1. **Introduction**

SLEUTH is a self-modifying probabilistic cellular automata model originally developed by Keith Clarke at the University of California at Santa Barbara. It’s an abbreviation for the input data layers: Slope, Land Use, Exclusion, Urban, Transportation, and Hill Shade[[1]](#endnote-1).

There are some limitations of SLEUTH. It is single threaded and it will cost 8 days to run one step and frequently fails during that 8 days run.

Although it has already been distributed across nodes manually to reduce run time, the manual process is tedious and error prone for data management. Therefore, developing a framework for doing those things instead of manually is a good opportunity to reduce the runtime. Additionally, the underlying solver is ripe for parallelization due to natural domain decomposition of the images and task decomposition of the data layers.

DSLEUTH is the distributed framework that can break up the search space in the input parameters into several smaller one then allocate them to different nodes in the cluster to calculate it. After finishing calculating, the framework will gather the results from those nodes.

1. **Requirements**

In order to speed up the SLEUTH model run time, the most effective approach is to develop a distributed framework for this model which includes four modules (1) break up the search space of input parameters (2) launch jobs on nodes in cluster and (3) gather results from those nodes. After developing the distributed framework, (4) we need to analyze the speed up and adjust the strategy of breaking up search space to achieve the best performance.

For the break up search space part, there are several steps need to complete. First, figure out the search space of input. Second, detect how many nodes available in the cluster. Third, break the search space into reasonable number of smaller search spaces. Generally, the number of split search spaces is a number that is times of nodes and it should not be so large as well so that it won’t cost a lot in the schedule process.

For the launch jobs on nodes in cluster part, (1) we need to run some cluster management platform and tools on the cluster. (2) Using shell script to allocate resource for one job (3) Using shell script launch a job on a node.

For the gather results part, we need to gather all results from different jobs executed on different nodes into one folder.

For the analyze speed up part, we need to do experiments to record the performances in different cases and compare it against the performance of SLEUTH in the corresponding case. Additionally, we change our strategies of breaking up the search space and compare the performance between different breaking up strategies in the same case. Finally, we choose an best performance strategy for the breaking up search space.

1. **Design**

Main Idea: DSLEUTH is a distribute version of SLEUTH. It breaks the input parameters range into several consecutive smaller ranges which can be taken as a single job. Then allocate available nodes in the cluster to do those jobs separately. The main node checks other nodes to make sure whether they complete the assigned job. Once a node completes a job, the main node will allocate a new job from the job list if it has jobs left.

Structure: There has two parts in DSLEUTH. One is the module that provides functions to break the parameters range in the input file then pack the new parameters in new input files. Another one is to detect how many nodes available in the cluster then get the node list and allocate those new input file to execute. It works as a scheduler. As the storage of flux cluster is shared, the output of each node can be existed in the same directory.

1

Node1

detect

2

scheduler

Node2

Original scenario file

break into

3

allocate

Node3

…

Node M

…

put

N ……… 3 2 1

Task queue

N

For each module in design part, the design details are in here.

Module (1) break up the search space of input parameters. This module includes a set of functions. Frist, it need to read the configuration file, then get the values of each search range related parameters. Next, it need to provide different strategies for breaking up the search spaces. Totally, there has five different search spaces.

* (a): break up the Diffusion search space such that each sub-search space contains one value taken by SLEUTH and the rest keeps the same as original.
* (b): break up the Breed search space such that each sub-search space contains one value taken by SLEUTH and the rest keeps the same as original.
* (c): break up the Spread search space such that each sub-search space contains one value taken by SLEUTH and the rest keeps the same as original.
* (d): break up the Slope search space such that each sub-search space contains one value taken by SLEUTH and the rest keeps the same as original.
* (e): break up the Road search space such that each sub-search space contains one value taken by SLEUTH and the rest keeps the same as original.
* (f): break up all five different types search space and combines five sub-search spaces from each type of search space.
* (g): break up the Diffusion search space into given number of sub-search spaces.

There are examples in the appendix.

Module (2) launch jobs on nodes in cluster. First, get the number of how many sub-jobs generated by module (1). Then detect how many nodes available in the cluster for running the whole job. There need to have a task queue in which those sub-jobs generated by module (1) are put. We adopt First come First service strategy to schedule the jobs. Thereafter, for each node available in the cluster grab a sub-job if the task queue is not empty and launch that sub-job on that node. Then for each node detect whether the sub-job is completed on the node. If it is completed, then grab another sub-job from the task queue if the task queue still has sub-jobs. If the task queue is empty. Then the node turns into idle state and wait for the rest nodes complete their jobs. If every node completes the job and the task queue is empty, then the whole job is done.

Module (3) gather results from those nodes. In the module (2) each sub-job will output their result in a separate directory. Therefore, we need gather all those output into one file so that we can compare the results with the result of original SLEUTH model. First read the output from each directory that each sub-job generated, then replace the run number and delete the headers but left the first one. Next, write them into one result file.

Module (4) analyze the speed up and adjust the strategy of breaking up search space to achieve the best performance. This module is mainly doing experiments using different strategies. Record the time of each strategy and compare it with the time cost by original model for the same input. From those records, we can get the best one.

1. **Implementation**

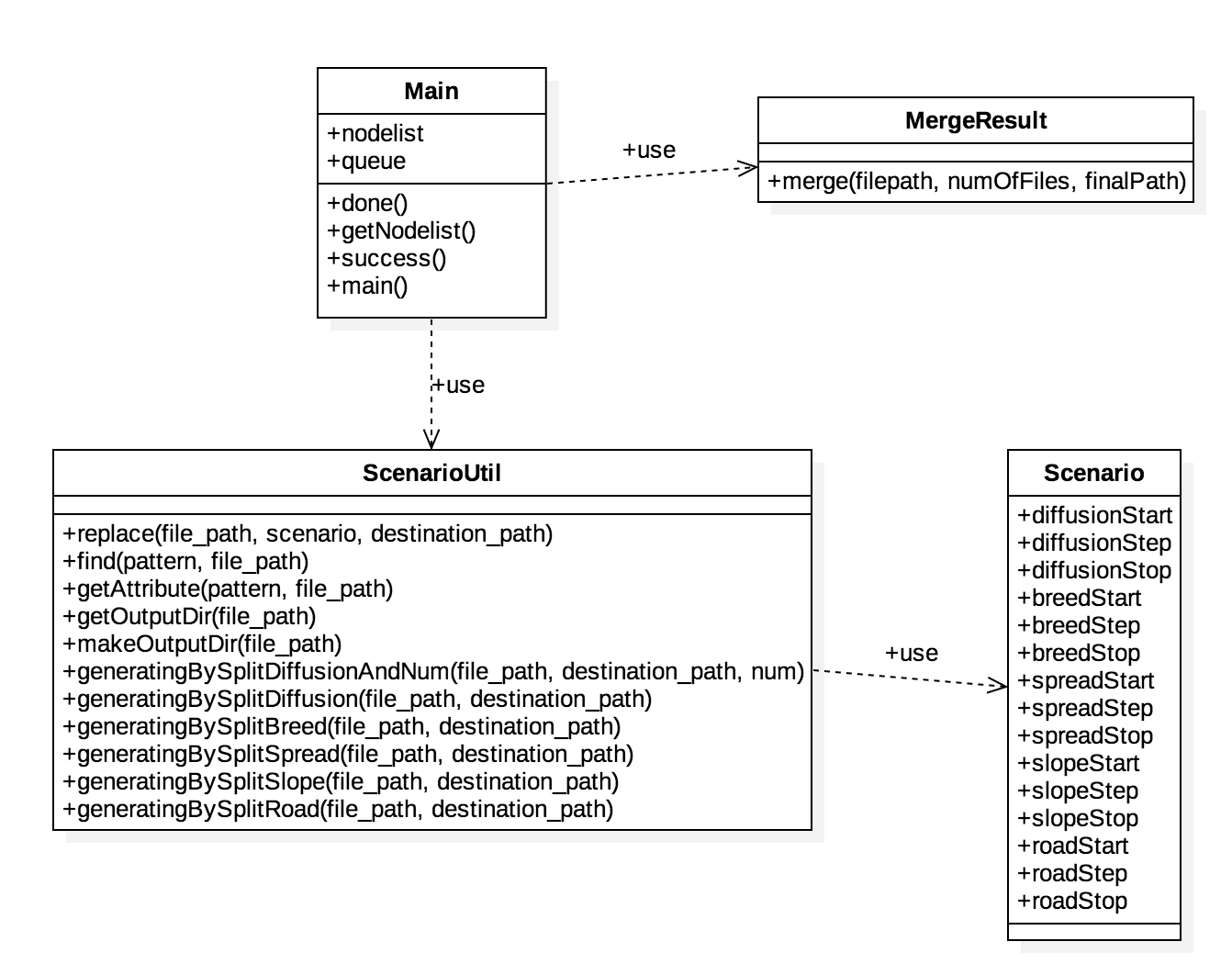
In the implementation process, we use the Iterative and incremental development methodology. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental), allowing software developers to take advantage of what was learned during development of earlier parts or versions of the system. Learning comes from both the development and use of the system, where possible key steps in the process start with a simple implementation of a subset of the software requirements and iteratively enhance the evolving versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added.

First, I implemented module (1) break up the search space. Implement all different strategies.

Second, I implemented module (2) launch jobs on nodes in cluster. First, launch one job on a specific node by using the cluster resource management tool SLURM. Then embed it in python code. Given a node and a job, this part can launch this job on the node. After that implement the function that detect whether the job is completed on the node. Then allocate the jobs

Thirdly, implement the module (3) gather results of those nodes.

Classes: There are four classes in the system: Main, Scenario, ScenarioUtil, MergeResult. The Main class correspond to the scheduler in the above graph and the ScenarioUtil class correspond to the break up search space process. The Scenario class is an entity class which represents the input parameters object that need to be broken up. Class diagram is shown as below.



1. **Results**

**Test Environment:**

UWL FLUX cluster which has one head node and 3 computation nodes, Linux system, SLURM resource management system[[2]](#endnote-2). Hardware of a compute node: CPUs: 8 x 2.00 GHz, Memory (RAM): 48 GB

**Original SLEUTH Performance:**

I ran the SLEUTH on node flux2 which is a computation node by using calibrate mode and using demo200 dataset as the input images whose total size is 101KB. I used the scenario.demo200 as the scenario file whose important parameters are as follow:

MONTE\_CARLO\_ITERATIONS = 4

CALIBRATION\_DIFFUSION\_START=0

CALIBRATION\_DIFFUSION\_STEP=25

CALIBRATION\_DIFFUSION\_STOP=100

CALIBRATION\_BREED\_START=0

CALIBRATION\_BREED\_STEP=25

CALIBRATION\_BREED\_STOP=100

CALIBRATION\_SPREAD\_START=0

CALIBRATION\_SPREAD\_STEP=25

CALIBRATION\_SPREAD\_STOP=100

CALIBRATION\_SLOPE\_START=0

CALIBRATION\_SLOPE\_STEP=25

CALIBRATION\_SLOPE\_STOP=100

CALIBRATION\_ROAD\_START=0

CALIBRATION\_ROAD\_STEP=25

CALIBRATION\_ROAD\_STOP=100

After the runs, I get the average running time of 3 times running SLEUTH by using the above input is 148mins.

**DSLEUTH Performance:**

I ran the DSLEUTH on three computation nodes flux2, flux3, flux4 by using calibrate mode as well and taking the samedataset as the input and the same scenario file.In this running, the DSLEUTH breaks the CALIBRATION\_DIFFUTION part into 5 smaller search spaces. The average running time of 3 times running SLEUTH is 60 minutes.

**Summary:**

Speedup = 148/60 = 2.45

**Appendix:**

1. Parameters of original scenario file

|  |  |
| --- | --- |
| MONTE\_CARLO\_ITERATIONS | 4 |
| CALIBRATION\_DIFFUSION\_START | 0 |
| CALIBRATION\_DIFFUSION\_STEP | 25 |
| CALIBRATION\_DIFFUSION\_STOP | 100 |
| CALIBRATION\_BREED\_START | 0 |
| CALIBRATION\_BREED\_STEP | 25 |
| CALIBRATION\_BREED\_STOP | 100 |
| CALIBRATION\_SPREAD\_START | 0 |
| CALIBRATION\_SPREAD\_STEP | 25 |
| CALIBRATION\_SPREAD\_STOP | 100 |
| CALIBRATION\_SLOPE\_START | 0 |
| CALIBRATION\_SLOPE\_STEP | 25 |
| CALIBRATION\_SLOPE\_STOP | 100 |
| CALIBRATION\_ROAD\_START | 0 |
| CALIBRATION\_ROAD\_STEP | 25 |
| CALIBRATION\_ROAD\_STOP | 100 |

1. Parameters of generated scenario files using strategy (a)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Job1 | Job2 | Job3 | Job4 | Job5 |
| MONTE\_CARLO\_ITERATIONS | 4 | 4 | 4 | 4 | 4 |
| CALIBRATION\_DIFFUSION\_START | 0 | 25 | 50 | 75 | 100 |
| CALIBRATION\_DIFFUSION\_STEP | 25 | 25 | 25 | 25 | 25 |
| CALIBRATION\_DIFFUSION\_STOP | 0 | 25 | 50 | 75 | 100 |
| CALIBRATION\_BREED\_START | 0 | 0 | 0 | 0 | 0 |
| CALIBRATION\_BREED\_STEP | 25 | 25 | 25 | 25 | 25 |
| CALIBRATION\_BREED\_STOP | 100 | 100 | 100 | 100 | 100 |
| CALIBRATION\_SPREAD\_START | 0 | 0 | 0 | 0 | 0 |
| CALIBRATION\_SPREAD\_STEP | 25 | 25 | 25 | 25 | 25 |
| CALIBRATION\_SPREAD\_STOP | 100 | 100 | 100 | 100 | 100 |
| CALIBRATION\_SLOPE\_START | 0 | 0 | 0 | 0 | 0 |
| CALIBRATION\_SLOPE\_STEP | 25 | 25 | 25 | 25 | 25 |
| CALIBRATION\_SLOPE\_STOP | 100 | 100 | 100 | 100 | 100 |
| CALIBRATION\_ROAD\_START | 0 | 0 | 0 | 0 | 0 |
| CALIBRATION\_ROAD\_STEP | 25 | 25 | 25 | 25 | 25 |
| CALIBRATION\_ROAD\_STOP | 100 | 100 | 100 | 100 | 100 |

1. Parameters of generated scenario files using strategy (b)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Job1 | Job2 | Job3 | Job4 | Job5 |
| MONTE\_CARLO\_ITERATIONS | 4 | 4 | 4 | 4 | 4 |
| CALIBRATION\_DIFFUSION\_START | 0 | 0 | 0 | 0 | 0 |
| CALIBRATION\_DIFFUSION\_STEP | 25 | 25 | 25 | 25 | 25 |
| CALIBRATION\_DIFFUSION\_STOP | 100 | 100 | 100 | 100 | 100 |
| CALIBRATION\_BREED\_START | 0 | 25 | 50 | 75 | 100 |
| CALIBRATION\_BREED\_STEP | 25 | 25 | 25 | 25 | 25 |
| CALIBRATION\_BREED\_STOP | 0 | 25 | 50 | 75 | 100 |
| CALIBRATION\_SPREAD\_START | 0 | 0 | 0 | 0 | 0 |
| CALIBRATION\_SPREAD\_STEP | 25 | 25 | 25 | 25 | 25 |
| CALIBRATION\_SPREAD\_STOP | 100 | 100 | 100 | 100 | 100 |
| CALIBRATION\_SLOPE\_START | 0 | 0 | 0 | 0 | 0 |
| CALIBRATION\_SLOPE\_STEP | 25 | 25 | 25 | 25 | 25 |
| CALIBRATION\_SLOPE\_STOP | 100 | 100 | 100 | 100 | 100 |
| CALIBRATION\_ROAD\_START | 0 | 0 | 0 | 0 | 0 |
| CALIBRATION\_ROAD\_STEP | 25 | 25 | 25 | 25 | 25 |
| CALIBRATION\_ROAD\_STOP | 100 | 100 | 100 | 100 | 100 |

1. Parameters of generated scenario files using strategy (c)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Job1 | Job2 | Job3 | Job4 | Job5 |
| MONTE\_CARLO\_ITERATIONS | 4 | 4 | 4 | 4 | 4 |
| CALIBRATION\_DIFFUSION\_START | 0 | 0 | 0 | 0 | 0 |
| CALIBRATION\_DIFFUSION\_STEP | 25 | 25 | 25 | 25 | 25 |
| CALIBRATION\_DIFFUSION\_STOP | 100 | 100 | 100 | 100 | 100 |
| CALIBRATION\_BREED\_START | 0 | 0 | 0 | 0 | 0 |
| CALIBRATION\_BREED\_STEP | 25 | 25 | 25 | 25 | 25 |
| CALIBRATION\_BREED\_STOP | 100 | 100 | 100 | 100 | 100 |
| CALIBRATION\_SPREAD\_START | 0 | 25 | 50 | 75 | 100 |
| CALIBRATION\_SPREAD\_STEP | 25 | 25 | 25 | 25 | 25 |
| CALIBRATION\_SPREAD\_STOP | 0 | 25 | 50 | 75 | 100 |
| CALIBRATION\_SLOPE\_START | 0 | 0 | 0 | 0 | 0 |
| CALIBRATION\_SLOPE\_STEP | 25 | 25 | 25 | 25 | 25 |
| CALIBRATION\_SLOPE\_STOP | 100 | 100 | 100 | 100 | 100 |
| CALIBRATION\_ROAD\_START | 0 | 0 | 0 | 0 | 0 |
| CALIBRATION\_ROAD\_STEP | 25 | 25 | 25 | 25 | 25 |
| CALIBRATION\_ROAD\_STOP | 100 | 100 | 100 | 100 | 100 |

1. Parameters of generated scenario files using strategy (d)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Job1 | Job2 | Job3 | Job4 | Job5 |
| MONTE\_CARLO\_ITERATIONS | 4 | 4 | 4 | 4 | 4 |
| CALIBRATION\_DIFFUSION\_START | 0 | 0 | 0 | 0 | 0 |
| CALIBRATION\_DIFFUSION\_STEP | 25 | 25 | 25 | 25 | 25 |
| CALIBRATION\_DIFFUSION\_STOP | 100 | 100 | 100 | 100 | 100 |
| CALIBRATION\_BREED\_START | 0 | 0 | 0 | 0 | 0 |
| CALIBRATION\_BREED\_STEP | 25 | 25 | 25 | 25 | 25 |
| CALIBRATION\_BREED\_STOP | 100 | 100 | 100 | 100 | 100 |
| CALIBRATION\_SPREAD\_START | 0 | 0 | 0 | 0 | 0 |
| CALIBRATION\_SPREAD\_STEP | 25 | 25 | 25 | 25 | 25 |
| CALIBRATION\_SPREAD\_STOP | 100 | 100 | 100 | 100 | 100 |
| CALIBRATION\_SLOPE\_START | 0 | 25 | 50 | 75 | 100 |
| CALIBRATION\_SLOPE\_STEP | 25 | 25 | 25 | 25 | 25 |
| CALIBRATION\_SLOPE\_STOP | 0 | 25 | 50 | 75 | 100 |
| CALIBRATION\_ROAD\_START | 0 | 0 | 0 | 0 | 0 |
| CALIBRATION\_ROAD\_STEP | 25 | 25 | 25 | 25 | 25 |
| CALIBRATION\_ROAD\_STOP | 100 | 100 | 100 | 100 | 100 |

1. Parameters of generated scenario files using strategy (e)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Job1 | Job2 | Job3 | Job4 | Job5 |
| MONTE\_CARLO\_ITERATIONS | 4 | 4 | 4 | 4 | 4 |
| CALIBRATION\_DIFFUSION\_START | 0 | 0 | 0 | 0 | 0 |
| CALIBRATION\_DIFFUSION\_STEP | 25 | 25 | 25 | 25 | 25 |
| CALIBRATION\_DIFFUSION\_STOP | 100 | 100 | 100 | 100 | 100 |
| CALIBRATION\_BREED\_START | 0 | 0 | 0 | 0 | 0 |
| CALIBRATION\_BREED\_STEP | 25 | 25 | 25 | 25 | 25 |
| CALIBRATION\_BREED\_STOP | 100 | 100 | 100 | 100 | 100 |
| CALIBRATION\_SPREAD\_START | 0 | 0 | 0 | 0 | 0 |
| CALIBRATION\_SPREAD\_STEP | 25 | 25 | 25 | 25 | 25 |
| CALIBRATION\_SPREAD\_STOP | 100 | 100 | 100 | 100 | 100 |
| CALIBRATION\_SLOPE\_START | 0 | 0 | 0 | 0 | 0 |
| CALIBRATION\_SLOPE\_STEP | 25 | 25 | 25 | 25 | 25 |
| CALIBRATION\_SLOPE\_STOP | 100 | 100 | 100 | 100 | 100 |
| CALIBRATION\_ROAD\_START | 0 | 25 | 50 | 75 | 100 |
| CALIBRATION\_ROAD\_STEP | 25 | 25 | 25 | 25 | 25 |
| CALIBRATION\_ROAD\_STOP | 0 | 25 | 50 | 75 | 100 |

**References**

1. *2009 Urban Remote Sensing Joint Event Simulate urban growth based on RS, GIS, and SLEUTH model in Shenyang-Fushun metropolitan area northeastern China* [↑](#endnote-ref-1)
2. *https://slurm.schedmd.com/* [↑](#endnote-ref-2)