NSGA-II 算法

姜孟冯
中国矿业大学 / 应急管理部信息研究院

日期：2024 年 10 月 30 日

1 算法背景

1.1 多目标优化问题

多目标优化问题 (MOP) 是涉及多个目标函数同时优化的数学问题。需要在两个或多个相互冲突的目标之间进行权衡的情况下作出最优决策。

定义 1.1. 多目标优化问题 (multi-objective optimization problem, 简称 MOP)

$$\begin{aligned} \min \quad & F(X) = (f_1(X), f_2(X), \ldots, f_r(X)) \\ \text{s.t.} \quad & g_i(X) \geq 0, i = 1, 2, \ldots, k, \\ & h_i(X) = 0, i = 1, 2, \ldots, l \end{aligned}$$

记该问题的可行解集为

$$\Omega = \{X \in \mathbf{R}^n | g_i(X) \geq 0, h_j(X) = 0 (i = 1, 2, \ldots, k, j = 1, 2, \ldots, l)\}$$

定义 1.2. Pareto 支配 (Pareto Dominance)
X 支配 Y, 记为 $X \prec Y$ ，当且仅当以下条件同时成立

$$\forall i \in \{1, 2, \ldots, r\}, f_i(X) \leq f_i(Y) \text{ 且 } \exists j \in \{1, 2, ..., r\}, f_j(X) < f_j(Y)$$

定义 1.3. Pareto 最优解集 (Pareto Optimal Set)
如果一个可行解 $X^* \in \Omega$ 被称之为 Pareto optimal solution，当且仅当 X 不被其他的解支配。易见 X^* 往往不是一个，常常用 $\{X^*\}$ 表示最优解集

$$P^* = \{X^*\} = \{X \in \Omega | \neg \exists X' \in \Omega, f_j(X') \leq f_j(X) (j = 1, 2, \ldots, r)\}$$

定义 1.4. Pareto 边界 (Pareto Optimal Front, 简称 PF)
Pareto Optimal Set 中每个解对应的目标值向量组成的集合称之为 Pareto Front（如图 1所示）。

$$PF^* = \{f(X) = (f_1(X), f_2(X), \ldots, f_r(X)) | X \in \{X^*\}\}$$

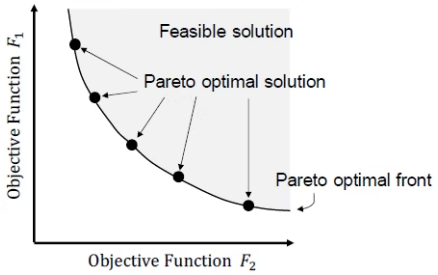

图 1: Pareto 边界示意图

1.2 多目标进化算法与分类

多目标进化算法 (Multi-objective Evolutionary Algorithm，简称 MOEA) 是一种非常有效的解决多目标优化问题 (MOP) 的方法。1989 年，David Goldberg 在其著作《Genetic Algorithm in Search, Optimization and Machine Learning[1]》中，提出了用进化算法实现多目标的优化技术，对多目标进化算法的研究具有重要的方向性指导意义。根据不同的进化机制，MOEA 可以分为三类[2]：

1. 基于分解的 MOEA：权重聚合函数、切比雪夫聚合、基于惩罚的边界交叉、MOEA/D
2. 基于支配关系的 MOEA：NSGA-II、NPGA、SPEA2、PESA、PAES、MOMGA、mBOA、MMOGA
3. 基于指标的 MOEA：SMS-EMOA、IBEA、HypE

1.3 NSGA-II 算法

1994 年，Srinivas 和 Deb 提出了 NSGA(nondominated sorting genetic algorithm)[3]，可以有效求解 MOP 问题，但仍存在一些不足之处：(1) 缺少最优个体保留机制 (2) 共享参数大小不易确定 (3) 构造非支配集的时间复杂度高 $(O(rN^3))$。2000 年，Deb 等对算法进行优化，提出了 NSGA-II[4]，2002 年又对其中的非支配集排序方法进行了改进[5]，把时间复杂度提高到 $(O(rN^2))$, 但只适用于处理低维优化问题 (维数小于等于 3)。2013 年，Deb 等又提出了 NSGA-III[6]，用以解决高维数问题。

本文主要集中讨论 NSGA-II 算法。

2 NSGA-II 算法原理及框架

2.1 算法框架

NSGA-II 算法是一种多目标进化算法，全称是 Elitist Non-dominated Sorting Genetic Algorithm（精英非支配排序遗传算法），NSGA-II 的主要特征有三点：

- 基于快速非支配排序的等级 (rank) 划分方法
- 计算拥挤度 (crowding-distance) 的方法
- 用以上二者构建的偏序比较算子代替共享参数 (sharing fuction)

NSGA-II 算法基于遗传算法的逐代进化思想，开始时随机产生一个父代种群 P_t，然后利用选择、交叉、变异的方式生成临时子代 Q_t，利用“非支配排序”与“拥挤度计算”来对父子种群中的每个个体进行适应度评估，选出新一代的 n 个个体作为新的父代 P_{t+1}，如此不断循环进化，直到满足结束条件。

NSGA-II 基本流程框架如图 2所示：

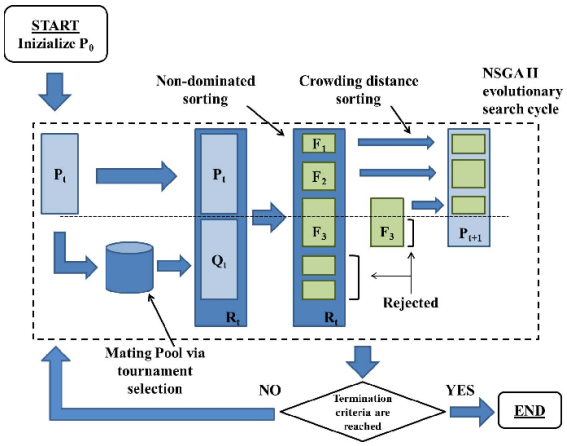

图 2: NSGA-II 流程示意图

2.2 快速非支配集排序方法

在 NSGA 中，非支配集排序方法的时间复杂度是 $(O(rN^3))$,NSGA-II 改进了非支配集的构造方式，进化群体 F 中的第 2 个个体只需要 1 次比较操作，F 中的第 3 个个体最多只需要 2 次比较，以此类推，F 中第 N 个个体最多只需要 (N-1) 次比较操作。在最坏情况下，即当尸中所有个体均为非支配个体时，算法的比较操作总次数为

$$1+2+3+\cdots+(N-1)=N^2/2$$

考虑到每次比较时有 r 个子目标，因此，算法的时间复杂度为 $O(rN^2)$。

```
fast-non-dominated-sort(P)
for each p ∈ P
  S_p = ∅
  n_p = 0
  for each q ∈ P
    if (p ≺ q) then                    If p dominates q
      S_p = S_p ∪ {q}                  Add q to the set of solutions dominated by p
    else if (q ≺ p) then
      n_p = n_p + 1                    Increment the domination counter of p
  if n_p = 0 then                      p belongs to the first front
    p_rank = 1
    F_1 = F_1 ∪ {p}
i = 1                                  Initialize the front counter
while F_i ≠ ∅
  Q = ∅                                Used to store the members of the next front
  for each p ∈ F_i
    for each q ∈ S_p
      n_q = n_q − 1
      if n_q = 0 then                  q belongs to the next front
        q_rank = i + 1
        Q = Q ∪ {q}
  i = i + 1
  F_i = Q
```

2.3 拥挤度计算方法

设下标为 i 的个体的拥挤度记为 $i_{distance}$, 目标函数值记为 $f_k(x_i)$, 用 f_k^{max} 与 f_k^{min} 分别表示目标函数的最大最小值，则拥挤度计算可使用以下公式：

边界上的点：$i_{distance}=\infty$

边界以外的点：$i_{distance}=\sum_{k=1}^{r}(\frac{f_k(x_{i+1})-f_k(x_{i-1})}{f_k^{max}-f_k^{min}})$

为了计算个体的拥挤度，需要按每个子目标函数值进行排序，最坏情况下对 r 个子目标进行排序的时间复杂度为 $O(rN\log N)$，而计算拥挤度本身的时间为 $O(rN)$，所以此算法的时间复杂度为 $O(rN\log N)$。

```
crowding-distance-assignment(I)
l = |I|                                number of solutions in I
for each i, set I[i]_distance = 0      initialize distance
for each objective m
  I = sort(I, m)                       sort using each objective value
  I[1]_distance = I[l]_distance = ∞    so that boundary points are always selected
  for i = 2 to (l − 1)                 for all other points
    I[i]_distance = I[i]_distance + (I[i + 1].m − I[i − 1].m)/(f_m^max − f_m^min)
```

2.4 精英保留策略

NSGA-II 的适应度评估过程中，考虑了个体 i 的两个属性，非支配排序等级与拥挤度，记作 i_{rank} 与 $i_{distance}$，二者可以构成一个偏序关系 $\prec_n$。

定义 2.1. 个体之间的偏序关系 $\prec_n$

$$i \prec_n j \; if (i_{rank} < j_{rank}) or ((i_{rank} = j_{rank}) and (i_{distance} > j_{distance}))$$

利用该偏序关系，可以构建偏序比较算子，该算子避免了任何用户定义的参数，同时改善了计算复杂度，有效地保护了多样性。精英保留策略 1. 首先将父代种群 C; 和子代种群 D: 合成种群 R:。2. 根据以下规则从种群见生成新的父代种群 Ci+1: 根据 Pareto 等级从低到高的顺序，将整层种群放入父代种群 C,-1，直到某一层该层个体不能全部放入父代种群 C:-1; 将该层个体根据拥挤度从大到小排列，依次放入父代种群 C+1 中，直到父代种群 C+1 填满。

3 算法步骤及实现

3.1 基本步骤

以 fast-non-dominated-sort() 表示非支配排序，以 crowding-distance-assignment() 表示拥挤度计算，以 $Sort(F_i, \prec_n)$ 表示基于偏序关系 $\prec_n$ 进行适应度排序，则 NSGA-II 主循环算法可用伪代码描述如下：

$R_t = P_t \cup Q_t$	combine parent and offspring population
$\mathcal{F} =$ `fast-non-dominated-sort`(R_t)	$\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, \ldots)$, all nondominated fronts of R_t
$P_{t+1} = \emptyset$ and $i = 1$	
until $\lvert P_{t+1}\rvert + \lvert\mathcal{F}_i\rvert \le N$	until the parent population is filled
`crowding-distance-assignment`$(\mathcal{F}_i)$	calculate crowding-distance in $\mathcal{F}_i$
$P_{t+1} = P_{t+1} \cup \mathcal{F}_i$	include ith nondominated front in the parent pop
$i = i + 1$	check the next front for inclusion
Sort$(\mathcal{F}_i, \prec_n)$	sort in descending order using $\prec_n$
$P_{t+1} = P_{t+1} \cup \mathcal{F}_i[1 : (N - \lvert P_{t+1}\rvert)]$	choose the first $(N - \lvert P_{t+1}\rvert)$ elements of $\mathcal{F}_i$
$Q_{t+1} =$ `make-new-pop`(P_{t+1})	use selection, crossover and mutation to create a new population Q_{t+1}
$t = t + 1$	increment the generation counter

具体来说，NSGA-II 使用快速非支配排序来保证收敛性，并且利用拥挤距离来保证分布性。

3.2 参数设置

NSGA-II 算法的参数设置对算法的性能有重要影响。主要参数包括：

- 种群规模（N）：控制种群中个体的数量。较大的种群规模可以提供更广泛的搜索空间，但计算成本更高。
- 最大进化代数（MaxGen）：控制算法运行的代数。较大的最大进化代数可以提高算法的收敛性，但计算成本也更高。
- 交叉概率（Pc）：控制交叉算子应用的概率。较高的交叉概率可以促进种群多样性，但可能导致过早收敛。
- 变异概率（Pm）：控制变异算子应用的概率。较高的变异概率可以引入新的个体，但可能破坏种群的收敛性。

3.3 开源实现

C nsga2: https://www.iitk.ac.in/kangal/codes.shtml

Java MOEA Framework：https://github.com/MOEAFramework/MOEAFramework

Python pymooo: https://github.com/anyoptimization/pymoo

Julia Metaheuristics: https://github.com/jmejia8/Metaheuristics.jl

R rmoo: https://github.com/Evolutionary-Optimization-Laboratory/rmoo

Rust Optirustic: https://github.com/s-simoncelli/optirustic

C# MOEAs: https://github.com/qshzhang/MOEAs

可见，NSGA-II 算法作为多目标优化 (MOO) 领域的经典算法，被众多编程语言予以实现，方便后人研究与改进。

4 算法应用

NSGA-II 算法应用非常广泛，

4.1 工程设计优化

工程设计优化是 NSGA-II 算法最常见的应用领域之一。在工程设计中，往往需要同时考虑多个目标，如成本、性能、可靠性等。NSGA-II 算法可以有效地处理多目标优化问题，找到满足所有目标约束的最佳解决方案。

案例：飞机机翼设计优化

飞机机翼设计是一个典型的多目标优化问题，需要同时考虑机翼的升力、阻力、重量等多个目标。使用 NSGA-II 算法对飞机机翼进行优化，可以找到满足升力、阻力、重量约束的最佳机翼形状。逻辑分析：

定义目标函数：计算飞机机翼的升力和阻力。定义 NSGA-II 算法：包含初始化种群、迭代优化、选择操作、交叉和变异操作等步骤。拥挤度计算：计算种群中个体的拥挤度，用于选择操作。交叉操作：对两个父个体进行交叉，生成两个子个体。变异操作：对个体进行变异，引入多样性。运行 NSGA-II 算法：迭代优化，找到最优解。输出最优解：打印飞机机翼的最佳形状。

4.2 资源分配优化

资源分配优化是 NSGA-II 算法的另一个重要应用领域。在资源分配问题中，需要在有限的资源约束下，将资源分配给多个任务或项目，以最大化整体收益或最小化成本。NSGA-II 算法可以有效地解决此类问题，找到满足资源约束的最佳分配方案。

案例：项目组合优化

项目组合优化是一个典型的资源分配优化问题，需要在有限的预算约束下，选择一组项目进行投资，以最大化投资回报率。使用 NSGA-II 算法对项目组合进行优化，可以找到满足预算约束的最佳项目组合。

NSGA-II 算法在风力发电机组优化中得到了广泛的应用。风力发电机组优化涉及到多目标，如发电量最大化、成本最小化和环境影响最小化。

应用步骤：

定义目标函数：

4.3 数据挖掘优化

数据挖掘优化是 NSGA-II 算法的又一应用领域。在数据挖掘中，需要从大量数据中提取有价值的信息，如模式、规则或分类器。NSGA-II 算法可以优化数据挖掘算法的参数，以提高挖掘效率和准确性。

案例：特征选择优化

特征选择是数据挖掘中的一个重要步骤，需要从原始数据集中选择最具区分性的特征。使用 NSGA-II 算法对特征选择进行优化，可以找到满足分类或回归任务要求的最佳特征子集。

参考文献

[1] GOLDBERG D E. Genetic Algorithm in Search, Optimization and Machine Learning, Addison[J]. W esley Publishing Company, R eading, MA, 1989, 1(98): 9.

[2] 郑金华, 邹娟. 多目标进化优化[M]. 科学出版社, 2017.

[3] SRINIVAS N, DEB K. Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms [J/OL]. Evolutionary Computation, 1994, 2(3): 221-248. DOI: 10.1162/evco.1994.2.3.221.

[4] DEB K, AGRAWAL S, PRATAP A, et al. A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II[C]//SCHOENAUER M, DEB K, RUDOLPH G, et al. Parallel Problem Solving from Nature PPSN VI. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000: 849-858.

[5] DEB K, PRATAP A, AGARWAL S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II[J/OL]. IEEE Transactions on Evolutionary Computation, 2002, 6(2): 182-197. DOI: 10.1109/4235.996017.

[6] DEB K, JAIN H. An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints[J/OL]. IEEE Transactions on Evolutionary Computation, 2014, 18(4): 577-601. DOI: 10.1109/TEVC.2013.2281535.