# NSGA-II
## Explicação geral do algoritmo evolutivo

GUSTAVO FERNANDES DE ALMEIDA
gsutavo@outlook.com
*Universidade de Brasília*
6 de Dezembro de 2015

**Resumo**

O objetivo desse documento é explicar como o algoritmo de otimização multiobjetivo NSGA-II funciona. Otimização multiobjetivo busca otimizar vários objetivos distintos sem associar graus de importância para cada função de otimização. Em especial busca-se aprofundar em como sua implementação do algoritmo é feita.

*Keywords:* NSGA-II , elitista , algoritmo genético , otimização multiobjetivo

## 1 Introdução

O NSGA-II [1] é o aprimoramento do algoritmo genético de ordenação não-dominada e busca solucionar três problemas que o seu anterior (NSGA) apresentava:

- complexidade computacional alta da ordenação não-dominada.
- falta de elitismo.
- necessidade de especificar o parâmetro de compartilhamento.

O prejuízo da alta complexidade computacional ($O(MN^3)$), M é o número de objetivos e N o tamanho da população, torna o NSGA muito caro qunado aplicado para populações muito grandes, essa complexidade tem base no processo de ordenação não-dominada realizado durante cada geração do algoritmo.

Elitismo em um algoritmo evolutivo discrimina soluções em melhores e piores dependendo de sua performance. As soluções que apresentam

resultados melhores são priorizadas durante as outras fases do algoritmo, em especial durante a seleção para formar a *mating pool*[1]. Elitismo traz boas contribuições a um algoritmo genético como aumentar sua performance e prevenir a perda de uma boa solução depois de ser encontrada.

Mecanismos tradicionais para garantir a diversidade em uma população afim de obter uma variedade de soluções equivalentes tem se baseado no conceito de compartilhamento. Um problema dessa abordagem é a necessidade de um parâmetro de compartilhamento que deve ser estabelecido. Um algoritmo sem esse requisito seria mais desejável.

# 2 Sobre o algoritmo

Para explicar os pilares que compõem o NSGA-II, temos que passar por três etapas: como é feita a *Fast Nondominated Sorting Approach*[2], como preservar a diversidade das soluções e como o *main loop*, o laço principal, no qual as iterações do algoritmo são feitos.

## 2.1 Fast Nondominated Sorting Approach

Essa abordagem busca resolver o problema da alta complexidade computacional que o NSGA tinha. Utilizando a maneira proposta a seguir $O(MN^2)$ computações serão necessárias.

Antes de começar detalhar os passos do algoritmo é preciso compreender o conceito de dominação no contexto desse algoritmo. Uma solução $x$ é dominante quando comparada a outra solução $y$ quando todos os resultados das funções de otimização são melhores para $x$ que para $y$. Se em qualquer uma das funções o resultado de $y$ for melhor que $x$ nenhuma das duas soluções é dominante ou dominada. No caso que de $y$ ser superior em todos os resultados temos que $y$ domina $x$.

O primeiro passo é calcular duas entidade para cada solução:

- O contador de dominação $n_p$, ele representa o número de soluções que dominam a solução $p$;
- O conjunto de soluções $S_p$, que lista todas as soluções que $p$ domina.

Depois de calcular esses duas entidades, devemos separá-las em *fronts*. *Fronts* são as estruturas responsáveis por manter o elitismo nesse algoritmo, um *front* agrupa soluções em níveis hierárquicos.As melhores soluções ficam

[1] *Mating pool*: estrutura de um algoritmo genético que representa o sub-grupo de uma população selecionado para reproduzir entre si e, portanto, passar seus genes adiante.

[2] Abordagem de ordenação não-dominada rápida, em tradução livre

em um *front* dedicado, as soluções não tão boas no segundo *front* e assim por diante.

O primeiro *front* é determinado pela soluções que tiverem $n_p$ igual a zero, ou seja, não são dominadas por outra solução. Em seguida para cada solução $p$ com $n_p = 0$ visitamos cada uma dos membros ($q$) de seu $S_p$ e diminuímos o valor do seu $n_p$ por uma contagem de um. Se o novo $n_p$ de $q$ for zero colocamos $q$ numa nova lista $Q$. $Q$ formará o segundo *front*. Repetimos esse processo com $Q$ para determinar o terceiro *front*. Realizamos esse processo até que todos os *fronts* sejam definidos e todas as soluções façam parte de um *front*.

É possível que exista o mesmo número de *fronts* do que de soluções. Nesse caso cada *front* contém uma única solução

## 2.2 Preservação de Diversidade

A maneira com que o NSGA-II mantém a variedade das soluções é composta por duas estratégias: estimação de densidade e operador de *crowded-comparison* [3].

Para obter uma estimativa da densidade de soluções ao redor de uma solução particular calculamos a distância média de dois pontos aos lados da solução para cada objetivo. Essa distância, chamada de *crowding distance*, é usada para estimar o perímetro do cuboide formado usando os vizinhos mais próximos como vértices. A *crowding distance* mostra o quão perto uma solução está das outras soluções mais próximas à ela. Soluções muito próximas são soluções similares enquanto soluções distantes são soluções diferentes.

Para calcular a *crowding distance* precisamos ordenar a população em ordem ascendente para cada uma das funções objetivo. Para cada função, as soluções nas extremidades (a de menor valor e a de maior) são as soluções de fronteira, suas distâncias são definidas como infinitas. As outras soluções intermediárias tem seus valores de distância calculado como a diferença absoluta normalizada dos valores das funções das suas duas soluções adjacentes. Esse cálculo é feito em todas funções de objetivo para cada uma das soluções. A *crownding distance* total é a soma de cada um dos valores de distância para cada função objetivo. As funções objetivo devem ser normalizadas antes de calcular a *crowding distance*.

Quando cada solução tem seu valor de *crowding distance* calculado é possível avaliar as soluções do ponto de vista de diversidade. *Crowding distance* pequena significa que a solução esta numa área densa de soluções parecidas enquanto um valor alto é indicativo de uma solução mais alternativa.

[3] *Crownded-comparison*: comparação de desnsidade, em tradução livre.

O operador de *crowded-comparison* ($\prec_n$) guia o processo de seleção em vários estágios do algoritmo afim de uniformemente espalhar o *front* de Pareto ótimo.

Para um cada indivíduo $i$ de uma população sendo $i_{rank}$ o rank de não-dominação e $i_{distance}$ o valor da *crowding distance* temos:

$$\begin{aligned} i \prec_n j \; se \; (i_{rank} < j_{rank}) \\ OU \; ((i_{rank} = j_{rank}) \; E \; (i_{distance} > j_{distance})) \end{aligned} \tag{1}$$

Dessa maneira respeitamos o elitismo enquanto mantemos uma variedade alta. Uma solução de rank inferior não é escolhida enquanto os *fronts* melhores ainda não estiverem esgotados e soluções diferentes são priorizadas.

### 2.3 Main Loop

O *main loop* define a iteração do algoritmo. A Figura 1 auxilia na compreensão das etapas. Inicialmente uma população $P_0$ (a população mãe) é criada aleatoriamente, é a fase 1 na imagem. A população é ordenada em relação ao seu fator de não-dominação ($n_p$). Utilizando técnicas de seleção por torneio binário, recombinação e operadores de mutação criamos a população descendente $Q_0$ (população filha) de tamanho $N$, fase 2. A inicialização do algoritmo é diferente das outras iterações.

O laço principal segue a sequência a seguir:

- A população $R_t = P_t \cup Q_t$ de tamanho 2N é formada, $R_t$ é a população completa da fase 3;
- A população $R_t$ é ordenada de acordo com o valor de não-dominação;
- O conjunto de soluções do primeiro *front* ($F_1$), as não-dominadas por nenhuma outra solução, são privilegiada mais que qualquer outro grupo. Se $F_1$ for menor que N todos seus membros irão formar a nova população $P_{t+1}$ (nova população mãe);
- Outros membros de $P_{t+1}$ são selecionados dos outros *fronts* pela ordem de seu ranking até completar N membros;
- Caso existam mais membros no *fronts* do que vagas em $P_{t+1}$ o operador de *crowding-comparison* ($\prec_n$) define os indivíduos o *front* que completarão as vagas remanescentes, essa é a fase 5 na imagem;
- A nova população $P_{t+1}$ de tamanho N passa por seleção, *crossover* e mutação para criar uma nova $Q_{t+1}$ de tamanho N;
- Outra iteração é feita até o número de gerações predeterminado é atingido.

A Figura 1 ilustra o processo que o algoritmo NSGA-II define.

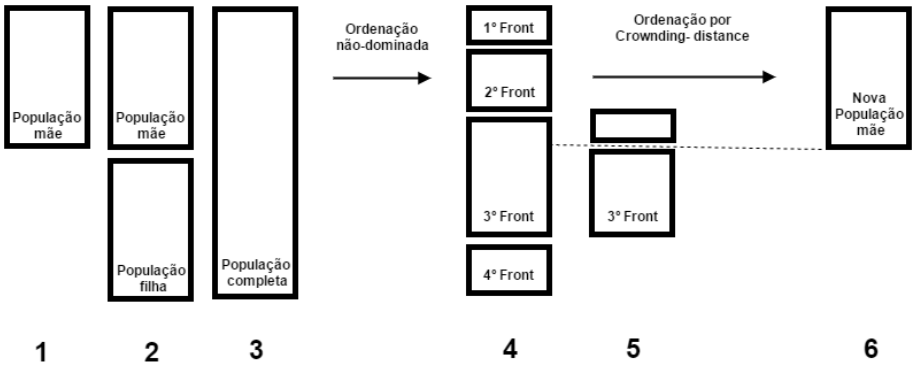


Figura 1: Figura representativa do processo do NSGA-II.

## Conclusão

O NSGA-II atinge todas os objetivos que se propõe, tornando-o uma boa alternativa ao seu antecessor.

Em especial, ele retira a necessidade de um parâmetro de compartilhamento introduzindo o procedimento de *crowding comparison* além de aplicar a noção de elitismo e ser computacionalmente mais eficiente o NSGA.

## Referências

**1.** Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.