

The FISHNET Case Study on Implementing and Scaling a Complex Earth Observation Workflow

Lorenz Gruber

Dept. of Computer Science
Julius Maximilian University
Würzburg, Germany
ORCID: 0009-0004-4343-6864

Nikolas Herbst

Dept. of Computer Science
Julius Maximilian University
Würzburg, Germany
ORCID: 0000-0003-3462-6426

Peter Friedl

German Remote Sensing Data Center
German Aerospace Center (DLR)
Wessling, Germany
ORCID: 0009-0000-9891-6655

Thanh Thi Nguyen

Dept. of Organic Plant Production
and Agroecosystems, University of Kassel
Kassel, Germany
ORCID: 0000-0003-1037-9731

Thomas Esch

German Remote Sensing Data Center
German Aerospace Center (DLR)
Wessling, Germany
ORCID: 0000-0002-5868-9045

Samuel Kounev

Dept. of Computer Science
Julius Maximilian University
Würzburg, Germany
ORCID: 0000-0001-9742-2063

Abstract—Earth observation (EO) scientists use sophisticated algorithms, large datasets, and high-performance data analytics (HPDA) clusters to develop and execute complex EO workflows. To improve portability and reusability within the domain, the Open Geospatial Consortium (OGC) published a set of best practices for developing EO workflows. Even though EO data products are often shared openly, representative EO workflow implementations are hardly available.

In this paper, we contribute a case study on implementing and scaling a complex EO workflow, adhering to the OGC best practices and demonstrating its portability by deploying it both locally and on the HPDA platform “terrabyte” at the Leibniz Supercomputing Centre. Contentwise, the contributed workflow analyzes settlement patterns by first delineating coherent settlements derived from leveraged satellite data that maps built-up areas and subsequently calculating centrality measures to characterize the spatial arrangement and hierarchy of settlements in a given region.

We demonstrate the workflow’s scalability and variance in resource demands by analyzing its time-to-result, total CPU time, and resource efficiency under different inputs and configurations. Interestingly, implicit parameters hidden in the input data semantics, like the number and area of settlements in the region of interest, significantly impact the time required to complete the processing. Concurrent processing is restricted to connected components of the settlement graph in the analysis stage, leading to an unbalanced workload distribution when analyzing large urban areas, showcasing the scalability challenges EO scientists face.

Index Terms—Scientific Workflows, Earth Observation, Workflow Engineering, Performance Measurements

I. INTRODUCTION

A. Context

Earth Observation (EO) scientists face numerous challenges when developing and instantiating a workflow. In addition to domain-specific hurdles, researchers must also deal with complex computer-science-related problems to efficiently process the enormous data volumes (up to petabytes) omnipresent in remote sensing applications [1]. Consequently, EO scientists

without professional IT skills often resort to tedious manual or semi-automated approaches to realize their workflows [2]. Moreover, EO processing pipelines are usually developed from scratch according to the domain-specific requirements and not considered or formalized as workflows, resulting in limited reusability and reproducibility due to the lack of common abstractions. Even though EO data products are often shared openly, representative EO workflow implementations are hardly shared in a findable way. One of the biggest cross-domain workflow registries, the *WorkflowHub* [3], with almost 800 shared workflows, only features two EO-related workflows up to now. To combat this issue, the Open Geospatial Consortium (OGC) introduced best practices for the development and packaging of EO workflows [4], building on the Common Workflow Language (CWL) [5] to abstract individual processing steps and ensure the portability, isolation, and reproducibility of workflow executions. However, the adoption of CWL within the EO domain is still limited, with EO exploitation platforms providing no native support for executing CWL workflows. An example of an EO exploitation platform we use in this paper is *terrabyte* [6]—a high-performance data analytics (HPDA) platform developed as part of a long-term collaboration between the Leibniz Supercomputing Centre and the German Aerospace Center. It comprises a powerful compute infrastructure with 1.3 petaFLOPS, 60 GPU accelerators, InfiniBand networking at 320 Gb/s, 50 petabytes of storage, and provides on-demand remote development services and rapid data access through an STAC [7] catalog.

B. Motivation

To investigate the challenges in modeling an EO workflow with CWL and executing it on an HPDA¹ platform, we developed a complex EO workflow featuring multiple stages, explained in detail in Section III. The workflow aims to

¹ Definition of HPDA <https://www.weka.io/learn/glossary/ai-ml/hpda/>

contribute to state-of-the-art research within the settlement pattern analysis domain, studying a realistic application scenario. In the research field of settlement pattern analysis, geoscientists are interested in the extent and distribution of settlements as well as the spatial relationships between them [8]. Current research often depends on remote sensing data acquired by satellites and leveraged through image processing techniques [9]. Due to the improved resolution of satellite images, a single large urban area is mapped to multiple disjoint settlement patches separated by roads or natural obstacles like rivers, ultimately distorting the analysis of settlement patterns. Consequently, individual settlements must be aggregated into urban areas before conducting the analysis. We propose a graph-based delineation of settlements, joining neighboring settlements with the edge contraction procedure. The subsequent analysis computes selected centrality measures upon the contracted settlement graph, allowing domain experts to identify important settlements within an area.

C. Contribution

We provide a case study on a realistic multi-stage scientific EO workflow with varying parallelism, modeled according to state-of-the-art best practices. The workflow implements a novel graph-based approach to delineate urban areas, leveraging the graph data structure to compute domain-specific centrality measures. The parallel execution of independent CWL jobs enables the procedure to process the most complex urban areas while yielding reproducible results. However, we also identify challenges to model arbitrary job dependencies in CWL, requiring a proprietary scheduler to orchestrate the workflow. In addition to the workflow's deployability on an HPDA system, we investigate the performance characteristics of the proposed multi-stage workflow to gain insights into the resource demands of different stages and the impact of the chosen input region and applied configuration on the time to result (TTR). Moreover, we analyze the contribution of each workflow stage to the total processing time. These measurements provide valuable insights into the performance and resource demand profile of a typical EO workflow, supporting further endeavors towards dynamic resource allocation in the context of scientific workflows.

II. BACKGROUND AND RELATED WORK

This section introduces best practices for constructing a workflow within the EO domain before reviewing related research in benchmarking scientific workflows. After that, we highlight the workflow's input dataset and related approaches to delineate urban areas and analyze settlement centrality.

A. OGC Best Practices and CWL

The Open Geospatial Consortium released best practices for modeling and packaging workflows to foster their reusability. It requires EO workflows to be executable as a non-interactive command line tool and be delivered in a container image with all necessary software, libraries, and configuration files [4]. Furthermore, the input and output of the application have to be

formally defined, for example, using STAC (SpatioTemporal Asset Catalog [7]). The OGC best practices for packaging an EO application require a valid CWL document formalizing a workflow's interface and processing steps. In addition to the input and output schema of the wrapped command line executable, the *.cwl* file may define a container image as an execution environment and capture log files [5]. A CWL-compliant runner then binds values to the input variables, defined in the *.cwl* file, and executes the workflow in the specified environment. The binding of input parameters can be accomplished directly through command line options of the runner or by supplying an additional JSON-based *job* file initializing the input variables. The reference CWL-runner *cwltool* [10] is maintained by the community and suited for the local execution of CWL jobs. However, more specialized CWL-compliant runners exist that are better suited for their respective target platforms. The *Toil* CWL-runner, for example, simplifies the deployment of CWL tasks on HPC systems [11] while *Calrissian* [12] enables the execution of CWL workflows in a *Kubernetes* cluster.

B. Benchmarking Scientific Workflows

Measuring the resource demand and time-to-result of scientific workflows on HPDA systems provides essential insights for the platform operators. The framework *WfCommons* provides tools for tracing or simulating workflow execution, and it additionally enables the generation and benchmarking of synthetic scientific workloads within the *WfBench* component [13], [14]. Additionally, the *Workflow Trace Archive* allows users to upload traces of real-world workflow runs, providing a detailed analysis of the resource demand profile and workflow progression [15]. Although featuring almost 100 entries, no EO workflow trace is available. By observing the resource demands on the execution platform, Rohwer et al. identified phases of a real-world EO workflow, gaining insights into the resource demand profile [16]. The previously mentioned *WorkflowHub* [3] features a large archive of workflows from various domains; however, it currently only contains two EO workflows [17], [18]. Lehmann et al. measured the efficiency and execution time of a production EO workflow ported to the scientific workflow system *Nextflow* [19]. Tian et al. orchestrated a deep learning EO workflow on an HPC platform with *Apache Airflow*, showcasing the linear scalability of the approach in the training time per epoch and the number of available nodes.

However, only a few detailed performance evaluation studies of real-world EO workflows have been published. In this paper, we provide in-depth performance measurements across multiple, heterogeneous workflow stages in Section IV.

C. WSF 2019 Dataset

The binary build-up layer of the World Settlement Footprint (WSF) 2019 [20] provides the initial spatial information about settlements that the workflow requires. The dataset is arranged as a grid of files, each covering roughly a surface area on Earth of two degrees longitude by two degrees latitude.

The settlement coverage data is stored in a *GeoTIFF* file, each pixel representing ten by ten meters on the Earth's surface.

D. Settlement Pattern Analysis and Urban Area Delineation

Related research in the geographic domain of the proposed workflow includes settlement pattern analysis and the delineation of settlement boundaries.

Common procedures in settlement pattern analysis include statistical [21] or manual [2] methods to classify settlement types and centrality analysis to identify important agglomerations in a region. The latter is usually based on centrality measures, requiring the construction of a settlement graph. Esch et al. introduced several domain-specific centrality measures that also take spatial relationships into account [22]. However, the selected graph construction procedure faced scalability challenges due to the quadratic asymptotic complexity in the number of settlements.

Administrative city borders, for example, available in UrbanAudit 2021 for Europe [23], define the official extent of a settlement. However, as exemplified by Taubenböck et al., these administrative delineations of a city rarely match its actual extent [24]. The authors propose a geometric approach based on circle sectors to approximate the dimensions of an urban area. More modern approaches resort to statistical, machine-learning methods to delineate settlements [25], [26]. As a novelty within the domain, we introduce a graph-based method to delineate the extent of urban areas. This approach is more flexible than established methods, since the criteria for determining neighboring settlements can easily be swapped, while centrality measures can be computed directly on the settlement graph.

III. SETTLEMENT DELINEATION AND ANALYSIS WORKFLOW

In this section, we illustrate the modeling, workload parallelization, and orchestration as well as the deployment of the *Settlement Delineation and Analysis (SDA)* workflow. The source code is available in the *Fishnet* repository [27].

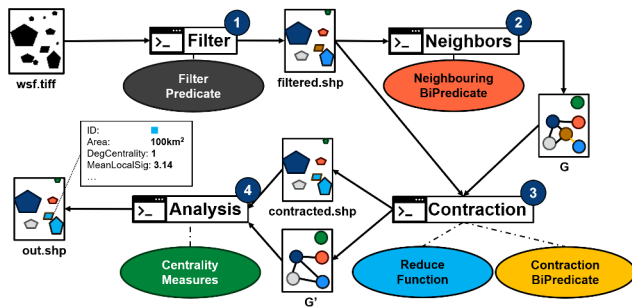


Fig. 1. Processing steps of the SDA workflow.

A. Modeling of Workflow Stages

The workflow comprises four main stages, each implemented as a standalone executable, with the interface wrapped in a CWL definition file. Figure 1 illustrates these steps for a

single input along with the artifacts produced by each stage. At first, the raster input is polygonized with the library GDAL [28], and settlements not fulfilling the *Filter Predicate* are dropped. In the subsequent stage, the adjacencies of the settlements are computed according to a user-configurable criterion. It uses an adapted plane sweep [29] to avoid unnecessary comparisons between far-apart settlements. The settlement graph is then contracted with edges fulfilling the *Contraction BiPredicate* being removed from the graph and the incident vertices merged. The connectivity to former neighbors of aggregated vertices is restored afterwards.

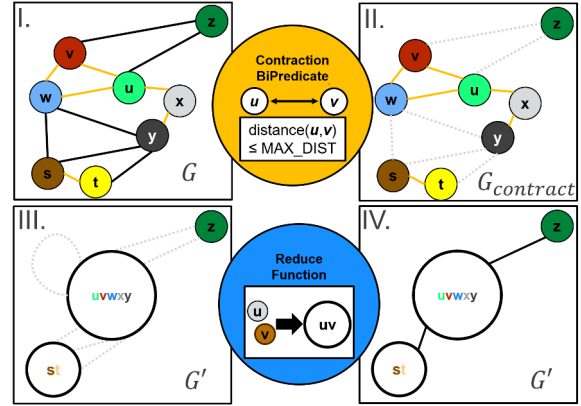


Fig. 2. Illustration of the edge contraction procedure, optimized through aggregating connected components in the contractable graph first and then reducing the vertices of a component to a single vertex.

Figure 2 illustrates this procedure with the contraction criterion dependent on the distance between vertices and the reduce function combining the vertices. The behavior of these functions can be swapped easily in the implementation. To improve the performance of this stage, the *Producer-Consumer* pattern [30] is applied with an adapted breadth-first search [31], uncovering components in the contractable subgraph while consumers concurrently reduce these clusters to a single vertex. In the provided visualizations, the reduction function aggregates the settlements of a cluster into a single multi-polygon. The final analysis stage computes selected centrality measures on the contracted graph and stores the centrality values for each polygon in the output shapefile. Optionally, this stage can visualize the graph's edges in a separate output file. It utilizes a thread pool for code-level parallelization to compute centrality measures and visualize the settlements and edges concurrently.

Across the stages, the SDA workflow produces intermediate results, which can be leveraged after a failure to restart the computation at that stage. Figure 3 exemplifies the progress of the workflow for the city of Nuremberg. After the filtering step, the raster input from WSF 2019 is polygonized with a unique identifier assigned to each polygon, visualized in different colors in the image. Additionally, insignificant settlements are dropped at this stage. After the contraction stage, the disjoint settlement patches are aggregated into a single multi-polygon.

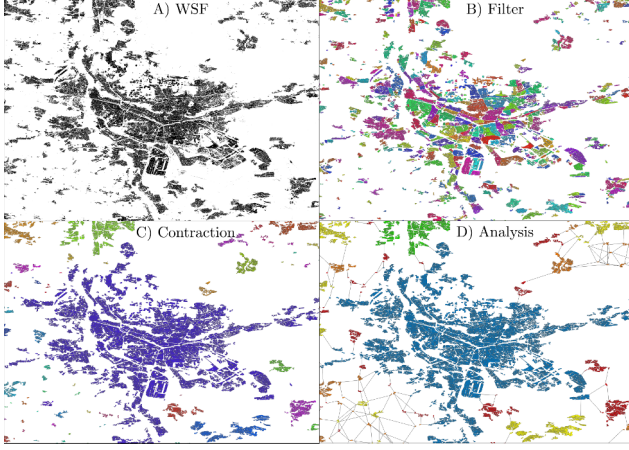


Fig. 3. The city of Nuremberg after different workflow stages. In B) and C), the polygons are colored according to their identifier; D) features a gradual coloring dependent on centrality.

The final result in Figure 3.D) contains the visualized edges and aggregated settlements, which are gradually colored from blue to red according to their degree centrality value.

B. Workflow Orchestration

To ensure the scalability of the workflow, the procedure requires workload distribution to enable parallel processing. A workload is abstracted as a CWL job, processing a partition of a stage's workload. Since the WSF dataset comes as a grid of files, workload splitting on the level of input files is natural. As illustrated in Figure 4, the first two stages benefit from this approach. While the polygonization and filtering can be processed independently for each input file, the subsequent `neighbors` stage depends on the filtered *Shapefiles* of bordering input regions since settlement adjacencies might cross file boundaries.

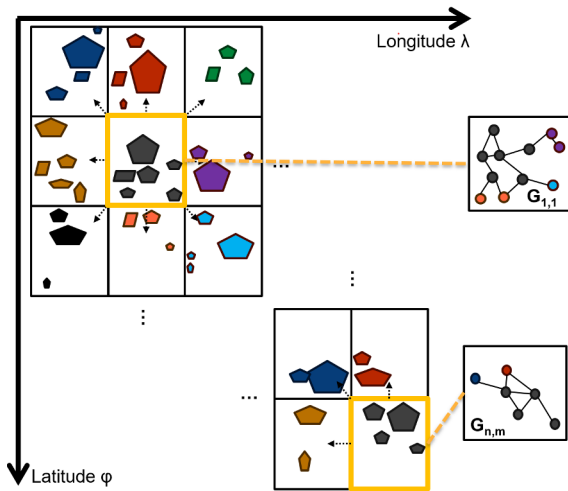


Fig. 4. Workload distribution of the `Filter` and `Neighbors` stage through file-level parallelization.

Depending on the user-defined contraction criterion, a connected component of the settlement graph might be reduced to a single vertex. However, since a connected component of the settlement graph can span multiple input files, parallelization on the level of input files is no longer applicable. Therefore, the workload of the `Contraction` and `Analysis` stages is distributed concerning connected components instead, ensuring a correct result despite concurrent execution. The workflow features a synchronization point after the `neighbors` stage completes, and the administrative components stage reveals the connected components of the settlement graph and generates the jobs for subsequent stages, as illustrated in Figure 5.

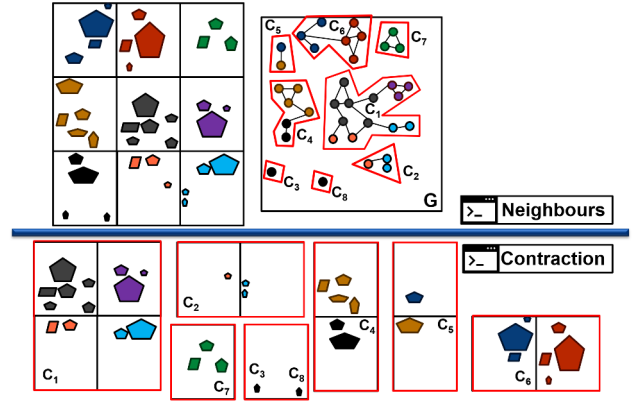


Fig. 5. Workload distribution of the `Contraction` and `Analysis` stage on the level of the settlement graph's connected components. Each workload contains only the settlement polygons part of the connected component(s).

Additional orchestrational tasks include the `split` and `merge` stages, initially further splitting the input and ultimately merging the analysis output, respectively. Domain-specific parameters and administrative settings are supplied to each stage through a configuration file in *JSON* format. A single management task orchestrates the workflow, generating the CWL jobs and modeling their interdependencies in a directed acyclic graph (DAG). Since the scatter feature of CWL is insufficient to model the complex job relationships, the workflow comes with a proprietary scheduler, leveraging the DAG to identify concurrently runnable workloads. It submits the jobs to the configured executor, which executes the CWL job with its respective runner in a separate child process. The SDA workflow stores its DAG and the evolving settlement graph in a central graph database (Memgraph [32]), which synchronizes the updates of concurrently running jobs. Race conditions are avoided through workload dependencies in the DAG and synchronized updates to the data structure itself.

During the workflow execution, the time-to-result of each job is logged, as well as additional settlement graph properties. The performance analysis in Section IV leverages these log files. Figure 6 provides an example of the Job DAG, highlighting the stage interdependencies and the change in parallelization level. In the example, the four split files are all neighboring each other; therefore, the `neighbors`

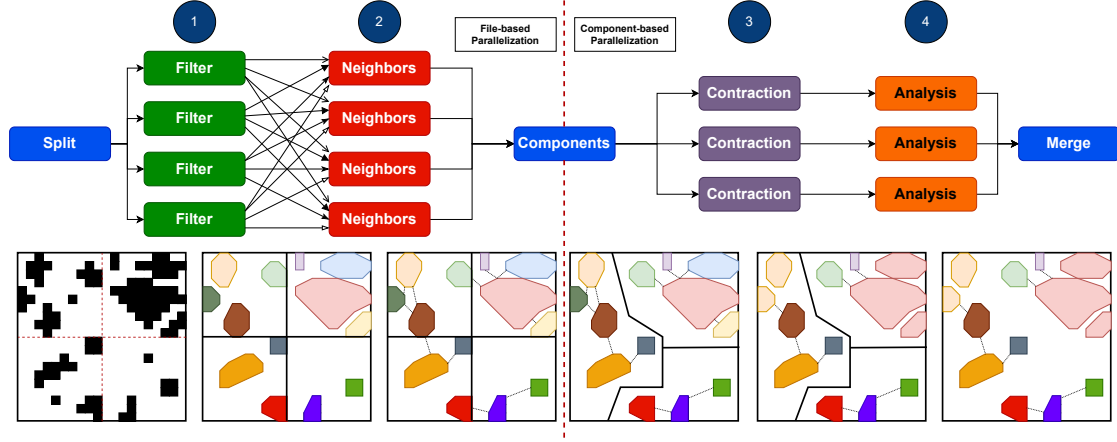


Fig. 6. Exemplary Job DAG of the SDA workflow with intermediate results.

jobs have to wait for the `filter` stage to complete. After partitioning according to connectivity, the `contraction` and `analysis` stages of every connected component are executed independently. While the workflow can be scaled using a finer-grained image splitting for the first two stages, its scalability is ultimately limited by the number of connected components, determined by the structure of the settlement graph at runtime.

C. Deployment and Configuration

Coherent with OGC best practices, the workflow comes packaged in a container with the necessary dependencies installed. The graph database *Memgraph* [32] is deployed in another container to separate concerns. A *docker-compose* file wires both instances into a single application. The workflow's executables can also be installed natively using the tool *CMake*, requiring a running instance of *Memgraph* to execute a workflow run. The third deployment option uses user-level *Charliecloud* containers, required for a containerized deployment on the HPDA platform *terabyte*.

The user can adapt the workflow processing to the region of interest through multiple configuration options. These are provided to the SDA workflow in a *json* configuration file. In addition to operational settings, the user can configure the domain-related properties shown in Table I. The upcoming performance measurements vary the input region and the configuration parameters below.

TABLE I
DOMAIN-RELATED CONFIGURATION OPTIONS OF THE SDA WORKFLOW.

Parameter	Unit	Description
Required Area	m^2	Minimum area for settlement to be considered
Max Neighbors	-	Maximum number of neighbors per settlement
Edge Distance	m	Maximum distance for settlements to be adjacent
Contraction Distance	m	Maximum distance for settlements to be merged

IV. WORKFLOW PERFORMANCE MEASUREMENTS

To satisfy the resource requirements of the SDA workflow for large-scale analysis, the procedure was deployed on the *terabyte* HPDA system. The platform restricts user privileges on the system, prohibiting *Docker* containers from executing since these inherently require superuser rights. Therefore, *Charliecloud* is chosen as a non-privileged alternative [33], packaging the workflow executables and executing the workflow in a containerized environment. Both the database and the SDA workflow come packaged in a separate *Charliecloud* container. A single *Slurm* [34] script starts the database and executes the workflow according to the supplied input and configuration. Additionally, computing resources, like the number of CPU cores or the amount of memory, must be statically allocated beforehand. The input files were split into 36 to 64 tiles, and the number of reserved CPU cores matches this value for maximum parallelization during the first workflow stages. By default, 100 GB of main memory was reserved for a run, increased to 1 TB for densely-populated regions. Joint with log files of each workflow job, the final *Slurm* report yields the performance characteristics of a workflow run. Additional measurements regarding I/O usage and the utilization profile of processing and memory resources were not feasible on the chosen HPDA platform.

A. Test Setup

Each experiment is conducted with a single WSF file as input. The following enumeration lists the experimental setups and their constant configurations, which are not altered during the measurements. We chose the domain-specific distances according to related research in settlement pattern analysis [22].

1) *VaryingInput*: This test setup changes the WSF input file while keeping the configuration constant across the different runs. Only orchestration-related parameters are adapted according to the complexity of the input file. The input file is split into 36 partitions, with the most complex region *Shanghai* being split into 64 tiles. The hardware concurrency

is set at 64 across all experiments. All settlements with an area less than $5,000 m^2$ are removed. Settlements less than $3,000 m$ apart are considered adjacent, with $500 m$ being the maximum contraction distance. Except for the densely populated *Shanghai* with 3; each settlement connects to at most 5 neighbors. Table II lists the chosen input regions, differing immensely in settlement complexity and distribution. The WSF tiles are available online on the EOC website ².

TABLE II
SELECTED INPUT REGIONS WITH VARYING COMPLEXITY

Name	WSF Tile	Description
Alps	10_46	Sparse settlements, clustered in the valleys of the Alps
Bavaria	10_48	Dense network of smaller settlements, few large cities
Greenland	—46_60	Isolated, few small villages, mostly empty
Ruhr Area	6_50	Dense network of medium to large settlements
Shanghai	120_30	Enormous urban area, covering the whole tile

2) *VaryingConfig*: This experimental setup investigates the impact of the provided configuration options. The experiments analyze the workflow’s runtime and properties of the settlement graph. All workflow runs receive the *Bavaria* WSF tile as input. Orchestration-related parameters are constant with the hardware concurrency set to 40, and the input is split into a grid of 36 files. For each run, the workflow runs from the initial splitting to the final merge stage, with edges visualized in a separate file and all 3 three available centrality measures computed. The baseline run discards settlements with an area below $5,000 m^2$ and allows at most 5 neighbors per vertex. Settlements are considered adjacent if their spatial separation is below $3,000 m$, and an edge in the settlement graph is contracted if the distance between the incident settlement polygons is less than $500 m$. The experiments vary the following configuration options, with the baseline value underlined.

- Required Area: $\underline{5,000 m^2}$; $10,000 m^2$; $25,000 m^2$
- Max Neighbors: 1, 3, $\underline{5}$, 10
- Edge Distance: $1,000 m$; $\underline{3,000 m}$; $5,000 m$
- Contraction Distance: $200 m$; $\underline{500 m}$; $1,000 m$

B. Determinism and Runtime Variance

To rule out inconsistencies in the orchestration and prove the deterministic behavior of the workflow, three independent runs were executed with input and configuration matching the baseline run of the *VaryingConfig* setup. The resulting output files of the three runs are completely identical, matching in file sizes, geometries, and settlement graph characteristics. The three runs have an almost identical time to result, varying only by seconds, which yields a deviation from the average completion time of less than 1% for every run.

²WSF 2019 Dataset <https://download.geoservice.dlr.de/WSF2019/>

C. Execution Time of Workflow Stages

This experiment analyzes the minimum, average, and maximum execution time of every workflow stage. Both experimental setups are tested, providing a good variation in the settlement distribution and workflow configurations. The results are visualized in Figure 7, showing the computation time distribution of each workflow step. The initial *split* and final *merge* workflow tasks have a high average computation time in minutes, up to over half an hour, since they do not benefit from workload distribution, featuring only code-level parallelization. The *components* task processes the whole settlement graph sequentially without requiring any settlement outlines, relying solely on the indices stored in the database. Consequently, it features a lower average computation time than the stages above, although it operates on the whole dataset. All other workflow stages usually operate on a fraction of the input size. The *neighbors* task takes the longest on average, while the *filter* task is the fastest. The worst-case runtime of the *contraction* stage and especially the *analysis* step exceeds the maximum execution time of the *neighbors* task by up to an order of magnitude. This occurs when large urban areas, such as *Shanghai*, are analyzed. Due to the high connectivity of the settlement graph, these jobs process a single large component. The possibly heavily unbalanced data distribution after the switch in parallelization level to connected components results in single jobs restraining the progress of the whole workflow. This effect also has a massive impact on the resource utilization of the workflow, discussed in Subsection IV-F.

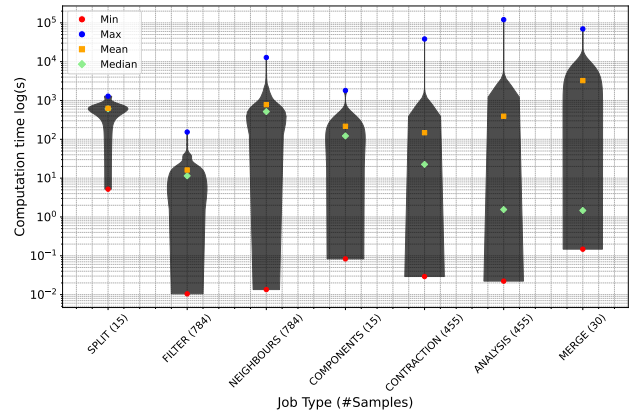


Fig. 7. Execution time distribution of each workflow stage plotted on a logarithmic time scale.

D. Impact of Input Region

To investigate the impact on resource demand of the workflow’s input region, the *VaryingInput* setup, introduced in Section IV-A, features five different regions varying significantly in settlement patterns. It includes regions featuring an even distribution of settlements, like *Bavaria*, and unbalanced areas like the *Alps*, concentrating human build-up in the

valleys. The *Ruhr Area* is more densely populated with larger urban areas, while the enormous settlement area of *Shanghai* represents the most extreme case. The orthogonal scenario to the immense urban area of *Shanghai* is the remote island *Greenland*, featuring only small and isolated settlements.

TABLE III
SETTLEMENT GRAPH CHARACTERISTICS OF WORKFLOW RUNS ON
DIFFERENT INPUT REGIONS, SORTED BY TTR.

Region	TTR	Nodes	Edges	Cluster	Nodes'
Greenland	0:01:41	371	2,562	27	47
Alps	0:16:08	9,528	62,355	148	2,198
Bavaria	3:11:02	25,876	171,488	8	6,095
Ruhr Area	12:31:34	34,750	279,869	8	4,668
Shanghai	2-17:56:59	98,895	667,353	11	1,378

As expected, the densely populated metropolitan area of *Shanghai*, spanning over an entire WSF tile, is the most demanding in computation time, taking about two days and 18 hours to complete. Also, the *Ruhr Area* workflow run is exceptionally demanding, with a time to result of twelve and a half hours. With decreasing settlement density, the time to complete the SDA workflow drops, requiring only 15 minutes when analyzing the alpine region around the city of *Innsbruck*.

Apart from the time to result, the settlement patterns in the input region also significantly impact the properties of the computed graph, which is outlined in Table III. The remote region in *Greenland* and the metropolitan area of *Shanghai* differ enormously in graph complexity due to the contrasting settlement landscape. With the population concentrated in disconnected valleys, the number of clusters, i.e., connected components of the settlement graph, in the *Alps* region is significantly higher compared to the neighboring *Bavaria*. *Greenland*, on the other hand, highlights the impact of the settlement density on the number of clusters in the graph, with a disproportionately high amount of isolated components compared to more densely populated areas, featuring fewer graph components. The impact of the contraction stage is showcased when comparing the evenly populated *Bavaria* with the more urbanized *Ruhr Area* region. Although the *Ruhr Area* region had almost 9,000 *m* more initial vertices than *Bavaria*, its contracted graph features fewer settlement vertices than the settlement graph computed in the *Bavaria* region. In summary, the settlement count, density, and distribution significantly impact the characteristics of the settlement graph and, therefore, the TTR of the workflow.

E. Impact of Workflow Configuration

To measure the impact of selected configuration options on the workflow's completion time, the *VaryingConfig* test setup is utilized. In addition to the baseline, a comprehensive run is executed for each varying setting, changing only one configuration option at a time. Figure 8 shows each workflow instance's CPU time, highlighting the CPU time share for each workflow stage. In addition to the previously outlined workflow stages, the *orchestration* stage aggregates the remaining

CPU time used by the database, as well as the management and scheduling of the SDA workflow.

The most impactful configuration option is the *Edge Distance*, due to an increased sweep line buffer in the CPU-intensive *neighbors* stage. Also, changing the *Required Area* for the filter stage significantly influences the TTR, as this setting restricts the workload processed by all subsequent workflow stages. The *Contraction Distance* configuration affects the complexity of the settlement polygons, analyzed in the final *analysis* stage, while the *Max Neighbors* setting has only a minor impact on time to result, mainly increasing the vertex count, and therefore workload, for the *contraction* stage.

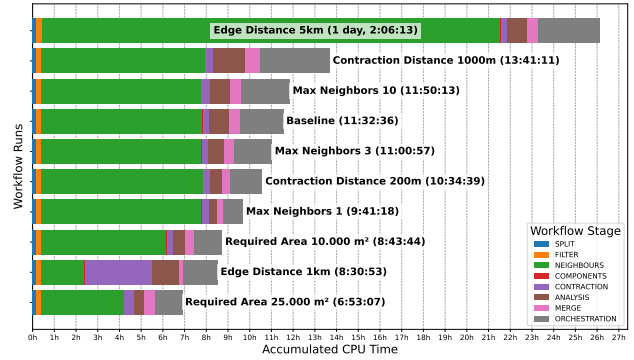


Fig. 8. CPU time per workflow stage of differently-configured workflow runs in Bavaria.

In summary, the configuration options of the workflow notably impact the CPU demand of a run. However, the input region characterized by its total area, settlement distribution, and density has a much more significant impact on the TTR.

F. Resource Utilization

The Slurm report, received after workflow completion, provides resource utilization characteristics of a workflow run. In addition to the TTR and CPU time, the main memory usage and the CPU and memory efficiency are evaluated. Table IV presents the minimum, maximum, and median demand on the aforementioned resources across all workflow runs of both test setups. The median is a more meaningful measure for the computation times since it disregards the large outliers, like, for example, the *Shanghai* run. The median TTR is about three hours, taking eleven hours of processing time. The CPU utilization is fairly poor, with a median value of 9%. On the one hand, the number of CPU cores is reserved according to maximum expected concurrency, that is, during the filtering and the *neighbors* stage. On the other hand, long-running *analysis* jobs prolong the execution time of the workflow while not fully utilizing the allocated computing resources. The memory efficiency is about 20% on average, which is also far from optimal. However, exceeding the allocated memory limit immediately terminates the *Slurm* job. Therefore, the required memory was intentionally overestimated to prevent this scenario. The median workflow run required only about 16

GB of main memory, while the settlement graph of *Shanghai* almost exceeds the memory limit of 1,000 GB that a single node can provide.

TABLE IV
MINIMUM, MAXIMUM, AND MEDIAN VALUE OF SELECTED COMPUTING RESOURCES ACROSS ALL 15 WORKFLOW RUNS WITH VARYING INPUTS AND CONFIGURATIONS.

Resource	Min	Max	Median
Time to Result	0:01:41	2-17:56:59	2:58:50
CPU Time	0:06:43	6-12:48:48	11:00:57
CPU Util. %	3.40%	15.78%	9.06%
Memory	2.99 GB	947.91 GB	16.44 GB
Memory Util. %	1.00%	94.79%	16.44%

In conclusion, the measurements show a significant variance in the time to result, influenced by the input region and the applied configuration. The varying parallelization level results in poor resource utilization due to the required static resource allocation on the HPDA platform.

V. DISCUSSION

Ultimately, the SDA workflow represents a complex EO workflow with varying parallelization, which comes packaged as an easy-to-deploy *docker compose* application. According to best practices within the domain, each workflow stage is modeled as a CWL task. Breaking with the best practices, the workflow orchestration itself could not be abstracted in CWL, as further discussed in Subsection V-A.

The workflow can process even the most densely populated regions on Earth through workload distribution and parallel computation. Throughout the workflow run, the degree of parallelization varies significantly. The first two stages process input files concurrently. Thereafter, the workflow synchronizes to compute the connected components of the settlement graph, representing the unit of parallelization for the subsequent stages.

Together with the parametric dependencies discussed in Subsection V-B, the workflow is a challenging workload for execution in a serverless environment, requiring the dynamic allocation of computing resources according to the current demand. Future endeavors, outlined in Subsection V-C, focus on serverless EO workflow orchestration, which then can be evaluated with the SDA workflow.

A. Shortcomings of CWL

CWL currently cannot express the inter-stage dependencies of the workflow. For example, each *neighbors* job must wait for the *filter* stage of its input and all surrounding files in the grid to complete. However, not all jobs of the *filter* stage have to be finished beforehand, as visualized in Figure 4. CWL does not offer this flexibility and is restricted to a strict processing order, stage by stage.

With the insights into DAG modeling, we aim to adapt and improve the CWL standard for geographic workflows. The

user should be able to model data dependencies flexibly and specify concurrently executed tasks with a dynamic degree of parallelism.

B. Parametric Dependencies

Implicit parametric dependencies, such as the settlement distribution in the region of interest, have the greatest impact on the resource demand profile of the workflow. Since the WSF dataset comes as a grid of compressed raster images (*.tiff*), each covering about the same area in angular units, neither the user nor the platform operator can estimate the resource demand of the workflow solely based on the input's metadata. Moreover, the applied configuration also significantly impacts the procedure's resource demand. Therefore, the SDA workflow is challenging for workflow platforms since the degree of parallelism and the total resource demand cannot be estimated beforehand. More insights about the workload are necessary for transitioning to serverless workflow exploitation systems, which require the dynamic allocation of computing resources. Understanding parametric dependencies is a major challenge in loosening the black box view on a workload, which is apparent in classic resource managers like Slurm. With the measurements in Section IV, we gained valuable insights into the parametric dependencies of a complex EO workflow.

C. Future Work

Improving the flexibility of CWL and providing workload annotations regarding its parametric dependencies and parallelism are the next steps toward a serverless execution of EO workflows. Further, we want to abstract typical EO workflow steps to generalize the measured performance insights and enable the component-based composition of new workflows. This allows geoscientists to develop new workflows by composing self-contained EO components, while the parametric annotations could enable a smart serverless workflow deployment. Through the inherent dynamic resource allocation, we expect improved resource utilization on next-generation serverless EO exploitation platforms.

Ultimately, the SDA workflow is an easy-to-deploy EO workflow implementation suitable for evaluating serverless workflow systems in the EO domain, featuring challenging (implicit) parametric dependencies and varying degrees of parallelism. Our case study revealed limitations in CWL in modeling arbitrary dependency graphs, and we measured the significant impact of parametric dependencies on the time to result. Additionally, the optimized settlement graph generation and component-wise clustering can also be applied to other workflows working on spatial settlement networks, like "Dimensioning Urbanization" [22] or the *OECD Afriapolis* Project [35]. Coherent with the FAIR principles for computational workflows [36], we published the SDA workflow on *WorkflowHub* [37], providing a reproducible, complex EO workflow to the workflows community.

VI. ACKNOWLEDGEMENT

The authors gratefully acknowledge the computational and data resources provided through the joint high-performance data analytics (HPDA) project “terabyte” of the German Aerospace Center (DLR) and the Leibniz Supercomputing Center (LRZ). We especially thank Julian Zeidler for the support during the initial deployment of the workflow on the HPDA cluster. The presented case study was discussed and refined in the SPEC RG Cloud group.

This research project is funded by the Bavarian Research Institute for Digital Transformation (bidt), an institute of the Bavarian Academy of Sciences and Humanities, and the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 522760169 (FOR 5696: SOS).

REFERENCES

- [1] C. Yang, M. Yu, Y. Li, F. Hu, Y. Jiang, Q. Liu, D. Sha, M. Xu, and J. Gu, “Big Earth Data Analytics: A Survey,” vol. 3, no. 2, pp. 83–107, 2019-04-03. doi: 10.1080/20964471.2019.1611175. [Online]. Available: <https://doi.org/10.1080/20964471.2019.1611175>
- [2] T. T. Nguyen, E. M. Hoffmann, and A. Buerkert, “Spatial Patterns of Urbanising Landscapes in the North Indian Punjab show Features predicted by Fractal Theory,” Universität Kassel, 2022. doi: 10.17170/KOBRA-202211097086. [Online]. Available: <https://www.nature.com/articles/s41598-022-05906-4>
- [3] O. J. R. Gustafsson, S. R. Wilkinson, F. Bacall, L. Pireddu, S. Soiland-Reyes, S. Leo, S. Owen, N. Juty, J. M. Fernández, B. Grüning, T. Brown, H. Ménager, S. Capella-Gutierrez, F. Coppens, and C. Goble. (2024-10-09) WorkflowHub: A Registry for Computational Workflows. doi: 10.48550/arXiv.2410.06941. [Online]. Available: <http://arxiv.org/abs/2410.06941>
- [4] OGC, “Best Practice for Earth Observation Application Package,” 2021-12-14. [Online]. Available: <http://www.opengis.net/doc/BP/eoap/1.0>
- [5] M. R. Crusoe, S. Abeln, A. Iosup, P. Amstutz, J. Chilton, N. Tijanić, H. Ménager, S. Soiland-Reyes, B. Gavrilović, C. Goble, and T. C. Community, “Methods Included: Standardizing Computational Reuse and Portability with the Common Workflow Language,” vol. 65, no. 6, pp. 54–63, 2022. doi: 10.1145/3486897. [Online]. Available: <https://cacm.acm.org/research/methods-included/>
- [6] German Aerospace Center, “Terrabyte,” 2024. [Online]. Available: <https://docs.terabyte.lrz.de/about/introduction/>
- [7] STAC, “STAC (SpatioTemporal Asset Catalogs) Specification,” 2025. [Online]. Available: <https://stacspec.org/>
- [8] John Kantner, “Settlement Pattern Analysis,” Oxford University Press, 2004. doi: 10.1093/acref/9780195076189.013.0392. [Online]. Available: <https://www.oxfordreference.com/view/10.1093/acref/9780195076189.001.0001/acref-9780195076189-e-0392>
- [9] M. Marconcini, A. Metz-Marconcini, T. Esch, and N. Gorelick, “Understanding Current Trends in Global Urbanisation - The World Settlement Footprint Suite,” vol. 1, pp. 33–38, 2021. [Online]. Available: <https://pdfs.semanticscholar.org/a7cc/62c7f227c314ec5d7a07d33fe81c54f7fed8.pdf>
- [10] CWL Community, “Cwltool.” [Online]. Available: <https://github.com/common-workflow-language/cwltool>
- [11] J. Vivian, A. A. Rao, F. A. Nothart, C. Ketchum, J. Armstrong, A. Novak, J. Pfeil, J. Narkizian, A. D. Deran, A. Musselman-Brown, H. Schmidt, P. Amstutz, B. Craft, M. Goldman, K. Rosenbloom, M. Cline, B. O’Connor, M. Hanna, C. Birger, W. J. Kent, D. A. Patterson, A. D. Joseph, J. Zhu, S. Zaranek, G. Getz, D. Haussler, and B. Paten, “Toil enables Reproducible, Open Source, Big Biomedical Data Analyses,” vol. 35, no. 4, pp. 314–316, 2017. doi: 10.1038/nbt.3772. [Online]. Available: <https://www.nature.com/articles/nbt.3772>
- [12] Duke Center for Genomic and Computational Biology, “Calrissian,” 2022. [Online]. Available: <https://github.com/Duke-GCB/calrissian>
- [13] T. Coleman, H. Casanova, L. Pottier, M. Kaushik, E. Deelman, and R. Ferreira da Silva, “WfCommons: A framework for enabling scientific workflow research and development,” vol. 128, pp. 16–27, 2022-03-01. doi: 10.1016/j.future.2021.09.043. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167739X21003897>
- [14] T. Coleman, H. Casanova, K. Maheshwari, L. Pottier, S. R. Wilkinson, J. Wozniak, F. Suter, M. Shankar, and R. F. Da Silva, “WfBench: Automated Generation of Scientific Workflow Benchmarks,” in *2022 IEEE/ACM International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*, 2022-11, pp. 100–111. doi: 10.1109/PMBS56514.2022.00014. [Online]. Available: <https://ieeexplore.ieee.org/document/10024036>
- [15] L. Versluis, R. Mathá, S. Talluri, T. Hegeman, R. Prodan, E. Deelman, and A. Iosup, “The Workflow Trace Archive: Open-Access Data From Public and Private Computing Infrastructures,” vol. 31, no. 9, pp. 2170–2184, 2020-09. doi: 10.1109/TPDS.2020.2984821. [Online]. Available: <https://ieeexplore.ieee.org/document/9066946>
- [16] I. Rohwer, M. Schwinger, N. Herbst, P. Friedl, M. Stephan, and S. Konev, “Resource Demand Profiling of Monolithic Workflows,” *ACM*, 2024. doi: 10.1145/3629527.3651425. [Online]. Available: <https://dl.acm.org/doi/10.1145/3629527.3651425>
- [17] F. Lehmann and D. Frantz. (2025) NF-Core Rangeland. [Online]. Available: <https://workflowhub.eu/workflows/1250>
- [18] A. Fouilloux. (2022) Galaxy workflow demonstrating the usage of EODIE Galaxy Tool. doi: 10.48546/WORKFLOWHUB.WORKFLOW.274.1. [Online]. Available: <https://workflowhub.eu/workflows/274?version=1>
- [19] F. Lehmann, D. Frantz, S. Becker, U. Leser, and P. Hostert, “FORCE on Nextflow: Scalable Analysis of Earth Observation Data on Commodity Clusters,” in *CIKM Workshops*, 2021. [Online]. Available: https://www.informatik.hu-berlin.de/de/forschung/gebiete/wbi/research/publications/2021/force_nextflow.pdf/@download/file/force_nextflow.pdf
- [20] DLR, “World Settlement Footprint 2019,” DLR, 2021. [Online]. Available: <https://geoservice.dlr.de/web/maps/eoc:wsf2019>
- [21] W. C. Jochem, D. R. Leasure, O. Pannell, H. R. Chamberlain, P. Jones, and A. J. Tatem, “Classifying Settlement Types from Multi-scale Spatial patterns of Building Footprints,” vol. 48, no. 5, pp. 1161–1179, 2021-06-01. doi: 10.1177/2399808320921208. [Online]. Available: <https://doi.org/10.1177/2399808320921208>
- [22] T. Esch, M. Marconcini, D. Marmanis, J. Zeidler, S. Elsayed, A. Metz, A. Müller, and S. Dech, “Dimensioning Urbanization – An Advanced Procedure for Characterizing Human Settlement properties and patterns using spatial network analysis,” vol. 55, pp. 212–228, 2014. doi: 10.1016/j.apgeog.2014.09.009. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0143622814002100>
- [23] European Commission, Eurostat (ESTAT), GISCO, “Urban Audit 2021 - Area Management - Dataset: EuroGeographics for the Administrative Boundaries,” European Commission, Eurostat (ESTAT), GISCO, 2022. [Online]. Available: <https://ec.europa.eu/eurostat/web/gisco/geodata/statistical-units/urban-audit>
- [24] H. Taubenböck, M. Weigand, T. Esch, J. Staab, M. Wurm, J. Mast, and S. Dech, “A new ranking of the world’s largest cities—Do administrative units obscure morphological realities?” vol. 232, 2019. doi: 10.1016/j.rse.2019.111353. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0034425719303724>
- [25] p. u. family=Bellefon, given=Marie-Pierre, P.-P. Combes, G. Duranton, L. Gobillon, and C. Gorin, “Delineating Urban Areas using Building Density,” vol. 125, p. 103226, 2021. doi: 10.1016/j.jue.2019.103226. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0094119019301032>
- [26] X. Huang, Q. Li, H. Liu, and J. Li, “Assessing and Improving the Accuracy of GlobeLand30 Data for Urban Area Delineation by Combining Multisource Remote Sensing Data,” vol. 13, no. 12, pp. 1860–1864, 2016. doi: 10.1109/LGRS.2016.2615318. [Online]. Available: <https://ieeexplore.ieee.org/document/7604100>
- [27] Lorenz Gruber. (2024) FISHNET: Framework for Identification of Settlement pattern Heuristics using Networks. [Online]. Available: <https://gitlab2.informatik.uni-wuerzburg.de/descartes/sos/fishnet>
- [28] GDAL/OGR contributors, “GDAL/OGR Geospatial Data Abstraction software Library,” Open Source Geospatial Foundation, 2022. doi: 10.5281/zenodo.5884351. [Online]. Available: <https://gdal.org>
- [29] M. Berg, O. Cheong, M. Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*, third edition ed. Springer-Verlag Berlin Heidelberg, 2008. [Online]. Available: https://cimec.org.ar/foswiki/pub/Main/Cimec/GeometriaComputacional/DeBerg_-_Computational_Geometry_-_Algorithms_and_Applications_2e.pdf
- [30] E. W. Dijkstra, “Cooperating Sequential Processes,” in *The Origin*

- of Concurrent Programming: From Semaphores to Remote Procedure Calls, Per Brinch Hansen, Ed. Springer-Verlag, 2002, pp. 65–138.
- [31] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, fourth edition ed. The MIT Press, 2022. [Online]. Available: <https://dl.ebooksworld.ir/books/Introduction.to.Algorithms.4th.Leiserson.Stein.Rivest.Cormen.MIT.Press.9780262046305.EBooksWorld.ir.pdf>
 - [32] Memgraph, “Memgraph,” 2024. [Online]. Available: <https://memgraph.com/docs>
 - [33] R. Friedhorsky and T. Randles, “Charliecloud: Unprivileged Containers for user-defined Software Stacks in HPC,” in *SC17: International Conference for High Performance Computing, Networking, Storage and Analysis*, B. Mohr and P. Raghavan, Eds. IEEE, 2017, pp. 1–10. doi: 10.1145/3126908.3126925. [Online]. Available: <https://ieeexplore.ieee.org/document/9926260>
 - [34] A. B. Yoo, M. A. Jette, and M. Grondona, “SLURM: Simple Linux Utility for Resource Management,” in *Job Scheduling Strategies for Parallel Processing*, ser. Lecture Notes in Computer Science, D. G. Feitelson and D. Feitelson, Eds. Springer, 2003, vol. 2862, pp. 44–60. doi: 10.1007/10968987. [Online]. Available: https://link.springer.com/chapter/10.1007/10968987_3
 - [35] P. Heinrichs, “Africapolis: Understanding the Dynamics of Urbanization in Africa,” no. Special Issue 22, pp. 18–23, 2020-12-23. [Online]. Available: <https://journals.openedition.org/factsreports/6233?lang=fr>
 - [36] S. R. Wilkinson, M. Alogalaa, K. Belhajjame, M. R. Crusoe, B. d. P. Kinoshita, L. Gadelha, D. Garijo, O. J. R. Gustafsson, N. Juty, S. Kanwal, F. Z. Khan, J. Köster, K. Peters-von Gehlen, L. Pouchard, R. K. Rannow, S. Soiland-Reyes, N. Soranzo, S. Sufi, Z. Sun, B. Vilne, M. A. Wouters, D. Yuen, and C. Goble, “Applying the FAIR Principles to Computational Workflows,” vol. 12, no. 1, p. 328, 2025-02-24. doi: 10.1038/s41597-025-04451-9. [Online]. Available: <https://www.nature.com/articles/s41597-025-04451-9>
 - [37] L. Gruber. (2025) Settlement Delineation and Analysis. doi: 10.48546/WORKFLOWHUB.WORKFLOW.1308.2. [Online]. Available: <https://workflowhub.eu/workflows/1308?version=2>