
Manual Completo: Otimização de Alocação de Leitos de UTI com NSGA-II e GDE3

---

1. Introdução

Este manual apresenta a implementação, execução e interpretação de algoritmos evolutivos multiobjetivo aplicados à **alocação de leitos de UTI**, comparando dois métodos: **NSGA-II** (Algoritmo Genético Multiobjetivo) e **GDE3** (Evolução Diferencial Multiobjetivo).

O objetivo é:

- Minimizar tempo de espera de pacientes;
- Maximizar utilização dos leitos;
- Reduzir risco clínico;
- Minimizar custo terminal.

O guia inclui:

1. Estrutura do código (NSGA-II e GDE3);
2. Preparação e validação dos dados de pacientes;
3. Definição do problema de otimização;
4. Execução e análise de resultados;
5. Interpretação de métricas e logs;
6. Comparação detalhada entre os algoritmos.

---

2. Fluxo Estruturado do NSGA-II

2.1 Importações e Setup

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
from datetime import datetime
import logging
from typing import List, Dict, Tuple, Any

# Pymoo
from pymoo.core.problem import ElementwiseProblem
from pymoo.algorithms.moo.nsga2 import NSGA2
from pymoo.operators.crossover.sbx import SBX
from pymoo.operators.mutation.pm import PM
from pymoo.operators.sampling.integer_random_sampling import
IntegerRandomSampling
from pymoo.optimize import minimize
from pymoo.visualization.scatter import PCP
```

2.2 Configuração de Logging

```
def configurar_logging(nivel_logging: str = 'INFO') -> logging.Logger:
  logger = logging.getLogger("OtimizacaoUTI")
  logger.setLevel(getattr(logging, nivel_logging))
```

```
  ch = logging.StreamHandler()
  formatter = logging.Formatter('%(asctime)s - %(levelname)s - %(message)s')
  ch.setFormatter(formatter)
  logger.addHandler(ch)
  return logger
```

- **Função:** Cria logger "OtimizacaoUTI".
- **Saída:** Log detalhado de execução.

2.3 Geração e Validação de Dados

Funções Principais:

1. **Gerar Dados Sintéticos Realistas**

gerar_dados_simulacao_realista(numero_pacientes, horizonte_tempo, logger)

- Gera lista de pacientes com id, tempo de chegada, gravidade, tempo estimado de UTI.
- Usa distribuições estatísticas.

2. **Carregar ou Gerar Dados**

carregar_ou_gerar_dados(nome_arquivo, numero_pacientes, horizonte_tempo, logger)

- Lê CSV se existir; senão gera nova base e salva.

3. **Ajustar Tempos de UTI**

ajustar_tempos_uti_para_limites_realistas(pacientes, limite_maximo_horas=120, logger=None)

- Corrige tempos que ultrapassem limite máximo.

4. **Verificar Viabilidade**

verificar_viabilidade_sistema(pacientes, numero_leitos, horizonte_tempo, logger)

- Calcula taxa de ocupação, capacidade, demanda e viabilidade.

5. **Calcular Valores de Referência**

calcular_valores_referencia_realistas(pacientes, horizonte_tempo, logger)

- Tempo de espera máximo;
- Risco clínico máximo;
- Custo terminal máximo.

6. **Calcular Ocupação Precisa**

calcular_ocupacao_precisa(tempo_inicio_uti, tempo_fim_uti, horizonte_tempo)

- Retorna vetor de ocupação binária (horas x leitos).

2.4 Definição do Problema de Otimização

```
class ProblemaOtimizacaoUTI(ElementwiseProblem):
  # Atributos
  pacientes
  numero_leitos
  horizonte_tempo
  numero_pacientes
  logger
  tempo_espera_max_ref
```

```
    risco_clinico_max_ref
    custo_terminal_max_ref

    def _evaluate(self, x, out, *args, **kwargs):
        # Calcula objetivos e restrições
        pass
```

- **Objetivos:** tempo de espera, risco clínico, custo terminal, utilização.
- **Restrições:** violação de capacidade, precedência temporal.

2.5 Análise e Relatórios

1. **Gerar CSV e Métricas**

```
gerar_relatorio_detalhado_plano(pacientes, cronograma, nome_arquivo_saida,
numero_leitos, horizonte_tempo, logger)
```

2. **Analisar Resultados**

```
analisar_resultados_otimizacao(resultado, problema, pacientes, numero_leitos,
horizonte_tempo, logger)
```

- Identifica soluções ótimas e balanceadas.

3. **Visualizar Fronteira de Pareto**

```
visualizar_fronteira_pareto(resultado, problema, logger)
```

2.6 Simulação Principal

```
def executar_simulacao_completa():
    # Parâmetros
    NUMERO_LEITOS_UTI = 12
    HORIZONTE_PLANEJAMENTO_HORAS = 168
    NUMERO_PACIENTES = 30
    NOME_ARQUIVO_BASE = "base_pacientes_uti_realista.csv"
    NUMERO_GERACOES = 250
    TAMANHO_POPULACAO = 350
    LIMITE_MAXIMO_UTI = 120

    # Fluxo
    logger = configurar_logging()
    pacientes = carregar_ou_gerar_dados(NOME_ARQUIVO_BASE, NUMERO_PACIENTES,
HORIZONTE_PLANEJAMENTO_HORAS, logger)
    ajustar_tempos_uti_para_limites_realistas(pacientes, LIMITE_MAXIMO_UTI, logger)
    viabilidade = verificar_viabilidade_sistema(pacientes, NUMERO_LEITOS_UTI,
HORIZONTE_PLANEJAMENTO_HORAS, logger)
    problema = ProblemaOtimizacaoUTI(...)
    alg = NSGA2(
        pop_size=TAMANHO_POPULACAO,
        sampling=IntegerRandomSampling(),
        crossover=SBX(prob=0.9, eta=15),
        mutation=PM(eta=20)
    )
    resultado = minimize(problema, alg, ('n_gen', NUMERO_GERACOES), verbose=True)
```

```
  analisar_resultados_otimizacao(resultado, problema, pacientes, NUMERO_LEITOS_UTI,
HORIZONTE_PLANEJAMENTO_HORAS, logger)
  visualizar_fronteira_pareto(resultado, problema, logger)
  return resultado, True
if __name__ == '__main__':
  resultado, sucesso = executar_simulacao_completa()
```

---

3. Fluxo Estruturado do GDE3

3.1 Importações e Setup

- Similar ao NSGA-II, mas inclui **GDE3** do pymoode.

```
from pymoode.algorithms.gde3 import GDE3
```

- PYMODE_AVAILABLE = True se GDE3 estiver instalado.

3.2 Logging, Dados e Problema

- Mesmas funções para:
  - Logging;
  - Geração e validação de pacientes;
  - Ajuste de tempos;
  - Verificação de viabilidade;
  - Definição da classe ProblemaOtimizacaoUTI.
- **Diferença:** Algoritmo GDE3 tem parâmetros próprios: variant, CR, F.

3.3 Configuração do Algoritmo GDE3

```
def configurar_algoritmo_gde3(config_gde3):
  return GDE3(
    pop_size=config_gde3["tamanho_populacao"],
    variant=config_gde3["variante"],
    CR=config_gde3["CR"],
    F=config_gde3["F"]
  )
```

- Exploração agressiva via mutação diferencial;
- Seleção baseada em dominância.

3.4 Relatórios e Visualização

- Mesmos métodos de análise, CSV e gráficos de Pareto.
- Logs e métricas compatíveis com NSGA-II para comparação direta.

---

4. Demonstração de Cálculo de Referências

4.1 Dados Base de Pacientes

id	Chegada	Gravidade	Tempo UTI
0	11	9	80
1	4	3	120
...	...	...	...
29	9	5	32

- **Total de pacientes:** 30
- **Horizonte:** 168 horas

4.2 Cálculo do Tempo de Espera Máximo

Tempo_espera_max_paciente = min(Horizonte - Chegada, Tempo_UTI)

- Soma total: **1748 horas**

4.3 Cálculo do Risco Clínico Máximo

Risco_paciente = Tempo_espera_max × Gravidade

- Soma total: **8474 pontos**

4.4 Cálculo do Custo Terminal Máximo

Se (Chegada + Tempo_UTI > Horizonte):
  Custo = (Chegada + Tempo_UTI - Horizonte) × Gravidade
Senão:
  Custo = 0

- Soma total: **20 pontos**
- Apenas o paciente 25 gera custo terminal.

---

5. Interpretação de Métricas de Convergência

Métrica	Significado
n_gen	Geração atual
n_eval	Avaliações acumuladas
n_nds	Tamanho da fronteira de Pareto
cv_min	Menor violação de restrição
cv_avg	Média de violação
eps	ε-indicator (convergência)
indicator	Referência usada (ideal/nadir/f)

Comparação NSGA-II vs GDE3:

Métrica	NSGA-II	GDE3	Observação
n_gen	gerações	gerações	igual
n_eval	avaliações	avaliações	igual
n_nds	crescimento gradual	saltos	GDE3 explora mais
cv_min	viabilidade rápida	lenta	NSGA-II converge mais rápido
cv_avg	converge estável	oscila	NSGA-II mais estável
eps	decresce suavemente	oscila	GDE3 mais exploratório
indicator	alternância suave	alternância intensa	comportamento diferenciado

---

6. Comparação NSGA-II vs GDE3

6.1 Elementos Idênticos

- Mesma classe ProblemaOtimizacaoUTI;
- Mesmos objetivos, restrições e normalização;
- Mesmas variáveis: 30 pacientes;
- Mesma estrutura de dados e relatórios;
- Logging e monitoramento idênticos.

6.2 Diferenças

Aspecto	NSGA-II	GDE3
Algoritmo	Genético	Evolução Diferencial
Operadores	Seleção, crossover, mutação	Mutação diferencial + crossover
Exploração	Diversificação via operadores genéticos	Vetores diferenciais exploram regiões abruptas
Exploitation	Elitismo + ranking	Dominância restrita
Parâmetros	prob_crossover=0.9, eta=20	CR=0.4, F=(0.5,0.6), variant="DE/best/1/bin"

6.3 Fluxo de Execução

1. Carregar dados;
2. Analisar viabilidade;
3. Configurar problema;

4. Executar algoritmo (NSGA-II ou GDE3);
5. Analisar resultados;
6. Gerar relatórios e gráficos.

---

7. Conclusão e Interpretação Final

- O sistema de 12 leitos com horizonte de 168 horas suporta 1752 horas de demanda → taxa ocupação **86,9%**.
- Valores de referência fornecem uma régua para normalização:
    - **Tempo de espera máximo:** 1748 h
    - **Risco clínico máximo:** 8474 pontos
    - **Custo terminal máximo:** 20 pontos
- NSGA-II converge mais rápido e de forma estável; GDE3 é mais exploratório, com saltos na fronteira e oscilações no ε-indicator.
- Comparabilidade direta é garantida pelo uso de métricas e relatórios idênticos.
