

# Gerardium Rush Project Report

## Group Sphalerite

**Date: 24/05/2024**

### Group Members:

Zhang, Xiaoye

Huang, Fan

Shen, Xilei

Kahie, Manawi

Crespin, Francois

Ge, Jiaqi

Wang, Chaodao

## Table of Contents

### 1. Software Development

- Overview
- Genetic Algorithm
- Validation
- Simulator
- Post-Processing
- Further Improvements

### 2. Analysis

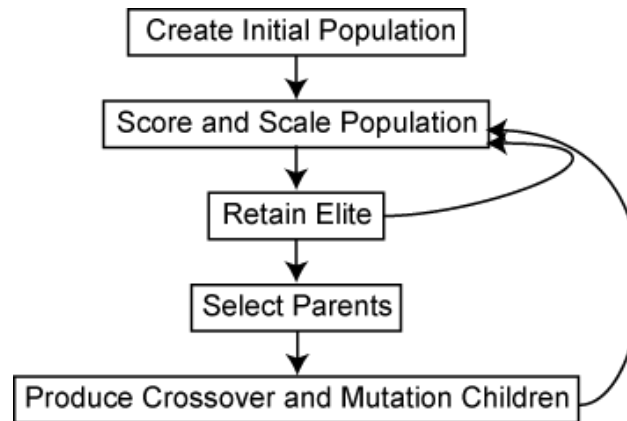
- Economic Variables
- Circuit Complexity and Configuration Changes
- Impact of Increased Waste Disposal Costs
- Analysis of Circuit Changes with the Number of Units
- Analysis of Execution Time vs. Number of Threads
- Analysis of Efficiency vs. Number of Threads

# Software Development

## Overview

The software development section outlines the creation and optimization of a genetic algorithm, the validation processes involved, the simulation of circuit performance, and post-processing techniques used to visualize results. This section also discusses potential future improvements.

## 1 Genetic Algorithm



Genetic algorithms (GAs) are search heuristics that mimic natural evolution. They generate high-quality solutions for optimization problems using bio-inspired operators such as mutation, crossover, and selection. In this project, the GA is used to optimize circuit design parameters, enhancing the efficiency and effectiveness of the circuits.

### 1.1 Algorithm Details

#### 1) Create Initial Population

The initial population is randomly generated with a predetermined number of individuals (NumPopulation). Each individual is subjected to a validity check. If found invalid, the individual is regenerated until a valid one is obtained. This ensures that the initial population meets basic feasibility requirements, setting a solid foundation for the optimization process.

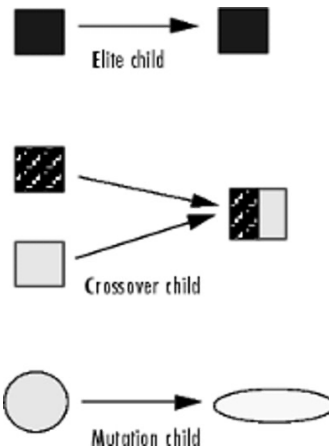
#### 2) Score and Scale Population

Each individual in the population is evaluated using the Evaluate\_Circuit function to determine its fitness. Valid individuals are scored, and the population is sorted by fitness. This sorting guides the selection process for generating the next generation, ensuring that higher fitness individuals have a greater chance of passing on their genes.

#### 3) Retain Elite

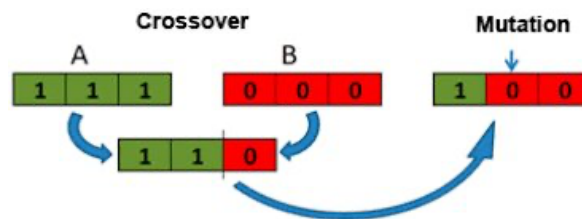
To preserve high-quality solutions, the top-performing individuals are retained in the population. Specifically, the first NumPopulation - NumOffspring individuals, representing the best performers, are carried over directly to the next generation.

#### 4) Select Parents



Parents are selected from the population to generate offspring. The selection is based on fitness, with the highest fitness individuals chosen as parents (NumParents). Two parents are randomly selected from this pool to generate NumOffspring offspring, ensuring genetic diversity.

#### 5) Produce Crossover



Crossover operations are performed to generate new offspring. A predetermined number of crossover points (NumCross) are selected for gene exchange, occurring based on a crossover probability. This introduces genetic diversity by combining genes from two parents, potentially leading to better solutions.

#### 6) Mutate Substitution

Mutation operations introduce variations in the offspring. A gene position is randomly selected and replaced with a new random valid value. This mutation occurs depending on the mutation rate, preventing premature convergence and maintaining genetic diversity.

### 1.2 Genetic Algorithm Methods we implemented

#### 1) Single-point Crossover

In single-point crossover, genes from the beginning to the crossover point are copied from one parent, while the remaining genes are copied from the other parent. This method introduces variability by combining genetic material from two parents.

#### 2) Two-point Crossover

In two-point crossover, genes between two points are swapped between the parents. This method introduces more variability than single-point crossover by allowing two segments of genes to be swapped.

### 3) Multi-point Crossover

Multi-point crossover involves alternating genes between each pair of points from both parents, providing higher exploration and variability. This method leads to potentially better solutions by thoroughly exploring the solution space.

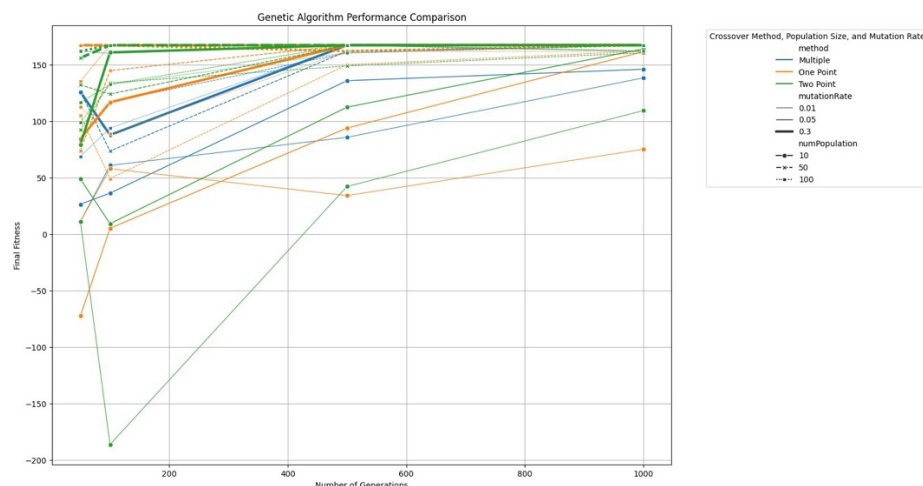
### 4) Uniform Crossover

In uniform crossover, each gene is independently chosen from either parent with a 50% probability. This maximizes randomness and diversity, aiding in a thorough exploration of the solution space.

### 5) Parameters Tuning

Parameters such as the number of offspring, crossover probability, and mutation rate are tuned to enhance GA performance. Multiple runs with different parameter configurations are conducted, and their performance is compared. The performance of different configurations is plotted over iterations to identify the best settings. Tuning parameters is crucial for achieving optimal performance, as the right combination can significantly enhance the algorithm's efficiency and effectiveness.

## 1.3 Performance Evaluation



### 1) Initial Population Performance

The performance of the initial population is evaluated to understand the starting point of the optimization process. Analyzing the fitness values of the initial population provides insights into the initial solution quality and the potential improvement scope.

### 2) Convergence Analysis

The convergence of the GA over generations is tracked by plotting fitness values over time. This analysis helps observe the convergence behavior and stability of the algorithm, indicating how quickly and effectively it reaches optimal solutions.

### 3) Comparison of Crossover Methods

The effectiveness of different crossover methods is compared by evaluating their performance. Single-point,

two-point, multi-point, and uniform crossover methods are analyzed to determine which method provides the best balance of exploration and exploitation, leading to optimal solutions.

## 1.4 Improvements we made

### 1) Hybrid Algorithms

Hybrid algorithms combine GA with other optimization techniques, such as Particle Swarm Optimization (PSO), to enhance performance. Leveraging the strengths of multiple optimization techniques can lead to better solutions, improving both convergence speed and solution quality.

### 2) Adaptive Parameters

Adaptive parameter tuning involves dynamically adjusting parameters like mutation rate and crossover probability during the optimization process. This adaptability improves the GA's flexibility and responsiveness to different stages of optimization, ensuring sustained performance improvements.

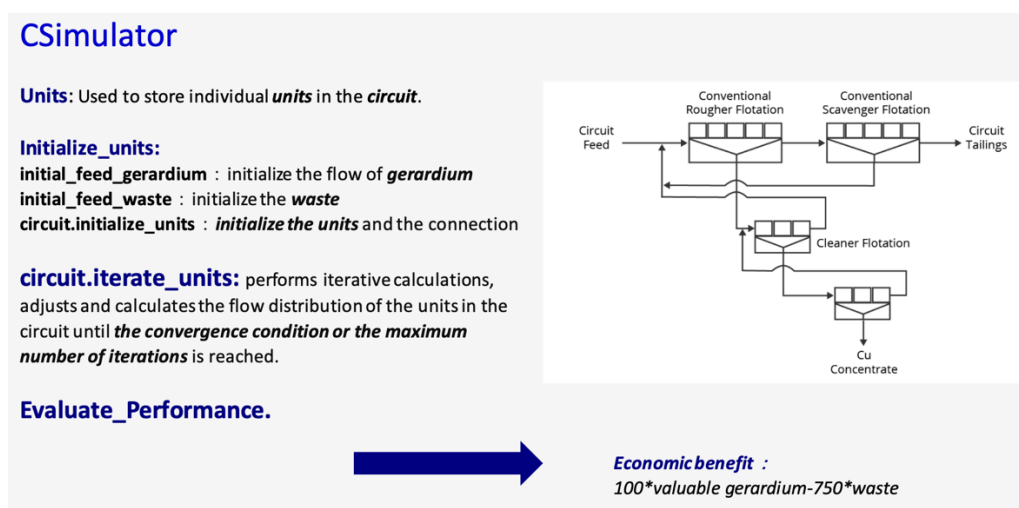
## 2 Validation

Main criteria used for validity checker:

- 1) **Self-Recycle:** Avoid feedback loops that invalidate results.
- 2) **Disconnected Units:** Ensure all units are part of the network from the feed to the output.
- 3) **Same Destination for All Streams:** Distribute outputs to different destinations for valid separation.
- 4) **No Route to All Streams:** Ensure each unit has paths to both concentrate and tailings streams.

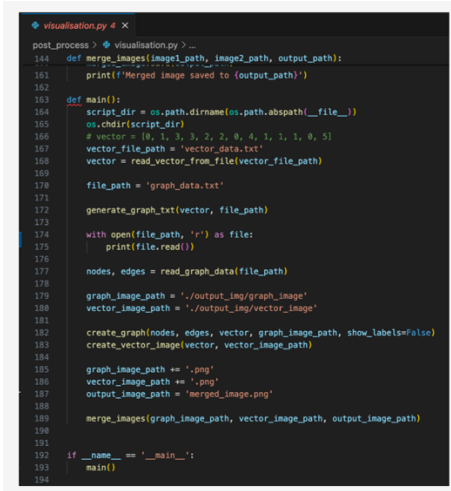
There are more criteria implemented in the code, with optimized order.

## 3 CSimulator



- 1) **Units Initialization:** Set up initial feed flows and unit connections.
- 2) **Iteration Process:** Perform iterative calculations to adjust and calculate flow distributions until convergence.
- 3) **Performance Evaluation:** Calculate economic benefits based on valuable gerardium and waste.

## 4 Post-Processing



- post\_process
  - └─ \_\_init\_\_.py
  - └─ graph\_data.txt
  - └─ merged\_image.png
  - └─ output\_img
    - └─ graph\_image.png
    - └─ vector\_image.png
  - └─ vector\_data.txt
  - └─ visualisation.py

python visualisation.py    inside post\_process folder

./run\_visualisation.sh    in the root directory

- 1) **Data Processing:** Use scripts to visualize the circuit data and generate graphical representations.
- 2) **Performance Metrics:** Analyze and display performance metrics such as grade and economic benefits.

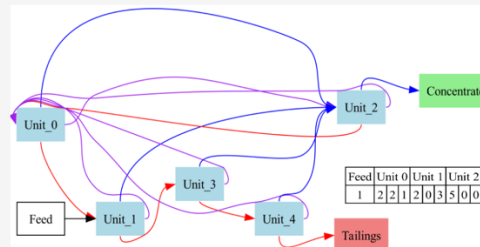
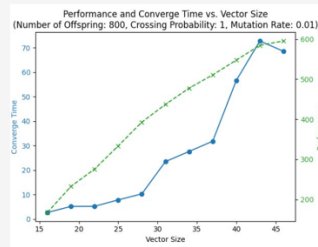
## 5 Further Improvements can do

- 1) **Population Initialization:** Optimize the initial population setup.
- 2) **Parallelization:** Shift from OpenMP to MPI for better performance.
- 3) **Crossover Methods:** Combine strengths of different crossover methods.
- 4) **Dynamic Interface Addition:** Implement real-time monitoring and interactive parameter adjustments.

# Analysis

## Economic Variables:

### Model analysis



The convergence time is almost proportional to the size of the vectors, but the performance at convergence is unstable. This could mean that the parameters were not set in such a way that the genetic algorithm outputs on the true global optimum when the vectors are large.

When a new benefit function  $300 * \text{valuable} - 450 * \text{waste}$  is set, a different permutation is created.

The economic variables examined in this project primarily include the price of gerardium and the cost of disposing of waste material. The price of gerardium remains constant at £100 per kg across all scenarios, providing a stable revenue figure for comparison. Conversely, the cost of waste disposal varies significantly, ranging from £500 to £1000 per kg. This variation in disposal costs is critical as it directly impacts the profitability and strategic decisions within the processing circuits.

Higher waste disposal costs incentivize the design of more complex processing circuits. The trend observed indicates that as waste disposal costs increase, there is a corresponding shift towards more intricate circuit configurations. This complexity is likely an attempt to reduce waste volume by recycling materials within the process or improving extraction efficiency, thereby mitigating higher disposal expenses. This dynamic highlights the balancing act between operational complexity and cost savings, where the goal is to minimize waste without excessively increasing operational overheads.

## Circuit Complexity and Configuration Changes:

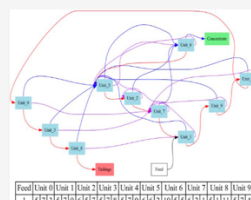
### How the optimum circuits change

The price paid for gerardium relative to the cost of disposing of the waste material



£100 for every kg of gerardium.  
£-500 for every kg of waste.

**Linear.**  
**Fewer circles.**



£100 for every kg of gerardium.  
£-750 for every kg of waste (default).



£100 for every kg of gerardium.  
£-1000 for every kg of waste.

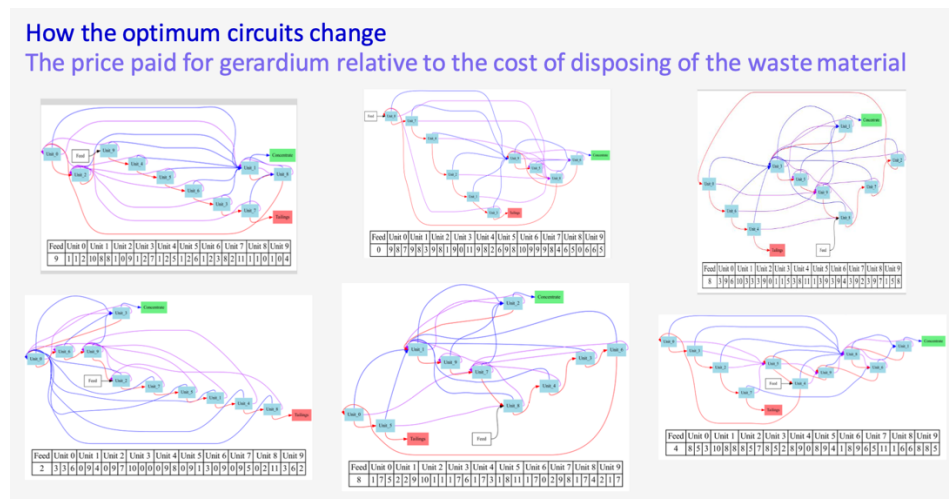
**Complex.**  
**More circles.**

The analysis of circuit complexity reveals a clear progression from simpler to more complex configurations as economic pressures change. At lower waste disposal costs, circuits tend to be more linear

with fewer connections, reflecting a straightforward processing approach. As the cost of waste disposal rises, circuits become increasingly complex with more interconnections and processing stages. This transformation suggests an optimization strategy aimed at either maximizing the yield of gerardium or minimizing waste production.

For instance, with a default waste disposal cost of £750 per kg, the circuits exhibit a moderate level of complexity, balancing operational efficiency with waste minimization. When the disposal cost escalates to £1000 per kg, circuits adopt the most complex structures observed, indicating a strategic shift to incorporate additional stages of material handling and refined separation techniques. This complexity allows for greater recycling within the units and potentially higher overall efficiency, albeit at the cost of increased energy use and maintenance requirements.

### Impact of Increased Waste Disposal Costs:



The increased cost of waste disposal has several implications on circuit design and operation:

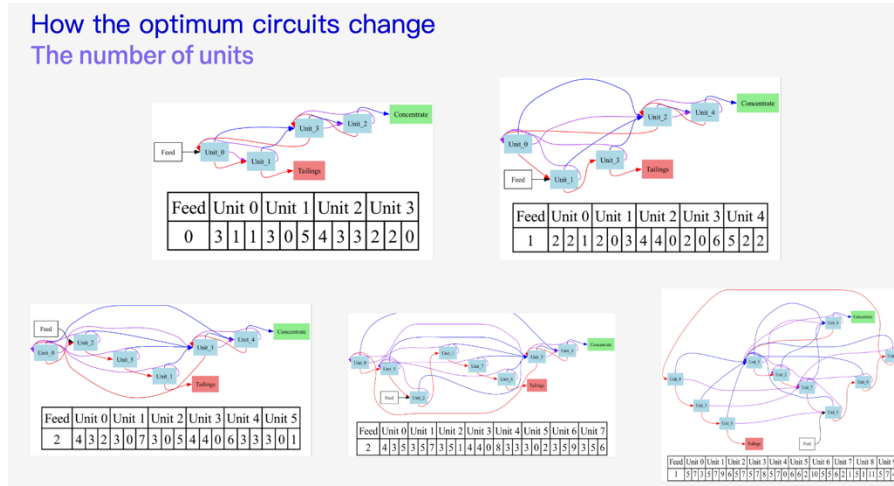
1. **Recycling and Reuse:** Higher disposal costs drive efforts to recycle materials within the process. By reprocessing waste streams, the system can extract more valuable material, reducing the amount of waste requiring costly disposal. This approach not only lowers disposal costs but also enhances the overall resource efficiency of the operation.
2. **Optimization of Resource Use:** Faced with high disposal costs, units are configured to maximize the extraction and processing efficiency of gerardium. This involves optimizing every stage of the process to ensure minimal waste generation. Techniques such as advanced separation methods and multi-stage processing become more prevalent, aiming to extract maximum value from the feed material.
3. **Complex Configurations and Increased Operational Costs:** While more complex circuits reduce waste disposal costs, they can lead to higher operational expenses. Complex systems typically require more energy, increased maintenance, and greater technical oversight. The trade-off between reduced disposal costs and higher operational expenses must be carefully managed to maintain



overall cost-effectiveness.

4. **Adaptation to Economic Pressures:** The ability to reconfigure processing circuits in response to economic changes underscores the flexibility and resilience of the production approach. This adaptability is crucial in industries with volatile market conditions or stringent environmental regulations, allowing operations to remain profitable and compliant under varying economic scenarios.

### Analysis of Circuit Changes with the Number of Units:

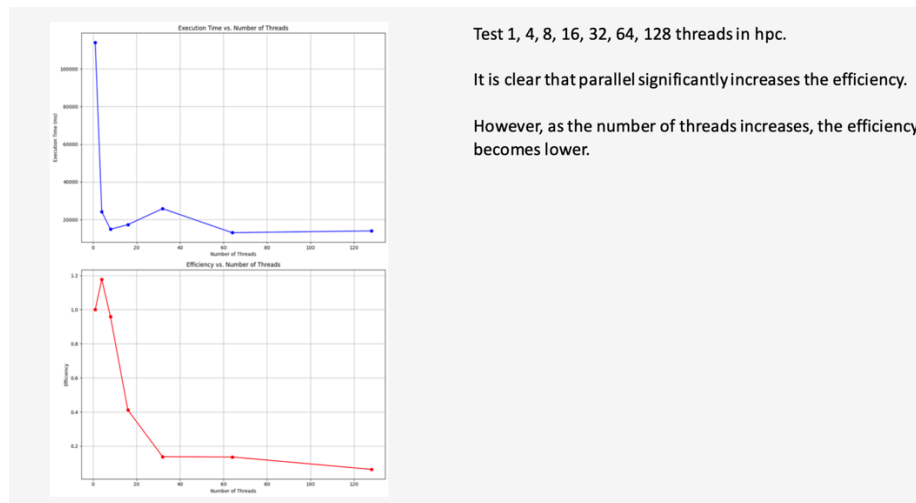


As the number of units in the circuit increases, there is a notable increase in system complexity and flexibility. This increase in units provides more options for processing paths, allowing for more optimized handling of materials based on specific criteria such as cost, efficiency, or output quality.

1. **System Complexity:** With more units, circuits exhibit increased connectivity and the presence of loops, which suggests recirculation of materials for additional processing. This complexity allows the system to adapt to different processing needs, enhancing its ability to improve yield, quality, and waste reduction.
2. **Variation in Connectivity:** More units lead to a greater number of interconnections and pathways. This increased connectivity provides multiple routes for material to travel through the system, offering flexibility in how inputs are processed to outputs. Such configurations can be dynamically adjusted to handle fluctuations in input quality or changes in desired output specifications.
3. **Strategic Use of Units:** The allocation and use of specific units within various configurations reflect strategic decisions based on unit efficiency, operational cost, and effectiveness in processing certain materials. The flexibility to adapt the circuit configuration allows for tailored approaches to optimize performance based on current economic and operational conditions.
4. **Tailings Management:** As units increase, tailings management becomes more integrated into circuit configurations. This integration suggests efforts to minimize environmental impact by extracting more value from the feed material before it reaches the tailings stage. Efficient tailings

management is crucial for reducing waste disposal costs and enhancing overall sustainability.

### Analysis of Execution Time vs. Number of Threads:



The relationship between execution time and the number of threads used in processing is pivotal in optimizing computational efficiency. The analysis reveals a clear pattern:

1. **Initial Decrease in Execution Time:** As the number of threads increases from 1 to 16, there is a significant reduction in execution time. This decrease indicates that the workload is effectively distributed among multiple threads, leveraging parallel processing to expedite computation.
2. **Plateau and Minor Fluctuations:** Beyond 16 threads, execution time stabilizes with minor fluctuations up to around 64 threads. This plateau suggests that the task has reached a level of parallelization where adding more threads no longer significantly impacts execution time. The limits of parallelism for the given workload become apparent, indicating other bottlenecks such as memory access patterns or thread management overhead.
3. **Minimal Impact Beyond Optimal Threading:** After around 64 threads, further increases in thread count have minimal impact on execution time. Factors such as thread management overhead, synchronization costs, and contention for shared resources likely contribute to this plateau. This observation highlights the importance of identifying the optimal number of threads to maximize computational efficiency without unnecessary resource consumption.

### Analysis of Efficiency vs. Number of Threads:

Efficiency provides insight into the optimal use of computational resources:

1. **Peak Efficiency:** Efficiency peaks at 4 threads, indicating an optimal balance between performance gain and resource utilization. At this point, the system achieves the highest return on investment in terms of computational resources.
2. **Sharp Decline in Efficiency:** As the number of threads increases beyond 4, there is a noticeable decline in efficiency. This decline can be attributed to the overhead associated with managing more

threads, such as context switching and synchronization costs. The contention for shared resources also plays a role, reducing the overall efficiency of the system.

3. **Low Efficiency at High Thread Counts:** At very high thread counts (e.g., 64 to 128 threads), efficiency is significantly lower. The additional threads contribute little to performance improvement but still consume computing resources, indicating diminishing returns. This scenario underscores the importance of understanding the trade-offs between parallelism and resource management to achieve optimal performance.

### **Conclusions for analysis:**

In conclusion, the balance between operational complexity, cost efficiency, and computational resource management is critical for achieving sustainable and profitable operations. The result gained from this analysis can guide the strategic decisions in circuit design and computational optimization, ensuring the system's adaptability and efficiency in varying economic conditions.