UNIEURO
CENTRO UNIVERSITÁRIO
GRUPO EDUCACIONAL CEUMA

Matéria: Pesquisa Operacional
Professor Dr.: Rafael Marconi Ramos
Alunos: Kelvin Sousa e Elson Gois
Período: 8º Semestre

Brasília - DF
2025

RELATÓRIO TÉCNICO

Sistema de Otimização Multi-Algoritmo para Maximização

Disciplina: Pesquisa Operacional

Aluno: Kelvin Sousa (58992) e Elson Gois (60771)

Arquivo executável: provab2.exe

Configuração: 10 parâmetros inteiros [1, 1000]

2. INTRODUÇÃO

Este relatório apresenta a implementação e análise comparativa de três algoritmos de otimização aplicados ao problema de maximização do executável provab2.exe. O sistema desenvolvido executa os algoritmos de forma sequencial, utilizando paralelização inteligente de threads quando aplicável, e realiza cronometragem detalhada com registro de data e hora em cada iteração.

O problema consiste em maximizar uma função objetivo implementada no executável provab2.exe, que recebe 10 parâmetros inteiros no intervalo [1, 1000] e retorna um valor de fitness. Foram executadas três corridas independentes do sistema completo, comparando os seguintes algoritmos:

- • Pattern Search (Hooke-Jeeves): Algoritmo de busca direta sequencial
- • Particle Swarm Optimization (PSO): Algoritmo de inteligência de enxame paralelizado
- • Híbrido (PSO + Pattern Search): Combinação de exploração global e refinamento local

O arquivo este no repositório git: https://github.com/kelvin-sous/PO/blob/main/README.md, commit: e494ea3c.

2. METODOLOGIA

2.1 Configuração do Sistema

O sistema foi desenvolvido em Python utilizando as seguintes tecnologias:

- • NumPy para operações vetoriais e manipulação de arrays
- • Multiprocessing para detecção automática de threads disponíveis
- • Datetime para registro temporal preciso de cada iteração
- • Subprocess para execução do programa externo

2.2 Detecção Automática de Recursos

O sistema detectou automaticamente 20 threads disponíveis no processador e realizou a divisão inteligente:

- • Pattern Search: 0 threads (algoritmo sequencial por natureza)
- • Particle Swarm: 10 threads (50% dos recursos disponíveis)
- • Híbrido (PSO + OS): 10 threads na fase PSO, 0 threads na fase Pattern Search

2.3 Configuração dos Algoritmos

Algoritmo	Parâmetros	Máx. Iterações
Pattern Search	Delta=1.0, delta_min=1e-6	50
Particle Swarm	n_particles=20, w=0.7, c1=1.5, c2=1.5	30
Híbrido	PSO: 20 partículas / PS: delta=0.1	PSO: 20 / PS: 20

2.4 Ponto Inicial

Todos os algoritmos iniciaram com o mesmo ponto: x0 = [50.5, 50.5, 50.5, 50.5, 50.5, 50.5, 50.5, 50.5, 50.5, 50.5], representando valores centrais do espaço de busca [1, 1000].

3. RESULTADOS EXPERIMENTAIS

3.1 Execução 1

Algoritmo	Fitness	Iterações	Tempo (min)
Pattern Search	520.000	31	3.48
Particle Swarm	1440.000	31	4.37
Híbrido	885.000	19	5.42
MELHOR	Particle Swarm	1440.000	13.31 min total

3.2 Execução 2

Algoritmo	Fitness	Iterações	Tempo (min)
Pattern Search	520.000	31	3.54
Particle Swarm	902.000	31	4.38
Híbrido	1194.000	19	5.41
MELHOR	Híbrido	1194.000	13.36 min total

3.3 Execução 3

Algoritmo	Fitness	Iterações	Tempo (min)
Pattern Search	520.000	31	3.49
Particle Swarm	906.000	31	4.34
Híbrido	1341.000	18	5.38
MELHOR	Híbrido	1341.000	13.25 min total

4. ANÁLISE COMPARATIVA

4.1 Resumo Estatístico

Algoritmo	Fitness Médio	Desvio Padrão	Tempo Médio (min)
Pattern Search	520.000	0.000	3.50
Particle Swarm	1082.667	306.181	4.36
Híbrido	1140.000	242.847	5.40

4.2 Performance dos Algoritmos

Pattern Search:

O algoritmo Pattern Search apresentou resultados consistentes, mas limitados. Em todas as três execuções, convergiu para o mesmo fitness de 520.0 com solução [51.5, 51.5, 51.5, 51.5, 51.5, 51.5, 51.5, 51.5, 51.5, 51.5]. Esta consistência absoluta (desvio padrão zero) indica que o algoritmo sempre encontra o mesmo mínimo local próximo ao ponto inicial, não conseguindo explorar outras regiões do espaço de busca. O tempo médio de execução foi de 3.50 minutos, o mais rápido entre os três algoritmos.

Particle Swarm Optimization:

O PSO demonstrou alta variabilidade nos resultados (desvio padrão de 306.18), obtendo fitness de 1440.0, 902.0 e 906.0 nas três execuções. Esta variação é característica de algoritmos estocásticos e demonstra a capacidade de exploração global do espaço de busca. O melhor resultado geral do estudo (fitness 1440.0) foi obtido pelo PSO na primeira execução. A paralelização com 10 threads resultou em tempo médio de 4.36 minutos, apenas 24.6% mais lento que o Pattern Search, mas com qualidade de solução muito superior.

Híbrido (PSO + Pattern Search):

O algoritmo híbrido apresentou o melhor desempenho médio (fitness 1140.0) e menor variabilidade entre os algoritmos estocásticos (desvio padrão de 242.85). A combinação da exploração global do PSO com o refinamento local do Pattern Search resultou em soluções consistentemente superiores. Obteve o melhor resultado em duas das três execuções (1194.0 e 1341.0). O tempo de execução foi de 5.40 minutos, refletindo a execução sequencial das duas fases (PSO paralelizado + PS sequencial).

5. CONCLUSÕES

5.1 Principais Descobertas

- 1. O algoritmo Híbrido demonstrou ser a escolha mais robusta, combinando boa performance média (1140.0) com consistência relativa (desvio padrão de 242.85). Venceu em 2 das 3 execuções.
- 2. O PSO apresentou o melhor resultado absoluto (1440.0), mas com maior variabilidade, tornando-o adequado quando múltiplas execuções são viáveis.
- 3. O Pattern Search, embora rápido, ficou preso em mínimo local (520.0) próximo ao ponto inicial, demonstrando limitação para problemas complexos de alta dimensionalidade.
- 4. A paralelização com threads foi efetiva: o PSO utilizou 10 threads e manteve tempo competitivo (4.36 min) com qualidade superior.

5.2 Relação Custo-Benefício

Algoritmo	Fitness/Minuto	Avaliação
Pattern Search	148.57	Baixo desempenho
Particle Swarm	248.32	Bom custo-benefício
Híbrido	211.11	Melhor equilíbrio

5.3 Recomendações

- Para otimização com restrição de tempo (execução única): Utilizar o algoritmo Híbrido, que oferece o melhor equilíbrio entre performance e consistência.
- Para otimização com múltiplas execuções disponíveis: Executar o PSO múltiplas vezes e selecionar o melhor resultado, aproveitando sua capacidade de exploração global.
- Para otimização em tempo real ou recursos limitados: O Pattern Search pode ser adequado apenas se o problema for unimodal ou o ponto inicial for próximo do ótimo global.

5.4 Implementação Técnica

O sistema implementado demonstrou eficiência na utilização de recursos computacionais:

- • Detecção automática de 20 threads disponíveis no processador
- • Divisão inteligente: 50% para PSO, 50% para Híbrido, 0% para Pattern Search
- • Aproveitamento de 100% dos recursos durante fase PSO
- • Cronometragem detalhada com data/hora em cada iteração para análise temporal precisa
- • Registro completo do histórico de convergência para análise posterior

Kelvin Sousa e Elson Gois

Data: 02 de Dezembro de 2025