Plano de Desenvolvimento RKO em Python (To-Do Simplificado)

Este plano detalha as próximas etapas para aprimorar seu Random-Key Optimizer (RKO) em Python, focado em modularidade e fidelidade ao framework original.

Fase 1: Ajustes e Implementações do Shaking

- **Objetivo:** Padronizar a introdução de perturbações nas chaves aleatórias para diversificação da busca.
- **To-Do:**
  - **Implementar `shaking(keys, beta_min, beta_max)`:** Crie um método que aplique perturbações controladas. Ele deve escolher aleatoriamente entre quatro tipos de movimentos (`Swap`, `Swap Neighbor`, `Mirror`, `Random`) e aplicá-los β×n vezes (onde β é uma taxa aleatória dentro de `[beta_min, beta_max]`).
  - **Substituir:** Remova as funções `vizinhos` e `pertubacao` e use `shaking` em todas as meta-heurísticas que precisam perturbar soluções.

Fase 2: Implementações das Buscas Locais

- **Objetivo:** Aprimorar a fase de intensificação usando o método de busca local mais sofisticado do RKO.
- **To-Do:**
  - **Implementar `RVND(keys, local_search_heuristics_list, max_iter_no_improve)`:** Crie o algoritmo *Random Variable Neighborhood Descent*. Ele iterará aleatoriamente sobre uma lista de heurísticas de busca local, reiniciando a exploração se uma melhoria for encontrada.
  - **Implementar `swap_local_search(keys)`:** Busca local que troca pares de chaves.
  - **Implementar `mirror_local_search(keys)`:** Busca local que inverte o valor de chaves.
  - **Implementar `farey_local_search(keys)`:** Busca local que ajusta chaves com base na sequência de Farey.
  - **Implementar `nelder_mead_local_search(keys)`:** Busca local baseada na otimização de simplex.
  - **Substituir:** Substitua a chamada `LocSearch` nas meta-heurísticas pela chamada a `RVND`.

Fase 3: Refinamento das Meta-Heurísticas

- **Objetivo:** Integrar os novos componentes de `shaking` e `RVND` e completar as implementações das meta-heurísticas conforme o framework RKO.
- **To-Do:**
  - **Implementar `blending(parent_a, parent_b, factor, prob_inherit, mu_prob)`:** Crie um método geral para o crossover que combina dois pais com opções de herança e mutação.
  - **Ajustar `SimulatedAnnealing` (SA):** Use `shaking` para perturbações e aplique `RVND` antes do resfriamento da temperatura.
  - **Ajustar `GRASP`:** Implemente a fase construtiva com "line search" e *Restricted Candidate List* (RCL). Use `RVND` para a busca local.
  - **Ajustar `VNS`:** Use `shaking` com intensidade ajustada (`k * beta_min`). Aplique `RVND` após o `shaking`.
  - **Ajustar `ILS`:** Use `shaking` para perturbação e `RVND` para busca local.
  - **Ajustar `BRKGA`:** Use o método `blending` para o crossover. Aplique `RVND` à melhor solução de cada geração.
  - **Implementar `PSO`:** Crie o algoritmo PSO, onde as partículas são chaves aleatórias, e aplique `RVND` periodicamente a uma partícula.
  - **Implementar `GA`:** Crie um algoritmo genético padrão usando `blending` para crossover e `shaking` para mutação, com `RVND` na melhor solução.
  - **Implementar `LNS`:** Crie o *Large Neighborhood Search* com fases de "destruição" (usando `shaking`) e "reparo" (usando `farey_local_search` ou similar), e `RVND` para intensificação.

Fase 4: Gerenciamento do Pool de Soluções Elite

- **Objetivo:** Centralizar e otimizar a gestão do pool de soluções compartilhadas.
- **To-Do:**
  - **Criar `ElitePoolManager` Class:** Desenvolva uma classe para gerenciar o pool.
  - **Método `add_solution(keys, fitness)`:** Implemente a lógica para adicionar soluções, verificar clones (e aplicar `shaking` a eles para garantir diversidade), e manter o pool ordenado e com tamanho limitado.
  - **Integrar:** A função principal `solve()` deve inicializar o `ElitePoolManager` e todas as meta-heurísticas devem interagir com ele para obter e compartilhar soluções.
  - **Atualizar Workers:** Ajuste as funções `_worker` para utilizar a interface do

Este projeto de iniciação científica propõe novas formas de resolver problemas complexos de corte e empacotamento bidimensionais com peças irregulares, desafios comuns em diversos setores industriais, nos quais o uso eficiente de materiais tem impacto direto sobre os custos e a sustentabilidade. Além disso, busca-se desenvolver um modelo de aprendizado por reforço profundo capaz de realizar buscas complexas, adaptando-se de forma modular a cada tipo de problema.

A proposta parte da aplicação dos Random Key Optimizers (RKO), um framework modular que representa soluções como vetores contínuos no intervalo [0,1), realizando a busca por soluções em um hipercubo unitário, com o objetivo de padronizar os algoritmos de forma unificada. Esses vetores são interpretados por um componente separado, o decodificador, que os transforma em soluções viáveis para o problema em questão. Essa separação entre o processo de busca e a lógica do problema permite adaptar o RKO a diferentes contextos com grande flexibilidade.

O foco inicial do projeto está na construção de decodificadores eficientes para três variantes clássicas e desafiadoras do problema: o 2D Irregular Knapsack Problem, o 2D Irregular Bin Packing Problem e o 2D Irregular Strip Packing Problem. Por se tratarem de problemas NP-difíceis e de relevância direta para indústrias como a têxtil, aeronáutica e automobilística, o objetivo é desenvolver soluções que superem os resultados da literatura, reduzindo perdas de material e aumentando a eficiência operacional. Para lidar com a complexidade geométrica das peças, o projeto incorpora estruturas como o Inner Fit Polygon (IFP) e o No-Fit Polygon (NFP), fundamentais para garantir o encaixe preciso e sem sobreposição, além da utilização das regras clássicas de posicionamento de peças, técnicas comuns na literatura.

A segunda fase do projeto propõe uma nova técnica: integrar o RKO com Aprendizado por Reforço (Reinforcement Learning). A ideia é substituir os métodos tradicionais de busca do RKO por um modelo treinado para explorar o espaço de soluções do hipercubo unitário de forma adaptativa, com base na experiência acumulada. Em vez de ajustes manuais ou regras heurísticas fixas, o modelo, utilizando Deep Reinforcement Learning, será capaz de gerar, refinar e guiar os vetores de chaves aleatórias de maneira mais eficaz, com foco em encontrar as melhores soluções, dado um novo decodificador específico para cada problema.

Essa integração busca construir um sistema que não apenas resolva um problema específico, mas que aprenda uma estratégia geral de otimização. Um sistema com essa capacidade pode ser transferido para diferentes classes de problemas combinatórios, desde que acompanhado de um decodificador adequado. A combinação entre a estrutura genérica do RKO e o potencial adaptativo do RL aponta para um caminho promissor na construção de otimizadores mais robustos, versáteis e aplicáveis a cenários industriais de alta complexidade.