# Evolutionary Multi-Objective Optimization

Yiming Chen, Linh Vu Tu

16 février 2025

**Résumé**

This is the abstract of the report.

## 1 Introduction

This is the introduction section. This is where we cite a paper [2].

## 2 Methodology

This is the methodology section. Another reference [1] is cited here.
Lorem ipsum

### 2.1 Task 2

The Pareto set of $LOTZ$ is defined by all of the individuals of the form $1^k0^{n-k}$ where $k \in [0..n]$. Indeed, given $\alpha \in \{0,1\}^*$ and $p, q \in \mathbb{N}$, $1^p0\alpha0^q \prec 1^{p+1}\alpha0^q$ and $1^p\alpha10^q \prec 1^p\alpha0^{q+1}$ with respect to $LOTZ$. Thus, the Pareto front of $LOTZ$ is $\{(k, n-k)|kin[0..n]\}$. Both of these sets have cardinality $n+1$.

These results easily generalize to the $mLOTZ$ objective function. Indeed, $mLOTZ$ simply computes $LOTZ$ on **disjoint** chunks of an individual. In other words, Pareto-optimal values are also optimal on the $m$ individual chunks of size $n'$. It follows that the Pareto set of $mLOTZ$ is simply the set of $m$-concatenations of individuals of size $n'$ in the Pareto set of $LOTZ$. Formally, we can write the Pareto set as $1^{k_1}0^{n'-k_1}1^{k_2}0^{n'-k_2}\dots1^{k_m}0^{n'-k_m}$ with $k_1, \cdots, k_m \in [0..n']$. The Pareto front is written as

$$\{(k_1, n'-k_1, k_2, n'-k_2, \cdots, k_m, n'-k_m)|(k_1, \cdots, k_m)in[0..n']^m\}$$

Both of these sets have cardinality $(n'+1)^{m/2}$.

### 2.2 Task 3

Our algorithm for non-dominated sorting builds the directed acyclic graph (DAG) corresponding to the partial order defined by the multi-objective function. Since we use adjacency lists, the time complexity is $\mathcal{O}(N^2)$.

Iterating on the fronts (`Graph::pop_and_get_fronts`) is a flood-fill algorithm. Since we use a queue, this behaves like breadth-first search as every vertex and every edge is visited only once. The complexity of BFS is $\mathcal{O}(N+M) = \mathcal{O}(N^2)$.

Finally, the algorithm terminates and runs in $\mathcal{O}(N^2)$ time.

Proving the correctness of our algorithm requires one extra step. When iterating on a front $i$, we decrement the in-degree of all of the neighbors for each outgoing edge from $i$. We can prove by induction over the fronts that the vertices ending up with in-degree zero are exactly those in front $i+1$, which concludes the correctness of the algorithm.

### 2.3 Task 4

The crowding distance of an individual is the sum of its crowding distance relative to each objective. Since there are $m$ objectives, the time complexity of the function has a $\mathcal{O}(m)$ multiplicative factor due to the outermost loop.

Sorting the individuals in a population is performed in $\mathcal{O}(NlogN)$ time, and computing the crowding distance of a single individual has $\mathcal{O}(1)$ runtime since evaluating $f$ is done in constant time.

Finally, the time complexity of computing the crowding distance of a population is $\mathcal{O}(mNlogN)$.

### 2.4 Task 6

In our batched experiments, we find out that the Pareto front is systematically covered after a few epochs when the population is small ($N \leq 1024$). We choose $m$ and $m$ so that $n > m$. Since we choose $N = 4M = 4\left(\frac{2n}{m}+1\right)^{m/2}$, $N$ grows exponentially with $m$, we had to constrain $m$ and $n$ to relatively small values.

## 3 Results

This is the results section.

## 4 Conclusion

This is the conclusion section.

## Références

[1] Another Author. *Sample Book Title.* Publisher Name, City, Country, 2024.

[2] Author Name. Example paper title. *Journal Name*, 10(2):123–145, 2025.