
- Gerar um número aleatório v.
- Se v < PM, aplicar a mutação por troca em dois genes aleatórios.
- Verificar se o filho mutado é admissível, caso contrário, corrigir.

6. Verificação de Admissibilidade:
Função `is_admissible(chromosome)`:
a. Verificar se um cromossoma é admissível:
- Verificar limites de participação dos enfermeiros.
- Verificar enfermeiros alocados a apenas um procedimento por período.
- Verificar se procedimentos complexos são realizados apenas por enfermeiros das categorias 2 e 3.

7. Correção de Cromossomas:
Função `repair(chromosome)`:
a. Corrigir um cromossoma não admissível:
i. Corrigir duplicações de enfermeiros num mesmo período:
- Para cada período no cromossoma:
- Identificar e substituir enfermeiros duplicados por enfermeiros aleatórios com disponibilidade.

ii. Garantir que enfermeiros não ultrapassem o limite de participação:
- Verificar a contagem de participação de cada enfermeiro.
- Se algum enfermeiro ultrapassar o limite, substituí-lo por outro enfermeiro aleatório entre os disponíveis e que não tenham passado esse limite.

iii. Substituir enfermeiros de categoria 1 em procedimentos complexos:
- Para cada procedimento complexo no cromossoma (acedido através dos índices):
- Verificar se o enfermeiro designado é da categoria 1.
- Se for, substituir por um enfermeiro de categoria diferente, selecionado aleatoriamente.

b. Se o cromossoma ainda não for admissível após o repair, repetir o processo até que seja admissível.

8. Algoritmo Genético:
Função `genetic_algorithm()`:
a. Gerar a população inicial.
b. Inicializar a melhor solução (`best_solution`) e a melhor aptidão (`best_fitness`) como infinito.
c. Inicializar o contador de iterações sem melhoria (`iter_without_improvement`) como 0.

d. Para cada geração até `MAX_GEN`:
i. Avaliar a população atual.
ii. Atualizar a melhor solução se uma solução melhor for encontrada.
iii. Incrementar o contador de iterações sem melhoria se não houver melhoria na melhor solução.

iv. Verificar condições de paragem:
- Interromper se a melhor aptidão for menor ou igual ao tempo máximo (`MAX_TIME`).
- Interromper se o número máximo de iterações (`MAX_ITER`) for atingido.
- Interromper se não houver melhoria por `MAX_ITER_NO_IMPROVEMENT` iterações consecutivas.

v. Realizar seleção, crossover e mutação para gerar a próxima população.
vi. Substituir a população atual pela nova população.

e. Retornar a melhor solução encontrada.

h) Implemente o procedimento definido na alínea g)

Primeiro começou-se por definir os parâmetros fundamentais para a execução do algoritmo genético, incluindo o tamanho da população (definido após alguma experimentação), as probabilidades de crossover e mutação, além dos limites de gerações, tempo e iterações, fundamentais para as condições de paragem do algoritmo. Além disso, são estabelecidas as categorias dos enfermeiros e os índices dos procedimentos complexos para que seja depois possível fazer a verificação da admissibilidade das soluções. A duração dos procedimentos foi carregada a partir do Excel disponibilizado no Moodle, e converteu-se num numpy array para facilitar a manipulação dos dados durante a execução do algoritmo.

```python
# Parâmetros do algoritmo
POP_SIZE = 200
PC = 0.8  # Probabilidade de crossover
PM = 0.1  # Probabilidade de mutação
MAX_GEN = 1000
MAX_TIME = 480  # 8 horas = 480 minutos
MAX_ITER = 100  # Número máximo de iterações
MAX_ITER_NO_IMPROVEMENT = 15  # Limite de iterações sem melhoria

# Categorias dos enfermeiros (1: categoria 1, 2: categoria 2, 3: categoria 3)
CATEGORIES = {
    "E1": 1, "E2": 1, "E3": 1, "E4": 1, "E5": 2, "E6": 2, "E7": 2, "E8": 2, "E9": 3, "E10": 3
}

# Procedimentos complexos (índices dos procedimentos complexos no cromossoma)
COMPLEX_PROCEDURES = [6,7,8,18,19,20,21,22,23,27,28,29,33,34,35]

# Duração dos procedimentos
durations = pd.read_excel('Trab_Grupo.xlsx').to_numpy()

# Converte enfermeiros para índices para fácil manipulação
nurses = list(CATEGORIES.keys())
nurse_to_idx = {nurse: idx for idx, nurse in enumerate(nurses)}
```

Figura 1 - Inicialização do algoritmo

Na função, generate_initial_population(), uma população inicial de cromossomas foi criada com base no POP_SIZE. Para cada indivíduo na população, são gerados 7 períodos de trabalho, cada um com 6 enfermeiros selecionados aleatoriamente, sem repetição. De seguida, verificamos se o cromossoma gerado é admissível, ou seja, se atende a todas as restrições do problema. Se não for admissível, a função repair() é chamada para corrigi-lo. O cromossoma é depois adicionado à população.

```python
def generate_initial_population():
    population = []
    for _ in range(POP_SIZE):
        chromosome = []
        for _ in range(7):  # 7 períodos
            period = random.sample(nurses, 6)  # 6 enfermeiros por período, sem repetição
            chromosome.extend(period)
        if not is_admissible(chromosome):
            chromosome = repair(chromosome)
        population.append(chromosome)
    return population
```

Figura 2 - Geração da população inicial

Já na função evaluate(chromosome), é calculada a duração total de um cromossoma, que é dividido em períodos, e para cada período, são identificados os procedimentos

atribuídos e as suas durações (do excel obtido do Moodle). A duração máxima entre os procedimentos de cada período é calculada e adicionada à duração total do cromossoma.

```python
def evaluate(chromosome):
    total_duration = 0
    for period in range(7):
        period_procedures = chromosome[period * 6:(period + 1) * 6]
        if period >= 1:
            period = period + period
        proc1_durations = [durations[period][int(s[1:])-1] for s in period_procedures[:3]]
        proc2_durations = [durations[period+1][int(s[1:])-1] for s in period_procedures[3:]]
        if period > 1:
            period = period - 1
        max_duration = max(max(proc1_durations), max(proc2_durations))
        total_duration += max_duration
    return total_duration
```

Figura 3 - Cálculo da aptidão de cada cromossoma

A função selection(population, fitnesses) realiza a seleção de cromossomas para reprodução utilizando o método do torneio. Para cada iteração, selecionam-se dois cromossomas aleatoriamente da população e comparam-se com base nas suas aptidões (fitnesses). O cromossoma com a menor aptidão (ou seja, o vencedor do torneio por ter o menor tempo de trabalho) é adicionado à lista de cromossomas selecionados. Este processo é repetido até que a metade da população seja selecionada, e a lista resultante de cromossomas selecionados seja retornada como a nova população de reprodução.

```python
def selection(population, fitnesses):
    selected = []
    for _ in range(POP_SIZE // 2):
        tournament = random.sample(range(POP_SIZE), 2)
        winner = tournament[0] if fitnesses[tournament[0]] < fitnesses[tournament[1]] else tournament[1]
        selected.append(population[winner])
    return selected
```

Figura 4 - Seleção de cromossomas para reprodução

Nesta função de crossover, estamos a realizar uma operação genética em que dois pais (cromossomas) são combinados para gerar dois filhos (novos cromossomas). A probabilidade de realizar o crossover é determinada pela taxa de crossover (PC). Se um número aleatório entre 0 e 1 for menor que a taxa de crossover, o crossover acontece. O ponto de crossover (m) é selecionado aleatoriamente entre 1 e o tamanho do cromossoma. Os genes dos pais são então combinados para criar os filhos. Após o crossover, verificamos se os filhos são admissíveis. Se não forem, são reparados usando a função de repair(). Depois disso, os filhos são devolvidos. Se o crossover não ocorrer, os pais são devolvidos diretamente, após verificação e, se necessário, correção.

```python
def crossover(parent1, parent2):
    if random.random() < PC:
        m = random.randint(1, len(parent1) - 1)
        child1 = parent1[:m] + parent2[m:]
        child2 = parent2[:m] + parent1[m:]
        if not is_admissible(child1):
            child1 = repair(child1)  # Corrigir possíveis duplicados
        if not is_admissible(child2):
            child2 = repair(child2)  # Corrigir possíveis duplicados
        return [child1, child2]
    else:
        if not is_admissible(parent1):
            parent1 = repair(parent1)
        if not is_admissible(parent2):
            parent2 = repair(parent2)
        return [parent1, parent2]
```

Figura 5 - Operador de crossover definido

Já com a função de mutação, procuramos introduzir variação genética no cromossoma. A mutação ocorre com uma probabilidade determinada pela taxa de mutação (PM). Se um número aleatório entre 0 e 1 for menor que a taxa de mutação, selecionamos dois genes aleatórios no cromossoma e trocamos as suas posições. Após a mutação, verificamos se o cromossoma mutado é admissível no contexto do problema da clínica. Se não for, corrigimos o cromossoma usando a função de repair().

```python
def mutate(chromosome):
    mutated_chromosome = chromosome[:]  # Cria uma cópia do cromossoma original
    if random.random() < PM:
        i, j = random.sample(range(len(mutated_chromosome)), 2)
        mutated_chromosome[i], mutated_chromosome[j] = mutated_chromosome[j], mutated_chromosome[i]
    if not is_admissible(mutated_chromosome):
        mutated_chromosome = repair(mutated_chromosome)  # Corrige o cromossoma mutado
    return mutated_chromosome
```

Figura 6 - Operador de mutação definido

É com a função is_admissible() que se cerifica se um cromossoma é admissível de acordo com as restrições do problema. Primeiro, contamos quantas vezes cada enfermeiro aparece nos períodos. Se algum enfermeiro aparecer mais de 5 vezes, o cromossoma é considerado como uma solução não admissível. De seguida, verificamos se há enfermeiros duplicados dentro de cada período. Se houver, o cromossoma é considerado não admissível também. Por fim, verificamos se algum enfermeiro de categoria 1 está designado para um procedimento complexo, o que também tornaria o cromossoma não admissível. Se todas essas condições forem cumpridas, o cromossoma é considerado admissível e a função retorna verdadeiro.

```python
def is_admissible(chromosome):
    nurse_counts = {nurse: 0 for nurse in nurses}
    for period in range(7):
        period_procedures = chromosome[period * 6:(period + 1) * 6]
        for nurse in period_procedures:
            nurse_counts[nurse] += 1
            if nurse_counts[nurse] > 5:
                return False  # Enfermeiro aparece em mais de 5 períodos
        if len(period_procedures) != len(set(period_procedures)):
            return False  # Enfermeiro duplicado no mesmo período
        for procedure_idx in COMPLEX_PROCEDURES:
            nurse = chromosome[procedure_idx]
            if CATEGORIES[nurse] == 1:
                return False  # Enfermeiro de categoria 1 designado para procedimento complexo
    return True
```

Figura 7 - Função para a verificação das restrições

A função repair() corrige cromossomas que não são admissíveis, garantindo que atendem às restrições do problema, e fazendo trocas por enfermeiros aleatórios e respectivas verificações para assegurar que nova solução reparada é admissível no contexto do problema da clínica.

```python
def repair(chromosome):
    max_iterations = 100  # Define um limite máximo de iterações sem alterações
    iteration_count = 0

    while not is_admissible(chromosome):
        for period in range(7):
            period_procedures = chromosome[period * 6:(period + 1) * 6]
            unique_nurses = []
            for nurse in period_procedures:
                if nurse not in unique_nurses:
                    unique_nurses.append(nurse)
            # Preencher os períodos com enfermeiros adicionais
            while len(unique_nurses) < 6:
                available_nurses = [n for n in nurses if n not in unique_nurses]
                if not available_nurses:
                    break
                nurse_to_add = random.choice(available_nurses)
                unique_nurses.append(nurse_to_add)
            chromosome[period * 6:(period + 1) * 6] = unique_nurses

        nurse_counts = {nurse: 0 for nurse in nurses}
        for period in range(7):
            period_procedures = chromosome[period * 6:(period + 1) * 6]
            for nurse in period_procedures:
                nurse_counts[nurse] += 1

        for nurse, count in nurse_counts.items():
            if count > 5:
                for _ in range(count - 5):
                    for period in range(7):
                        if chromosome[period * 6] == nurse:
                            chromosome[period * 6] = random.choice([n for n in nurses if n != nurse])

        for procedure_idx in COMPLEX_PROCEDURES:
            nurse = chromosome[procedure_idx]
            if CATEGORIES[nurse] == 1:
                available_nurses = [n for n in nurses if CATEGORIES[n] != 1]
                chromosome[procedure_idx] = random.choice(available_nurses)

        iteration_count += 1
        if iteration_count > max_iterations:
            # Se atingir o limite de iterações sem alterações, reinicia o cromossoma
            def initialize_chromosome():
                chromosome = []
                for _ in range(7):  # 7 períodos
                    for _ in range(6):  # 6 enfermeiros por período
                        nurse = random.choice(nurses)  # Seleciona um enfermeiro aleatório
                        chromosome.append(nurse)
                return chromosome
            chromosome = initialize_chromosome()  # Reinicializar o cromossoma
            iteration_count = 0  # Reiniciar o contador de iterações

    while not is_admissible(chromosome):
        repair(chromosome)

    return chromosome
```

Figura 8 - Função para a correção de soluções não admissíveis

A função genetic_algorithm() é a responsável pela implementação do algoritmo genético para resolver o problema em questão. Começamos por gerar uma população inicial de cromossomas admissíveis. Depois, iteramos por um número máximo de gerações, avaliando a aptidão de cada cromossoma na população e atualizando a melhor solução encontrada até então. O algoritmo continua até que se verifique um dos critérios de paragem: não haver melhorias ao fim de 15 iterações, obter-se uma alocação de enfermeiros cuja duração total do dia de trabalho seja menor ou igual que as 8 horas definidas pela clínica ou até que o número máximo de iterações seja atingido. Durante cada iteração, selecionamos cromossomas para reprodução, realizamos crossover e mutação, e geramos a próxima

população. No fim, retorna-se a melhor solução encontrada e a respetiva duração dessa solução.

```python
def genetic_algorithm():
    population = generate_initial_population()
    best_solution = None
    best_fitness = float('inf')
    iter_without_improvement = 0

    for generation in range(MAX_GEN):
        fitnesses = [evaluate(chromosome) for chromosome in population]
        min_fitness = min(fitnesses)
        min_index = fitnesses.index(min_fitness)

        if min_fitness < best_fitness:
            best_fitness = min_fitness
            best_solution = population[min_index]
            iter_without_improvement = 0
        else:
            iter_without_improvement += 1

        # Se não houver melhoria por muitas iterações, interromper o algoritmo
        if iter_without_improvement >= MAX_ITER_NO_IMPROVEMENT:
            print('Algoritmo parou ao fim de ', iter_without_improvement, ' iterações sem melhorias significativas')
            break

        # Imprimir os cromossomas testados e os seus valores de duração
        print("Geração:", generation)
        for i, chromosome in enumerate(population):
            print("Cromossoma:", chromosome, "Duração:", fitnesses[i])

        if min_fitness <= MAX_TIME:
            print('Algoritmo parou por se obter ', min_fitness)
            break

        if generation >= MAX_ITER:
            print('Algoritmo parou ao fim de ', generation , ' iterações')
            break

        selected = selection(population, fitnesses)
        next_population = []

        while len(next_population) < POP_SIZE:
            parents = random.sample(selected, 2)
            offspring = crossover(parents[0], parents[1])
            for child in offspring:
                mutated_child = mutate(child)
                next_population.append(mutated_child)

        population = next_population

    return best_solution, best_fitness
```

Figura 9 - Implementação completa do algoritmo genético

i) Execute o código desenvolvido e faça uma breve análise à solução admissível obtida para o problema da clínica

Após a execução do **algoritmo genético** que foi desenvolvido, foi encontrada uma **solução** que **respeita todas as restrições** impostas pelo problema da clínica, tendo esta solução uma **duração total de 474 minutos**, que é **inferior ao limite máximo** de afetação dado pela clínica de **480 minutos** (ou seja, 8 horas).

O cromossoma obtido com a aplicação deste algoritmo genético foi ['E4', 'E3', 'E7', 'E8', 'E2', 'E6', 'E8', 'E9', 'E10', 'E4', 'E6', 'E2', 'E2', 'E3', 'E1', 'E6', 'E5', 'E10', 'E6', 'E7', 'E10', 'E9', 'E8', 'E5', 'E3', 'E10', 'E7', 'E5', 'E9', 'E8', 'E9', 'E4', 'E10', 'E5', 'E7', 'E6', 'E8', 'E2', 'E1', 'E4', 'E9', 'E5'], que indica uma alocação de enfermeiros, conforme se pode ver na tabela 2.

Período	Alocação
1	Enfermeiros 'E4', 'E3', 'E7' ficam alocados ao procedimento 1 (regular), e enfermeiros 'E8', 'E2', 'E6' ficam alocados ao procedimento 2 (regular).
2	Enfermeiros 'E8', 'E9', 'E10' ficam alocados ao procedimento 3 (complexo), e enfermeiros 'E4', 'E6', 'E2' ficam alocados ao procedimento 4 (regular).
3	Enfermeiros 'E2', 'E3', 'E1' ficam alocados ao procedimento 5 (regular), e enfermeiros 'E6', 'E5', 'E10' ficam alocados ao procedimento 6 (regular).
4	Enfermeiros 'E6', 'E7', 'E10' ficam alocados ao procedimento 7 (complexo), e enfermeiros 'E9', 'E8', 'E5' ficam alocados ao procedimento 8 (complexo).
5	Enfermeiros 'E3', 'E10', 'E7' ficam alocados ao procedimento 9 (regular), e enfermeiros 'E5', 'E9', 'E8' ficam alocados ao procedimento 10 (complexo).
6	Enfermeiros 'E9', 'E4', 'E10' ficam alocados ao procedimento 11 (regular), e enfermeiros 'E5', 'E7', 'E6' ficam alocados ao procedimento 12 (complexo).
7	Enfermeiros 'E8', 'E2', 'E1' ficam alocados ao procedimento 13 (regular), e enfermeiros 'E4', 'E9', 'E5' ficam alocados ao procedimento 14 (regular).

Tabela 2 - Alocação de enfermeiros com a solução obtida com o algoritmo genético

Importa reforçar que **nenhuma das restrições** estabelecidas na alínea a) foi **violada**, nesta solução, isto é, cada enfermeiro participa no máximo em 5 procedimentos durante o dia, nenhum dos enfermeiros participa em dois procedimentos do mesmo período e a solução garante que nenhum dos procedimentos complexos foi atribuído a enfermeiros da categoria 1. A tabela 3 sumariza a carga de procedimentos por enfermeiro.

Enfermeiro	Número de procedimentos a que foi alocado
E1	2
E2	4
E3	3
E4	4
E5	5
E6	5
E7	4
E8	5
E9	5
E10	5

Tabela 3 - Carga de procedimentos de cada enfermeiro

Em suma, com isto foi encontrada uma **solução admissível** para o **problema da clínica**. O algoritmo implementado utilizou seleção, crossover e mutação para explorar o espaço de soluções e iterativamente melhorar a população de cromossomas. Além disso, foram realizadas avaliações das soluções para garantir a conformidade com as restrições de tempo e reparações para corrigir cromossomas inválidos, assegurando que todas as restrições fossem atendidas ao longo do processo. Em relação ao parâmetro 'POP_SIZE'(tamanho da população) = 200, uma população maior promove maior diversidade genética, aumentando a probabilidade de encontrar boas soluções ao longo das gerações, no entanto implica mais custo computacional, pois mais cromossomas precisam de ser avaliados em cada geração.