# Limit Crossover Algorithm

## Introduction

In support of the objective to update the look and feel, and functionality of the Remote Monitoring portal, a new web portal was developed in collaboration with the external consulting company, Softdel. A requirement of this new web poral was the easy integration of more advanced condition assessment algorithms, which would be developed and deployed to the portal by Gastops. In this way, the logic used for the assessment of equipment conditions could also be kept confidential.

As these algorithms may require different dependencies, it was decided that they would be packaged as Docker images, which could then be defined and deployed via an analytics engine in the portal. This analytics engine had several requirements, including:

* The ability to easily add new algorithms to the portal.
* Ensure that the latest version of the algorithms is deployed during condition assessment to allow for later improvements to the algorithms.
* Prepare and provide the required data to the algorithms.
* Retrieve and the store the output conditions from the algorithms against the applicable equipment.

To facilitate the development of this analytics engine, the limit-crossover algorithm defined in this document, was developed, and supplied to Softdel for inclusion in the first release of the portal.

## Requirements

### Condition Assessment

The following condition assessment requirements were considered during the design of the limit crossover algorithm:

* The algorithm shall return a list of the identified condition changes, sorted according to timestamps.
* Each condition in the list shall indicate a change in the associated equipment’s condition.
* Conditions shall be determined from both the upper and lower bounds provided in the limits.
* Each condition in the returned list shall represent the worse condition at that timestamp for that equipment item across all supplied data fields and limits.

### Deployment

The following deployment requirements were considered during the design of the limit crossover algorithm:

* The algorithm shall be packaged with its required dependencies as a Docker image to simplify deployment within different environments.
* Define a file structure for required input data, which will be used by the analytics engine to prepare the data to be supplied to the algorithm.
* Include support for both local and S3 storage filesystems to allow for deployment in AWS, and testing in local environments.
* Provide the ability to include other metadata in the output conditions, which may be used by the analytics engine to associate the conditions with the correct equipment, and model definition.

## Algorithm Definition

The limit crossover algorithm is designed to identify the worst condition for each equipment item and at each timestamp in the supplied data (Appendix A). This is achieved by determining the condition at each timestamp and equipment item, which is then appended to the list of final conditions if it passes the following checks:

* It is the worst condition for that equipment item at that time out of all applicable data fields.
* It represents a change in the condition of the equipment at that timestamp, when compared to the previous worst condition.

It is possible for conditions at later timestamps to exist after newly inserted conditions. Therefore, following the insertion of a new condition, the algorithm will loop through these later conditions and remove them if the condition applies to a different data field compared to the newly added condition and the condition is not as severe as the new condition. If the condition is removed, then the algorithm performs the same check on the next condition in the list if it exists until it reaches the end of the list or identifies a condition that does not pass the check. The algorithm will add a supplied parameter to each condition in the final condition list, which may be used for the association of the conditions with other records, such as parent or component equipment, or the organization to which the equipment is assigned.

To allow for the deployment of the algorithm to a cloud server as well as testing in both local and cloud environments, the algorithm includes support for the following:

* Retrieval of the required data files from either local or AWS S3 storage.
* Return the final conditions either as a JSON formatted file in the same storage location as the input files, or via a supplied REST API endpoint and credentials.

## Test Cases

The following test cases were used during the evaluation of the algorithm:

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Steps** | **Passed** |
| 1 | The algorithm produces the same result when run natively or as a Docker image | 1. Execute the algorithm with and Without Docker containerization with the same input files and arguments. 2. Confirm that the output conditions from both execution methods are the same | Yes |
| 2 | The algorithm can successfully load the input files from local and AWS S3 storage | 1. Store the same file locally and on AWS S3 2. Execute the algorithm with the different storage locations 3. Confirm the output conditions are the same | Yes |
| 3 | The algorithm can return the final conditions as a JSON file and via a REST API endpoint | 1. Execute the algorithm with the same inputs and arguments with the different return arguments 2. Confirm that the conditions are returned successfully via both methods | Yes |
| 4 | The algorithm successfully appends new conditions if they are different from the previous condition for a given data field | 1. Execute the algorithm with input data that cycles between condition levels 2. Confirm that the returned conditions and timestamps correct | Yes |
| 5 | The algorithm successfully appends new conditions for different data fields compared to the previous condition if it is worse | 1. Execute the algorithm with data containing more than one data field 2. Confirm that the return conditions are the worst for at each timestamp out of all data fields | Yes |
| 6 | The algorithm removes later conditions after inserting a new condition if it they are for different data fields and are less severe compared to the new condition | 1. Execute the algorithm with data containing different data fields in which one of the data fields has a known condition worse than the other at a timestamp before the other field 2. Repeat the execution with the order of the fields swapped 3. Confirm that the return conditions are the same and correct from both executions | Yes |
| 7 | The algorithm correctly differentiates the conditions from different equipment items in the data | 1. Execute the algorithms with data and limits for different equipment items that overlap in data field names and timestamps 2. Confirm the return conditions is the worst for the relevant equipment item at that timestamp 3. Ensure that no condition changes are missed in the returned results | Yes |

**Appendix A**: Block flow diagram of the limit crossover algorithm

