VOL. 03 MAIO 2025

CHRONO CODE

ENTRE LUZES DE NÉON E REGRAS RÍGIDAS: O UNIVERSO DOS CSPS

0 00035 54562 0

CHRONO CODE

EDITORIAL

ENTRE LUZES DE NÉON E REGRAS RÍGIDAS: O UNIVERSO DOS CSPS

Nas aulas de Inteligência Artificial, um dos temas centrais explorados foi o Constraint Satisfaction Problem (CSP) — um tipo de problema em que o estado é definido por variáveis que devem obedecer a um conjunto de restrições. Durante os encontros, aprendemos como esse modelo pode ser aplicado a diversos contextos do cotidiano computacional, desde a pintura de mapas (sem repetir cores entre regiões adjacentes), passando pela otimização de linhas de montagem, até chegar ao clássico problema das 8 rainhas.

Além disso, vimos os diferentes tipos de consistência — de nó, de arco (AC-3), de caminho e de k-consistência — e como garantir essas propriedades ajuda a resolver os problemas de forma mais eficiente. Exploramos ainda algoritmos fundamentais como o backtracking, o forward-checking e a propagação de restrições, bem como a técnica de decomposição de árvores, aplicando tudo isso para resolver desafios como Sudoku e o problema das 4 rainhas.

. Nesta terceira edição da Chrono Code vamos nos aprofundar no universo dos CSPs, Abordando:

- **Uma explicação clara e acessível sobre o que são os CSPs e como funcionam;**
- **Uma análise dos tipos de CSPs, com exemplos visuais e funcionais;**
- **Os principais algoritmos utilizados para resolvê-los;**
- **Uma reflexão sobre vantagens e desvantagens dessa abordagem;**
- **E um exemplo prático resolvido passo a passo para mostrar como os CSPs podem ser aplicados de maneira criativa e relevante.**

O código em Python que resolve esse exemplo prático apresentado, pode ser encondrado no repositório do GitHub do ChronoCode.

PROBLEMAS DE SATISFAÇÃO DE RESTRIÇÕES (CSP(S))

DESVENDANDO OS QUEBRA-CABEÇAS DA IA

A inteligência artificial (IA) está em toda parte, e por trás de muitas de suas soluções mais inteligentes, encontramos os Problemas de Satisfação de Restrições (CSPs). Pense neles como quebra-cabeças lógicos que a IA precisa montar, onde cada peça tem seu lugar e suas regras. Um CSP é, essencialmente, um desafio matemático que envolve atribuir valores a um conjunto de elementos (variáveis) de forma que todas as condições (restrições) sejam atendidas (ALMABETTER, 2024; RUSSELL; NORVIG, 2020). Estes problemas representam as entidades como uma coleção homogênea de restrições finitas sobre variáveis, sendo resolvidos por métodos de satisfação de restrições (ALMABETTER, 2024; RUTTKAY, 1997).

O estudo dos CSPs é um campo de pesquisa ativo tanto na inteligência artificial quanto na pesquisa operacional (ALMABETTER, 2024). A formulação declarativa dos CSPs, que define o que precisa ser satisfeito em vez de como deve ser satisfeito, facilita essa modelagem. Essa abstração permite que avanços nas técnicas de resolução, como a propagação de restrições e as variantes de backtracking, sejam amplamente aplicados (ALMABETTER, 2024; RUSSELL; NORVIG, 2020). A programação de restrições (Constraint Programming - CP) é o campo de pesquisa que se dedica especificamente a abordar esses tipos de problemas, sendo reconhecida como uma tecnologia pervasiva e altamente bem-sucedida na resolução de uma ampla variedade de CSPs (ALMABETTER, 2024; RUTTKAY, 1997).

VARIÁVEIS, DOMÍNIOS E RESTRIÇÕES

Todo CSP é definido por três pilares:

- Variáveis: São as "incógnitas" do problema, os elementos que precisam de um valor Em um Sudoku, por exemplo, cada célula vazia é uma variável. No clássico Problema das Oito Rainhas, as variáveis podem representar a posição de cada rainha em uma coluna específica do tabuleiro de xadrez (ALMABETTER, 2024; RUTTKAY, 1997).
- Domínios: Para cada variável, há um conjunto de valores possíveis que ela pode assumir . No Sudoku, o domínio para cada célula vazia seria o conjunto de números de 1 a 9 (ALMABETTER, 2024; RUSSELL; NORVIG, 2020).
- Restrições: São as regras ou condições que especificam as relações entre as variáveis, limitando as combinações de valores que elas podem assumir. As restrições podem ser classificadas de acordo com o número de variáveis que envolvem:
- Unárias: Afetam uma única variável (por exemplo, uma tarefa deve começar após um determinado horário) (ALMABETTER, 2024; RUTTKAY, 1997).
- Binárias: Envolvem duas variáveis (por exemplo, duas tarefas não podem ocorrer simultaneamente) (ALMABETTER, 2024; RUTTKAY, 1997).
- N-árias: Abrangem três ou mais variáveis (por exemplo, três tarefas não podem se sobrepor) (ALMABETTER, 2024; RUTTKAY, 1997).

Uma solução para um CSP é alcançada quando um valor é atribuído a cada variável a partir de seu respectivo domínio, de tal forma que todas as restrições definidas são integralmente satisfeitas . Para ser considerada uma solução válida, a atribuição deve ser consistente (não violar nenhuma restrição) e completa (incluir todas as variáveis do problema).
A aparente simplicidade de definir um CSP esconde um desafio computacional significativo. A "satisfação" das restrições implica um resultado binário (satisfeito ou não), o que o caracteriza como um problema de decisão (ALMABETTER, 2024; DECHTER, 1992).

No entanto, o espaço de busca para problemas de tamanho moderado pode ser astronomicamente grande, resultando em uma explosão combinatória. Isso significa que, embora a formulação do problema seja elegante e clara, o processo de solução muitas vezes se assemelha a uma exploração exaustiva que rapidamente se torna intratável.

TIPOS DE CSP(S)

Embora a estrutura básica de um CSP seja universal, a complexidade e a natureza das restrições podem variar significativamente, dando origem a diferentes tipos de problemas.
Os CSPs Clássicos representam a formulação mais comum, onde o objetivo é encontrar uma atribuição completa de valores que satisfaça todas as restrições. Nestes, as restrições são consideradas "rígidas" (hard), o que significa que são imperativas e inflexíveis; cada solução válida deve satisfazê-las integralmente. Exemplos notáveis incluem:

- Problema das Oito Rainhas: O desafio de posicionar 8 rainhas em um tabuleiro de xadrez 8x8 de forma que nenhuma ataque a outra (ALMABETTER, 2024; RUTTKAY, 1997).
- Coloração de Grafos: Consiste em colorir os vértices de um grafo com k cores de modo que vértices adjacentes tenham cores diferentes (DECHTER, 1992; RUTTKAY, 1997).
- Sudoku: O popular quebra-cabeça de preencher células vazias com números de 1 a 9, sem repetições em linhas, colunas ou blocos 3x3 (ALMABETTER, 2024; RUSSELL; NORVIG, 2020).

A granularidade das restrições também define categorias de CSPs:

- CSPs Binários: São a forma mais simples, onde as restrições existem apenas entre pares de variáveis.
- CSPs Não-Binários: Envolvem restrições que se aplicam a três ou mais variáveis simultaneamente.

A distinção entre Restrições Rígidas (Hard) e Flexíveis/Suaves (Soft) é crucial para a aplicabilidade prática dos CSPs. As restrições rígidas são obrigatórias para qualquer solução válida (ALMABETTER, 2024; RUSSELL; NORVIG, 2020). Em contraste, as Restrições Flexíveis (Soft Constraints) relaxam a suposição de que todas as restrições devem ser completamente satisfeitas. Elas permitem que a solução não cumpra todas as restrições, frequentemente associando um nível de importância ou um custo a cada uma (ALMABETTER, 2024; RUSSELL; NORVIG, 2020).

A distinção entre Restrições Rígidas (Hard) e Flexíveis/Suaves (Soft) é crucial para a aplicabilidade prática dos CSPs. A capacidade de lidar com a evolução do problema no tempo é característica dos CSPs Dinâmicos. Nesses problemas, as variáveis ou restrições podem mudar ao longo do tempo (ALMABETTER, 2024).

isso exige algoritmos que possam se adaptar a essas mudanças em tempo real, como em problemas de agendamento em ambientes voláteis (ALMABETTER, 2024).

Por fim, os Problemas Super-Restritos (Over-constrained Problems) ocorrem quando é impossível satisfazer todas as restrições simultaneamente (ALMABETTER, 2024). Nesses casos, o objetivo é encontrar uma solução de compromisso, que satisfaça as restrições mais importantes ou que minimize as violações (ALMABETTER, 2024). O *framework* clássico de CSPs, nesses casos, apenas reporta a falha, o que é uma limitação para usuários que precisam de uma "melhor" relaxação ou de uma solução de compromisso (ALMABETTER, 2024).

VANTAGENS

- Representação Estruturada e Declarativa: Permitem modelar problemas complexos de forma clara, focando no "o quê" precisa ser resolvido, e não no "como" (ALMABETTER, 2024; RUSSELL; NORVIG, 2020).
- Versatilidade e Reusabilidade: São capazes de modelar uma vasta gama de problemas em diversas áreas, desde agendamento até jogos, e seus frameworks são reutilizáveis (ALMABETTER, 2024; RUSSELL; NORVIG, 2020).

- Técnicas de Resolução Eficientes: Algoritmos específicos, como backtracking e propagação de restrições, otimizam a busca por soluções, tornando-a mais rápida que métodos de força bruta (ALMABETTER, 2024; RUSSELL; NORVIG, 2020).
- Capacidade de Otimização: Com o uso de restrições flexíveis, permitem encontrar as melhores soluções em cenários complexos (ALMABETTER, 2024; RUSSELL; NORVIG, 2020).

DESVANTAGENS

- Complexidade Computacional: O problema geral é NP-completo, o que significa que, para instâncias muito grandes, a resolução pode se tornar intratável devido à explosão combinatória do espaço de busca (ALMABETTER, 2024; RUSSELL; NORVIG, 2020).
- Escalabilidade: Enfrentam dificuldades com problemas de grande porte, pois o espaço de busca cresce exponencialmente (ALMABETTER, 2024).

- Técnicas de Resolução Eficientes: Algoritmos específicos, como backtracking e propagação de restrições, otimizam a busca por soluções, tornando-a mais rápida que métodos de força bruta (ALMABETTER, 2024; RUSSELL; NORVIG, 2020).
- Capacidade de Otimização: Com o uso de restrições flexíveis, permitem encontrar as melhores soluções em cenários complexos (ALMABETTER, 2024; RUSSELL; NORVIG, 2020).

DESVANTAGENS

- Complexidade Computacional: O problema geral é NP-completo, o que significa que, para instâncias muito grandes, a resolução pode se tornar intratável devido à explosão combinatória do espaço de busca (ALMABETTER, 2024; RUSSELL; NORVIG, 2020).
- Escalabilidade: Enfrentam dificuldades com problemas de grande porte, pois o espaço de busca cresce exponencialmente (ALMABETTER, 2024).

- **Problemas Super-Restritos: Encontrar compromissos em cenários onde nem todas as restrições podem ser satisfeitas simultaneamente pode ser computacionalmente caro (ALMABETTER, 2024).**
- **Dinamicidade e Evolução: Exigem adaptações complexas para lidar com problemas onde variáveis ou restrições mudam ao longo do tempo (ALMABETTER, 2024).**
- **Ausência de Explicação: Os métodos clássicos não são inerentemente projetados para explicar o porquê de uma solução específica (RUTTKAY, 1997).**

APLICAÇÕES NO MUNDO REAL

- **Agendamento e Planejamento: Amplamente utilizados para criar horários universitários, agendar voos e planejar a produção, gerenciando dependências e recursos (ALMABETTER, 2024; RUSSELL; NORVIG, 2020).**
- **Alocação de Recursos: Essenciais para distribuir recursos limitados, como salas de aula ou equipamentos, garantindo que os limites não sejam excedidos (ALMABETTER, 2024).**

- **Jogos e Quebra-Cabeças: Fornecem a estrutura lógica para resolver uma vasta gama de desafios, incluindo Sudoku, palavras-cruzadas e o clássico Problema das Oito Rainhas (ALMABETTER, 2024; RUSSELL; NORVIG, 2020).**
- **Design e Configuração: Empregados em tarefas de engenharia e na configuração de produtos, garantindo que requisitos específicos sejam satisfeitos (DECHTER, 1992).**
- **Outras Áreas Emergentes: Encontram uso em visão computacional, processamento de linguagem natural, diagnóstico, robótica e bioinformática (DECHTER, 1992; RUTTKAY, 1997).**

REFERÊNCIAS:

- ALMABETTER. Constraint Satisfaction Problem in AI. 24 nov. 2024. Disponível em: https://www.almabetter.com/bytes/tutorials/artificial-intelligence/constraint-satisfaction-problem-in-ai. Acesso em: 27 maio 2024.
- DECHTER, R. Constraint Satisfaction and Propagation Tools. 1992. Disponível em: https://ics.uci.edu/~csp/r85.pdf. Acesso em: 27 maio 2024.
- RUSSELL, S. J.; NORVIG, P. Artificial Intelligence: A Modern Approach. 4. ed. Boston: Pearson, 2020.
- RUTTKAY, Z. Constraint satisfaction—a survey. Journal of Universal Computer Science, v. 3, n. 8, p. 1025-1045, 1997. Disponível em: https://www.researchgate.net/publication/228580853_Constraint_satisfaction-a_survey. Acesso em: 27 maio 2025.

EXEMPLO PRÁTICO

A Infiltração do Y0KA1HUNT3R

Em Neo-Xian, uma metrópole futurista onde dragões cibernéticos patrulham os céus e a magia ancestral se funde com a tecnologia de ponta, a guilda criminosa "A Serpente de Jade" controla o comércio de "Núcleos Espirituais" – artefatos bio-engenheirados que amplificam habilidades cibernéticas e rituais místicos.

Nosso protagonista, o "Y0KA1HUNT3R" (um hacker e místico), precisa se infiltrar em três fortalezas da Serpente de Jade para roubar Núcleos Espirituais específicos. Cada fortaleza possui sistemas de segurança únicos que exigem um tipo particular de Núcleo Espiritual para serem desativados ou para acessar áreas restritas. Y0KA1HUNT3R tem um inventário limitado de Núcleos e precisa decidir qual levar para cada fortaleza, além de escolher a melhor abordagem de infiltração para maximizar suas chances de sucesso e minimizar os riscos.

Modelagem como um Problema de Satisfação de Restrições Não-Binário (CSP)

Um CSP é definido por um conjunto de variáveis, seus domínios (valores possíveis) e um conjunto de restrições que as variáveis devem satisfazer. Neste caso, as restrições são "não-binárias" porque podem envolver mais de duas variáveis simultaneamente.

Variáveis

- Nucleo_Fortaleza1: O tipo de Núcleo Espiritual a ser levado para a Fortaleza 1 (Templo dos Sussurros).
- Nucleo_Fortaleza2: O tipo de Núcleo Espiritual a ser levado para a Fortaleza 2 (Ciber-Pagode).
- Nucleo_Fortaleza3: O tipo de Núcleo Espiritual a ser levado para a Fortaleza 3 (Distrito do Mercado Flutuante).
- Abordagem_Fortaleza1: O método de infiltração para a Fortaleza 1.
- Abordagem_Fortaleza2: O método de infiltração para a Fortaleza 2.
- Abordagem_Fortaleza3: O método de infiltração para a Fortaleza 3.

Domínios:

- Núcleos Espirituais (DOMINIO_NUCLEOS):
  - Núcleos Espirituais (DOMINIO_NUCLEOS):
  - 'Fogo do Dragão': Núcleo ofensivo, bom para combate.
  - 'Sussurro Fantasma': Núcleo etéreo, bom para furtividade.
  - 'Vontade de Ferro': Núcleo robusto, bom para hacking.
  - 'Véu Sombrio': Núcleo ilusório, bom para disfarce/furtividade.
  - 'Coração de Jade': Núcleo diplomático, bom para negociação.
- Núcleos Espirituais (DOMINIO_NUCLEOS):
  - 'Fogo do Dragão': Núcleo ofensivo, bom para combate.
  - 'Sussurro Fantasma': Núcleo etéreo, bom para furtividade.
  - 'Vontade de Ferro': Núcleo robusto, bom para hacking.
  - 'Véu Sombrio': Núcleo ilusório, bom para disfarce/furtividade.
  - 'Coração de Jade': Núcleo diplomático, bom para negociação.
  - Abordagens de Infiltração (DOMINIO_ABORDAGENS):
  - 'Furtividade': Infiltração silenciosa, evitando confrontos.
  - 'Combate': Infiltração direta, confrontando inimigos.
  - 'Hacking': Infiltração digital, desativando sistemas.
  - 'Diplomacia': Infiltração social, negociando ou persuadindo.

Restrições (Não-Binárias - Penalidades):

Para o Algoritmo Genético, as restrições serão modeladas como "penalidades". Quanto maior a penalidade, pior a solução. Nosso objetivo será minimizar a soma das penalidades (ou maximizar uma função de aptidão que subtrai as penalidades de um valor base).

- Restrição de Núcleo para Fortaleza 1 (Templo dos Sussurros):
  - Se Abordagem_Fortaleza1 for 'Furtividade', Nucleo_Fortaleza1 deve ser 'Sussurro Fantasma'. (Penalidade se não for).
  - Se Abordagem_Fortaleza1 for 'Combate', Nucleo_Fortaleza1 deve ser 'Fogo do Dragão'. (Penalidade se não for).
  - Qualquer outra combinação de núcleo/abordagem para F1 resulta em uma penalidade maior.
- Restrição de Núcleo para Fortaleza 2 (Ciber-Pagode):
  - Se Abordagem_Fortaleza2 for 'Hacking', Nucleo_Fortaleza2 deve ser 'Vontade de Ferro'. (Penalidade se não for).
  - Se Abordagem_Fortaleza2 for 'Furtividade', Nucleo_Fortaleza2 deve ser 'Véu Sombrio'. (Penalidade se não for).
  - Restrição de Núcleo para Fortaleza 3 (Distrito do Mercado Flutuante):

- Restrição de Núcleo para Fortaleza 3 (Distrito do Mercado Flutuante):
  - Se Abordagem_Fortaleza3 for 'Diplomacia', Nucleo_Fortaleza3 deve ser 'Coração de Jade'. (Penalidade se não for).
  - Se Abordagem_Fortaleza3 for 'Combate', Nucleo_Fortaleza3 deve ser 'Fogo do Dragão'. (Penalidade se não for).
- Inventário Limitado de Núcleos:
  - A Serpente de Jade possui apenas um de cada tipo de Núcleo Espiritual. Isso significa que Nucleo_Fortaleza1, Nucleo_Fortaleza2 e Nucleo_Fortaleza3 devem ser todos diferentes entre si. (Penalidade se houver repetição).
- Sinergia/Conflito de Abordagens:
  - Usar 'Combate' em duas fortalezas adjacentes (F1 e F2, ou F2 e F3) aumenta drasticamente o nível de alerta. (Penalidade alta se Abordagem_Fortaleza1 == 'Combate' e Abordagem_Fortaleza2 == 'Combate', ou Abordagem_Fortaleza2 == 'Combate' e Abordagem_Fortaleza3 == 'Combate').
  - 'Diplomacia' só é eficaz na Fortaleza 3 devido à sua natureza de mercado. (Penalidade se Abordagem_Fortaleza1 == 'Diplomacia' ou Abordagem_Fortaleza2 == 'Diplomacia').

- **Penalidade Geral por Desalinhamento Núcleo-Abordagem:**
  - **Uma penalidade base para qualquer combinação de núcleo e abordagem que não seja explicitamente ideal, mesmo que não seja proibida. Isso incentiva as soluções mais sinérgicas.**

Resolução com Algoritmo Genético

O Algoritmo Genético (AG) é uma meta-heurística inspirada na seleção natural. Ele busca soluções ótimas (ou quase ótimas) para problemas complexos através de um processo iterativo de evolução.

Passos do Algoritmo Genético:

- **Representação (Cromossomo): Cada solução potencial (um "indivíduo" ou "cromossomo") será uma lista de 6 elementos, representando as escolhas para as 6 variáveis: [Nucleo_F1, Nucleo_F2, Nucleo_F3, Abordagem_F1, Abordagem_F2, Abordagem_F3]**

- Função de Aptidão (Fitness Function): Esta função avalia a "qualidade" de um cromossomo. Ela começará com um valor base alto (ex: 1000) e subtrairá penalidades por cada restrição violada. Quanto maior a aptidão, melhor a solução.
  - aptidao = 1000
  - Aplicar penalidades baseadas nas restrições definidas acima.
  - Exemplo de penalidade: se Nucleo_Fortaleza1 não for 'Sussurro Fantasma' quando Abordagem_Fortaleza1 é 'Furtividade', subtrair 50 da aptidão.
- Inicialização: Gerar uma população inicial de cromossomos aleatórios. Cada cromossomo é uma combinação válida (aleatória) dos valores dos domínios.
- Seleção: Escolher os "pais" da população atual para criar a próxima geração. Usaremos a seleção por torneio:
  - Selecionar aleatoriamente k indivíduos da população.
  - O indivíduo com a maior aptidão entre os k selecionados é escolhido como pai.
  - Repetir para selecionar o segundo pai.

- Crossover (Cruzamento): Combinar o material genético de dois pais para criar novos "filhos". Usaremos o crossover de um ponto:
  - Escolher um ponto de corte aleatório no cromossomo.
  - Os genes antes do ponto de corte vêm de um pai, e os genes depois vêm do outro.
  - Ex: Pai1 = [N1, N2, N3, A1, A2, A3], Pai2 = [n1, n2, n3, a1, a2, a3]
  - Ponto de corte = 3
  - Filho1 = [N1, N2, N3, a1, a2, a3]
  - Filho2 = [n1, n2, n3, A1, A2, A3]
- Mutação: Introduzir pequenas alterações aleatórias nos cromossomos para manter a diversidade genética e evitar que o algoritmo fique preso em mínimos locais.
  - Com uma pequena probabilidade de mutação, escolher um gene aleatório no cromossomo e alterá-lo para um valor diferente dentro de seu domínio.
- Nova Geração: A nova população é formada pelos filhos gerados através de seleção, crossover e mutação.
- Critério de Parada: O algoritmo continua por um número fixo de gerações (ex: 1000) ou até que uma solução com aptidão perfeita (0 penalidades) seja encontrada.

A SOLUÇÃO EM PYTHON DESSE PROBLEMA PODE SER ENCONTRADO NO REPOSITÓRIO DO GITHUB DO CHRONOCODE : https://github.com/GG555-13/Chrono-Code---2025.1/tree/main/Entrega%203

Conclusão e Análise do Resultado

A saída do código mostrará a melhor solução encontrada e sua aptidão. Uma aptidão de 1000 indica que uma solução perfeita foi encontrada, ou seja, todas as restrições foram satisfeitas sem penalidades. Se a aptidão for menor, significa que a solução encontrada é a melhor possível dentro do número de gerações, mas ainda pode ter algumas penalidades menores (ou seja, não é uma solução "perfeita" no sentido de penalidade zero, mas é a que mais se aproxima).

Interpretação da Solução:

A solução final apresentará as escolhas para cada fortaleza:

- Fortaleza 1 (Templo dos Sussurros): Qual Núcleo levar e qual Abordagem usar.
- Fortaleza 2 (Ciber-Pagode): Qual Núcleo levar e qual Abordagem usar.
- Fortaleza 3 (Distrito do Mercado Flutuante): Qual Núcleo levar e qual Abordagem usar.

Por exemplo, uma solução ideal poderia ser:

- F1: Núcleo: 'Sussurro Fantasma', Abordagem: 'Furtividade'
- F2: Núcleo: 'Vontade de Ferro', Abordagem: 'Hacking'
- F3: Núcleo: 'Coração de Jade', Abordagem: 'Diplomacia'

Esta solução satisfaria as restrições de núcleo/abordagem para cada fortaleza e a restrição de núcleos únicos. Além disso, evitaria penalidades por abordagens de combate adjacentes ou diplomacia em locais errados.

A solução final apresentará as escolhas para cada fortaleza:

- Fortaleza 1 (Templo dos Sussurros): Qual Núcleo levar e qual Abordagem usar.
- Fortaleza 2 (Ciber-Pagode): Qual Núcleo levar e qual Abordagem usar.
- Fortaleza 3 (Distrito do Mercado Flutuante): Qual Núcleo levar e qual Abordagem usar.

Por exemplo, uma solução ideal poderia ser:

- F1: Núcleo: 'Sussurro Fantasma', Abordagem: 'Furtividade'
- F2: Núcleo: 'Vontade de Ferro', Abordagem: 'Hacking'
- F3: Núcleo: 'Coração de Jade', Abordagem: 'Diplomacia'

Esta solução satisfaria as restrições de núcleo/abordagem para cada fortaleza e a restrição de núcleos únicos. Além disso, evitaria penalidades por abordagens de combate adjacentes ou diplomacia em locais errados.