Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

Problema da Distribuição Hoteleira

Autores: João Vitor Motta e Thiago Sampaio de Paiva
Orientador: Prof. Dr. Bruno César Ribas

Brasília, DF
2023

João Vitor Motta e Thiago Sampaio de Paiva

Problema da Distribuição Hoteleira

Relátorio submetido a disciplina Tópicos Especiais em Engenharia de Software do curso de graduação em Engenharia de Software da Universidade de Brasília.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Prof. Dr. Bruno César Ribas

Brasília, DF

2023

Sumário

1 Introdução

A indústria hoteleira passou por uma evolução significativa nas últimas décadas, impulsionada pelo avanço da tecnologia e pela crescente demanda dos consumidores por experiências personalizadas e convenientes. Nesse contexto, a distribuição eficiente de recursos tornou-se um dos principais desafios enfrentados pelos organizadores de evento na busca por minimizar os gastos e garantir a satisfação e conforto dos hóspedes.

Neste contexto, métodos computacionais de otimização têm desempenhado um papel fundamental na resolução desses problemas complexos. Entre esses métodos, destacam-se as funções pseudo-Booleanas, que permitem a modelagem matemática e a resolução eficiente de problemas de otimização combinatória.

1.1 Problemática

A problemática da distribuição hoteleira foi proposta inicialmente por Ribas (2023) e consiste, principalmente, em alocar um grupo fixo de pessoas em um hotel de uma maneira específica, seguindo critérios como atender as demandas específicas dos hóspedes, minimizar os gastos.

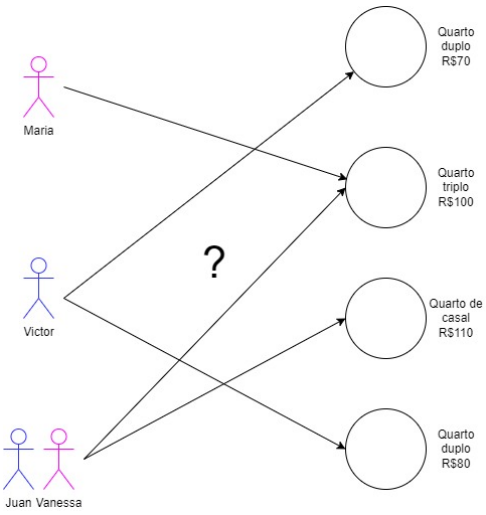
O problema é composto por dois tipos de hóspedes, sendo um único hóspede e um casal. Cada hóspede, obrigatoriamente, deve estar alocado em um único quarto. Além disso, o casal deve sempre ser alocado no mesmo quarto.

Há 4 tipos de quartos, podendo ser: Quarto de casal (C), que possui uma cama inseparável; Quarto Duplo (D), que possui duas camas de solteiro que podem ser unidas para formar uma de casal; Quarto Triplo (T), que possui três camas de solteiro e que duas podem ser unidas para formar uma de casal; Quarto Quadruplo (Q), que possui quatro camas e que duas podem se formar para se tornar uma de casal.

Sobre o grupo de pessoas, é fornecido informações como gênero, grau de afinidade entre eles e quais formam um casal. A afinidade é representada por um número entre 0 a 100 sendo 0 o menor grau de afinidade e 100 o maior. Esse numero é estabelecido de um para outro, mas reciprocamente não necessariamente é o mesmo numero. Se a informação sobre a afinidade é omitida, entende-se que 50 é a afinidade padrão caso as pessoas tenham o mesmo gênero e 0 caso possuem gêneros diferentes.

A Figura 1 apresenta um exemplo de distribuição de vários hóspedes em vários quartos. Desta maneira, cabe refletir qual seria a melhor maneira de alocação, visando minimizar o custo com o aluguel de quartos e maximizar a afinidade entre as pessoas que compartilham o mesmo quarto.

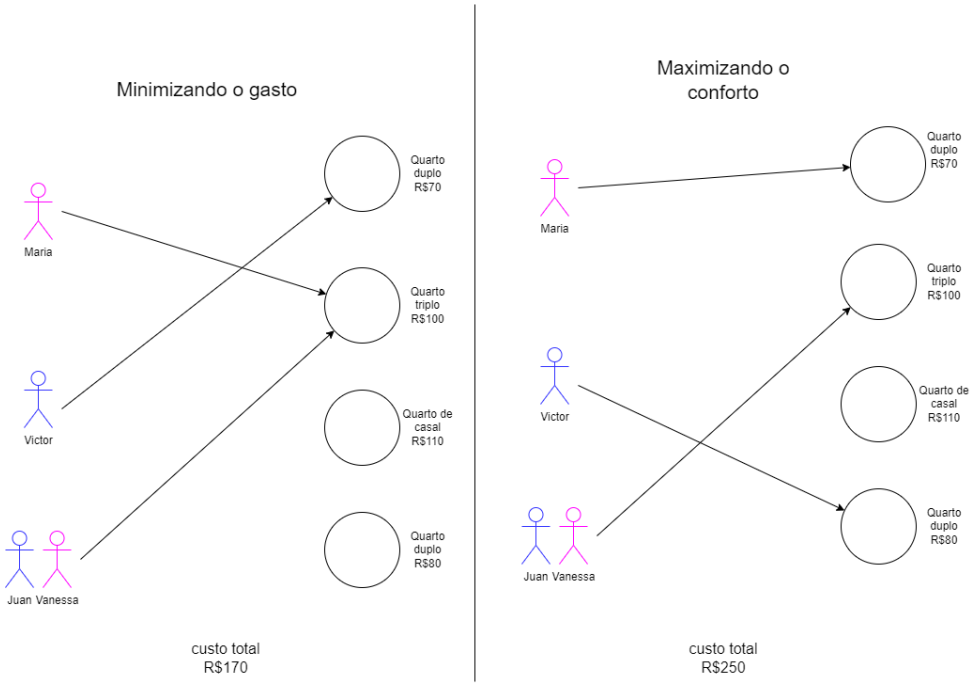
Figura 1 – Exemplo da problemática



Fonte: Autores.

Há duas possibilidades de solução, uma minimizando o custo e outra maximizando o conforto e são apresentadas na Figura 2.

Figura 2 – Exemplo da problemática resolvido



Fonte: Autores.

2 Trabalhos Correlatos

Este capitulo visa apresentar brevemente trabalhos que também usam formulações pseudo-Booleanas e otimização por restrições para encontrar uma possível solução em tempo hábil para alguns problemas estabelecidos.

2.1 Uma abordagem pseudo-Booleana para consolidação de máquinas virtuais com restrições de afinidade

Consolidação de máquinas virtuais é um problema que tenta alocar de maneira eficiente máquinas virtuais em um conjunto de hardwares, com o objetivo de minimizar a quantidade de hardware ligado (RIBAS, 2012). O objetivo da monografia do Lara (2020) é ampliar uma abordagem já existente proposta pelo Ribas (2012), que usa uma estratégia pseudo-Booleana, para que sejam contempladas regras adicionais relacionadas a conflitos de máquinas virtuais, que ocorrem quando dois ou mais tipos de máquinas disputam pelos mesmos recursos quando alocadas em mesmo hardware.

3 Formulação

A formulação proposta para resolver O Problema da Distribuição Hoteleira se baseia em cláusulas pseudo-Booleanas, que restringem a distribuição dos hóspedes nos quartos, mais uma função que minimiza o custo dos quartos alugados e maximiza a afinidade dos hóspedes que dividem o mesmo quarto.

Os solucionadores, geralmente, não realiza maximização diretamente, apenas minimização. Portanto, a afinidade entre os hóspedes foi transformada de modo que se consiga realizar a minimização, obtendo o mesmo efeito de maximização. Para isso, a afinidade entre os hóspedes foi transformada em antipatia, sendo o contrário de afinidade, por exemplo, se o hóspede A tiver 100 de afinidade com o hóspede B, então terá 0 de antipatia, já se tiver 0 de afinidade, então terá 100 de antipatia. Assim, é possível minimizar a antipatia, que terá um efeito maximizador da afinidade entre os hóspedes. A função para se obter a antipatia por meio da afinidade é mostrada na Fórmula 3.1.

$$Antipatia(afinidade) = 100 - afinidade \tag{3.1}$$

3.1 Minimização

Cada quarto do hotel tem um custo de aluguel e os hóspedes tem um grau de antipatia entre si e o problema exige que o total desses valores seja o mínimo. Assim, foi criado uma cláusula de minimização que minimiza esses valores com uma variável atrelada que pode ser ligada ou desligada. Desta forma, quanto menos variáveis dessa formula estiver ligada, menor será o custo dos quartos e maior será a afinidade entre os hóspedes.

Sendo q a quantidade total de quartos, q_i o i-ésimo quarto do hotel e $CUSTO_i$ a constante que representa o custo do quarto i, o custo total dos quartos alugados é mostrado na Fórmula 3.2. Desta forma, o valor total muda conforme a variável q_i tiver a valoração 0 ou 1.

$$\sum_{i=1}^{q} CUSTO_i \cdot q_i \tag{3.2}$$

Sendo h a quantidade de hóspedes, i e j pares do conjunto gerado pela combinação $\binom{h}{2}$ entre os hóspedes, $ANTIPATIA_{ij}$ a constante com o valor da antipatia dos hóspedes i e j e a_{ij} a variável que representa quando os hóspedes i e j estão no mesmo quarto, o total das antipatias dos hóspedes que compartilham o mesmo quarto é mostrado na

Fórmula 3.3. Desta forma, a variável a_{ij} pode ser valorada em 0 ou 1 e mudar o valor total das antipatias.

$$\sum_{i=1}^{h-1} \sum_{j=i+1}^{h} ANTIPATIA_{ij} \cdot a_{ij} \tag{3.3}$$

Assim, a cláusula de minimização é composta pela Fórmula 3.2 e 3.3 é mostrada na Fórmula 3.4

$$min : \left(\sum_{i=1}^{q} CUSTO_i \cdot q_i \right) + \left(\sum_{i=1}^{h-1} \sum_{j=i+1}^{h} ANTIPATIA_{ij} \cdot a_{ij} \right) \tag{3.4}$$

Como é possível observar na Fórmula 3.4 de minimização, acontece uma soma simples entre os custos e antipatias, sem uma normalização. Isso pode causar imprecisões quando o custos dos quartos se encontram em uma faixa próxima ao intervalo da antipatia. Deste modo, cabe como melhoria, para uma melhor precisão, a normalização com pesos, visando evidenciar a prioridade do custo sobre a antipatia.

3.2 Cláusulas

3.2.1 Base

As cláusulas-base são responsáveis por garantir que todos os hóspedes sejam alocados em um quarto e a quantidade de hóspedes em um quarto qualquer não seja maior que a capacidade máxima dele. São detalhadas a seguir.

3.2.1.1 Todos os hóspedes devem estar em somente um quarto

Sendo h a quantidade de hóspedes, q a quantidade de quartos, $h_i q_j$ a variável que representa o i-ésimo hóspede no j-ésimo quarto, a fórmula que garante que cada hóspede vai estar somente em um único quarto é mostrada na Fórmula 3.5.

$$\forall i \in 1..h \left(\sum_{j=1}^{q} h_i q_j = 1 \right) \tag{3.5}$$

3.2.1.2 A quantidade de hóspede em um quarto deve ser menor ou igual à capacidade do quarto

Sendo q a quantidade de quartos, $h_i q_j$ o i-ésimo hóspede no j-ésimo quarto, $CAPACIDADE_j$ a constante que representa a capacidade máxima de um quarto j e q_j a variável representa o j-ésimo quarto alugado, a fórmula que garante que a capacidade de um quarto é respeitado é mostrada na Fórmula 3.6.

$$\forall j \in 1..q \left(\sum_{i=1}^{h} h_i q_j \leq CAPACIDADE_j \cdot q_j \right) \tag{3.6}$$

3.2.2 Casal

A seguir é detalhado as cláusulas relacionadas aos hóspedes que são um casal e aos quartos de cama de casal.

3.2.2.1 O casal deve estar no mesmo quarto

Sendo c a quantidade de casais, q a quantidade de quartos, $cx_z q_j$ o primeiro do z-ésimo casal no j-ésimo quarto e $cx_z q_j$ o mesmo, mas para o segundo do casal, a fórmula que garante que o par de hóspedes que são um casal e devem estar no mesmo quarto é mostrada na Fórmula 3.7.

$$\forall z \in 1..c, j \in 1..q \left(cx_z q_j = cy_z q_j \right) \tag{3.7}$$

3.2.2.2 Apenas casais podem estar em quartos de casal

Sendo k a quantidade de quartos de casal, $\sim c$ a quantidade de hóspedes que não são um casal e $h_i k_j$ a variável que representa o i-ésimo hóspede em um j-ésimo quarto de casal, a fórmula que garante que aqueles que não fazem parte de um casal não possam estar em um quarto de casal é mostrada na Fórmula 3.8.

$$\forall j \in 1..k \left(\sum_{i=1}^{\sim c} h_i k_j = 0 \right) \tag{3.8}$$

3.2.3 Antipatia

Quando dois hóspedes compartilham um mesmo quarto, gera uma antipatia atrelada a eles que deve ser minimizada.

3.2.3.1 Hóspedes compartilhando o mesmo quarto gera uma antipatia

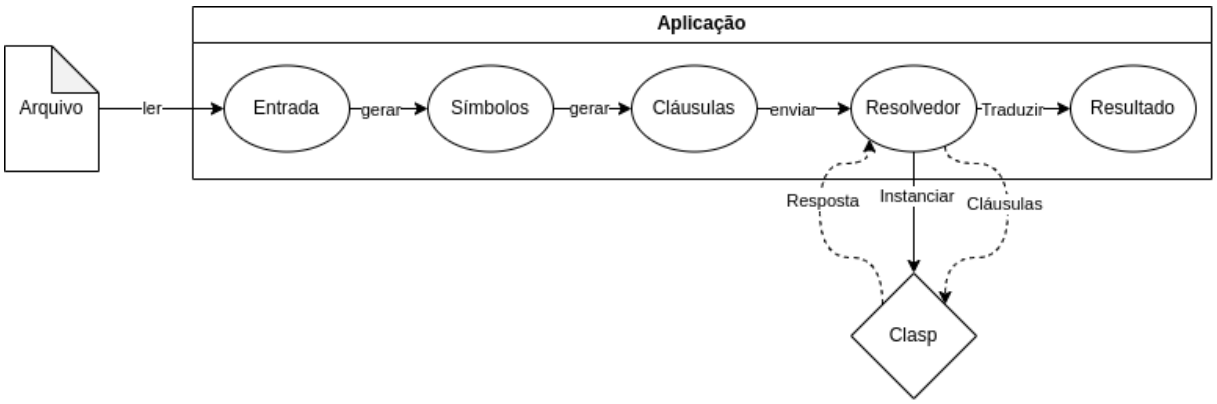
Sendo h a quantidade de hóspede, a Fórmula 3.9 garante que sempre que dois hóspedes compartilham o mesmo quarto, a variável de antipatia a_{ij} tenha a valoração 1. Desta forma, é possível minimizar aqueles hóspedes com uma grande antipatia entre si.

$$\forall i \in 1..(h-1), j \in (i+1)..h, z \in 1..q \left(h_i q_z + h_j q_z \leq a_{ij} + 1 \right) \tag{3.9}$$

4 Implementação

A implementação das formulações em uma aplicação foi construída por meio da linguagem Python e do solucionador Clasp, proposto por Gebser, Kaufmann e Schaub (2012). Antes da implementação em código de fato, uma preparação foi realizada nas formulações para um melhor desempenho e suporte ao Clasp. O fluxo da aplicação é mostrado na Figura 3.

Figura 3 – Fluxo da aplicação



Fonte: autores

Como mostra a Figura 3, se inicia lendo o arquivo com os dados dos quartos e hóspedes, por conseguinte, os símbolos $h_i q_j$, q_j e $a_i j$ são criados sequencialmente a partir do número 1 e as cláusulas são criadas, onde são enviadas, por meio de um *pipe*, para a entrada padrão do Clasp, que irá resolver e retornar a valoração dos símbolos, que por sua vez será traduzido e mostrado na tela. O código-fonte está disponível no repositório thiagohdaqw/hotel-pbs no Github.

A normalização das cláusulas, cálculo da antipatia e a codificação da formulação são detalhados nas próximas seções.

4.1 Normalização das cláusulas

As formulações propostas no capítulo anterior precisam ser normalizadas para serem utilizadas no Clasp, uma vez que, para uma melhor desempenho, as clausulas precisam estar no formato "$+A\ x1 + B\ x2 ... + M\ xn >= C$". Sendo assim, as cláusulas normalizadas são mostradas a seguir.

4.1.1 Todos os hóspedes devem estar em somente um quarto

$$\forall i \in 1..h \left(\sum_{j=1}^{q} h_i q_j \geq 1 \right) \tag{4.1}$$

$$\forall i \in 1..h \left(\sum_{j=1}^{q} h_i \bar{q}_j \geq q - 1 \right) \tag{4.2}$$

4.1.2 A quantidade de hóspede em um quarto deve ser menor ou igual à capacidade do quarto

$$\forall j \in 1..q \left(\sum_{i=1}^{h} (h_i \bar{q}_j) + CAPACIDADE_j \cdot q_j \geq h \right) \tag{4.3}$$

4.1.3 O casal deve estar no mesmo quarto

$$\forall z \in 1..c, j \in 1..q \left(cx_z q_j + c\bar{y_z} q_j \geq 1 \right) \tag{4.4}$$

$$\forall z \in 1..c, j \in 1..q \left(c\bar{x_z} q_j + cy_z q_j \geq 1 \right) \tag{4.5}$$

4.1.4 Apenas casais podem estar em quartos de casal

$$\forall j \in 1..k \left(\sum_{i=1}^{\sim c} h_i k_j > 0 \right) \tag{4.6}$$

$$\forall j \in 1..k \left(\sum_{i=1}^{\sim c} h_i \bar{k}_j >\sim c \right) \tag{4.7}$$

4.1.5 Hóspedes compartilhando o mesmo quarto gera uma antipatia

$$\forall i \in 1..(h-1), j \in (i+1)..h, z \in 1..q \left(h_i \bar{q}_z + h_j \bar{q}_z + a_{ij} \geq 1 \right) \tag{4.8}$$

4.2 Antipatia

Quando dois hóspedes compartilham um mesmo quarto, pode haver uma antipatia entre eles. O valor da antipatia é calculado como o oposto da afinidade, como mostra a função da Fórmula 3.1. Além disso, a afinidade pode depender do gênero e da avaliação do próprio hóspede, que não é reciproco. Sendo assim, é necessário definir uma lógica para gerar o valor da constante $ANTIPATIA_{ij}$ da Fórmula 3.3, e é mostrada na Figura 4.

Figura 4 – Algoritmo do cálculo da antipatia entre hóspedes

```
#Localização:   input.py -> parse_dislikes()
    for guest_a, guest_b in itertools.combinations(utils.iter_all_guests(guests), 2):
        type_a, type_b = guest_a[0], guest_b[0]
        dislike_ab = dislike_ba = 50

        if type_a != type_b:
            dislike_ab = dislike_ba = 100
        if guest_a in guests_dislikes and guest_b in guests_dislikes[guest_a]:
            dislike_ab = guests_dislikes[guest_a][guest_b]
        if guest_b in guests_dislikes and guest_a in guests_dislikes[guest_b]:
            dislike_ba = guests_dislikes[guest_b][guest_a]

        dislikes[f"{guest_a}{guest_b}"] = int((dislike_ab + dislike_ba) / 2)

    return dislikes
```

Fonte: Autores.

Como é possível observar na Figura 4, para cada combinação de 2 hóspedes é gerado uma antipatia. Por padrão, a antipatia é de 50, mas se forem de gênero diferente, torna-se 100, a menos que algum deles tenha avaliado especificamente o outro. No final, para se ter um único valor, é tirada a média entre as antipatias.

4.3 Codificação dos símbolos

O símbolo de cada variável é um número sequencial que se inicia em 1 e são armazenados em um dicionário. A chave do dicionário é o identificador do símbolo, sendo "q_i"o quarto e seu índice, "$h_i q_j$"o hóspede i em um quarto j e "a_{ij}"a antipatia entre dois hóspede i e j. Um exemplo é mostrado na Figura 5.

Figura 5 – Exemplo dos símbolos

```
{
    'C0': 1, 'C1': 2, 'D0': 3, 'T0': 4, 'T1': 5, 'Q0': 6,
    'M0C0': 7, 'M0C1': 8, 'M0D0': 9, 'M0T0': 10, 'M0T1': 11, 'M0Q0': 12,
        'M1C0': 13, 'M1C1': 14, 'M1D0': 15, 'M1T0': 16, 'M1T1': 17, 'M1Q0': 18,
        'F0C0': 19, 'F0C1': 20, 'F0D0': 21, 'F0T0': 22, 'F0T1': 23, 'F0Q0': 24,
    'AM0M1': 25, 'AM0F0': 26, 'AM1F0': 27
}
```

Fonte: Autores.

4.4 Codificação das cláusulas

A codificação acontece a partir das cláusulas normalizadas, onde cada cláusula gerada é um *string* e é armazenada em uma lista de cláusulas. A cláusula segue o padrão do Clasp e tem o formato "$+A\ x1 + B\ x2 ... + M\ xn >= C$".

A codificação da cláusula da capacidade de cada quarto é mostrada na Figura 6. Nela é possível observar que há uma iteração entre todos os quartos e hóspedes, construindo a Fórmula 4.3 e adicionando na lista *constraints* de cláusulas.

Figura 6 – Codificação da cláusula da capacidade de cada quarto

```python
#Localização:   pbs.py:generate_capacity_room_constraint()
def generate_capacity_room_constraint(guests, rooms, room_type, capacity):
    guest_count = len(guests[GuestType.Masculine.value]) + len(guests[GuestType.Feminine.value])

    for room in utils.iter_list(room_type, rooms):
        constraint = ""
        for guest in utils.iter_all_guests(guests):
            symbol = symbols[f"{guest}{room}"]
            constraint += f"+1 ~x{symbol} "

        room_symbol = symbols[room]
        constraints.append(constraint + f"+{capacity} x{room_symbol} >= {guest_count};")
```

Fonte: Autores.

4.5 Integração com o Clasp

A integração com o Clasp acontece por meio da instanciação do binário do Clasp pela classe Popen e é criado um *pipe* na entrada e saída padrão para a comunicação, a codificação é mostrada na Figura 7.

Figura 7 – Codificação da integração com o Clasp

```python
#LOCALIZAÇÃO:   solver.py:solve()
def solve(guests):
    with Popen(["clasp"], stdin=PIPE, stdout=PIPE, stderr=PIPE, text=True) as proc:
        write_pbs(proc.stdin)
        proc.stdin.close()
        result = translate_solution(proc.stdout, guests)
    return result

#LOCALIZAÇÃO:   solver.py:solve()
def write_pbs(file):
    file.write(pbs.generate_header())
    file.write("\n")

    file.write(pbs.min_constraint)
    file.write("\n")

    for constraint in pbs.constraints:
        file.write(constraint)
        file.write("\n")
```

Fonte: Autores.

Na entrada padrão do Clasp, por meio da função *write_pbs()*, é enviado o cabeçalho com a quantidade de símbolos e de cláusulas, a minimização e as cláusulas.

Na saída padrão do Clasp, é retornado o resultado se é satisfazível ou não. Caso for satisfazível, é retornado também a valoração das variáveis. Como as variáveis tem um formato x_i, é necessário traduzir para saber em qual quarto cada hóspede fora alocado. A codificação da tradução é mostrada na Figura 8.

Figura 8 – Codificação da tradução do resultado do Clasp

```python
#LOCALIZAÇÃO:   solver.py:translate_solution()
def translate_solution(file, guests):
    result = {}
    is_solution_line = lambda line: line.startswith("v ")
    symbols = [*pbs.symbols.items()]
    symbols.sort(key=lambda x: x[1])

    for line in file:
        if not is_solution_line(line):
            continue

        _, *vars = line.split()
        for var in vars:
            if var.startswith("-"):
                continue

            symbol = symbols[int(var[1:]) - 1][0]

            if symbol[0] not in "MF":
                continue

            guest_room = re.match(r"^(.[0-9]+)(.[0-9]+)$", symbol)

            room = guest_room.groups()[1]
            guest = guest_room.groups()[0]
            guest_name = guests[guest[0]][int(guest[1:])]
            if room not in result:
                result[room] = []
            result[room].append(guest_name)

    return result
```

Fonte: Autores.

4.6 Resultado

Por fim, o resultado é apresentado com a listagem de hóspedes alocados em cada quarto e no final o custo total do aluguel de cada quarto. Um exemplo de resultado utilizando o exemplo do Ribas (2023) é mostrado na Figura 9.

Figura 9 – Resultado do exemplo na aplicação implementada

```
Resultado ================================================
{'C0': ['Otaviano', 'Vanessa'],
 'D0': ['Bruno', 'Rose'],
 'D1': ['Iasmine', 'Chris'],
 'Q1': ['Casimiro', 'Igor', 'DavyJones']}
Custo: 400
```

Fonte: Autores.

5 Experimentos

A experimentação da solução aconteceu no Cluster Chococino na máquina pos2 e o *benchmark* é mostrado na Tabela 3.

Tabela 1 – *Benchmark* na máquina pos2

	Hóspedes	**Quartos**	**Variáveis**	**Cláusulas**	**Tempo**
enunciado	9	10	136	434	0m 0.05s
h2q13	2	13	40	32	0m 0.048s
h11q10	11	10	175	175	0m 0.333s
c6h14q13	14	13	260	1211	0m 0.333s
f11q9	11	9	163	526	0m 3.435s
h13q12	13	13	246	1024	0m 4.850s
h14q6	14	6	181	606	0m 1.811s
c5h14q18	14	18	361	1878	0m 2.034s
c4h14q18	14	18	361	1842	0m 2.635s
h14q13	14	13	286	1280	0m 11.706s
h15q13	15	13	313	1464	36m 40s

A seguir o *benchmark* com as entradas compartilhadas com a turma com *time limit* de 10 minutos.

Tabela 2 – *Benchmark* na máquina pos2 com a entrada do trio Daniel, Ruan e Wagner

	Tempo
input1	0m 0.05s
input2	0m 0.05s
input3	0m 0.04s
input4	0m 0.07s
input5	0m 0.07s
input6	0m 0.06s
input7	0m 0.12s
input8	0m 0.08s
input9	0m 0.07s
input10	TLE

Tabela 3 – *Benchmark* na máquina pos2 com a entrada do trio Marcos, Johan e Victor

	Tempo
fedorento	0m 0.04s
garanhao	0m 0.04s
infiel	0m 0.03s
input	0m 0.04s
Lobosolitario	0m 0.04s
Mais_gente_que_quarto	0m 0.02s
odiado	0m 0.04s
sem_casal	0m 0.04s
so_casal	0m 0.04s
so_F	0m 0.04s
so_H	0m 0.03s

6 Conclusão

Como foi visto neste trabalho, é possível modelar o problema da distribuição hoteleira com o método pseudo-Booleano, onde cada fórmula abordava uma característica do problema, reduzindo as possibilidades, minimizando o custo e maximizando o conforto.

Durante os experimentos, foi possível observar que o tempo para resolução para um conjunto pequeno é menos de 1 minuto. No entanto, para casos com mais de 15 hóspedes, como o caso de teste h15q13, o tempo de solução passa dos 36 minutos. Assim, é necessário otimizações, que visam reduzir a quantidade de variáveis e cláusulas.

Algumas otimizações que podem ser aplicadas em trabalhos futuros: unificar o casal em uma única variável, uma vez que eles são iguais na Fórmula 3.7; e omitir as variáveis de hóspedes solteiros em um quarto de casal, uma vez que é impossível, como mostra na Fórmula 3.8.

Outro ponto importante, trata-se da normalização valores de natureza diferentes na minimização, no caso do custo e antipatia. Como o *range* de valores do somatório da antipatia é entre 0 e 100, e se a média dos custos variarem neste *range*, pode acarretar não ter uma minimização do custo. Assim, séria interessante uma abordagem de normalização do custo no mesmo *range* da antipatia e utilizar estratégias de pesos, como na média ponderada, para permitir que o usuário escolha a prioridade entre antipatia e custo.

Referências

GEBSER, M.; KAUFMANN, B.; SCHAUB, T. Conflict-driven answer set solving: From theory to practice. 2012. Disponível em: <https://doi.org/10.1016/j.artint.2012.04.001>. Citado na página 13.

LARA, U. B. M. Uma abordagem pseudo-booleana para consolidação de máquinas virtuais com restrições de afinidade. 2020. Disponível em: <https://bdm.unb.br/bitstream/10483/30391/1/2020_UlyssesBernardMendesLara_tcc.pdf>. Citado na página 7.

RIBAS, B. On modelling virtual machine consolidation to pseudo-boolean constraints. 2012. Disponível em: <https://www.brunoribas.com.br/publicacoes/files/iberamia-2012-consolidation.pdf>. Citado na página 7.

RIBAS, B. Problema da distribuição hoteleira. 2023. Disponível em: <https://brunoribas.com.br/flia/2023-1/trabalhos/hotel.html>. Citado 2 vezes nas páginas 5 e 17.