Catchment Works Designer
User Documentation

Version: 3.0
Author : Francis Pantus © 2012-2026
Latest edits: 03/08/2026

CWD
Catchment Works Designer
User Manual

Manual for CWD web UI.

Part of this user manual is AI-generated

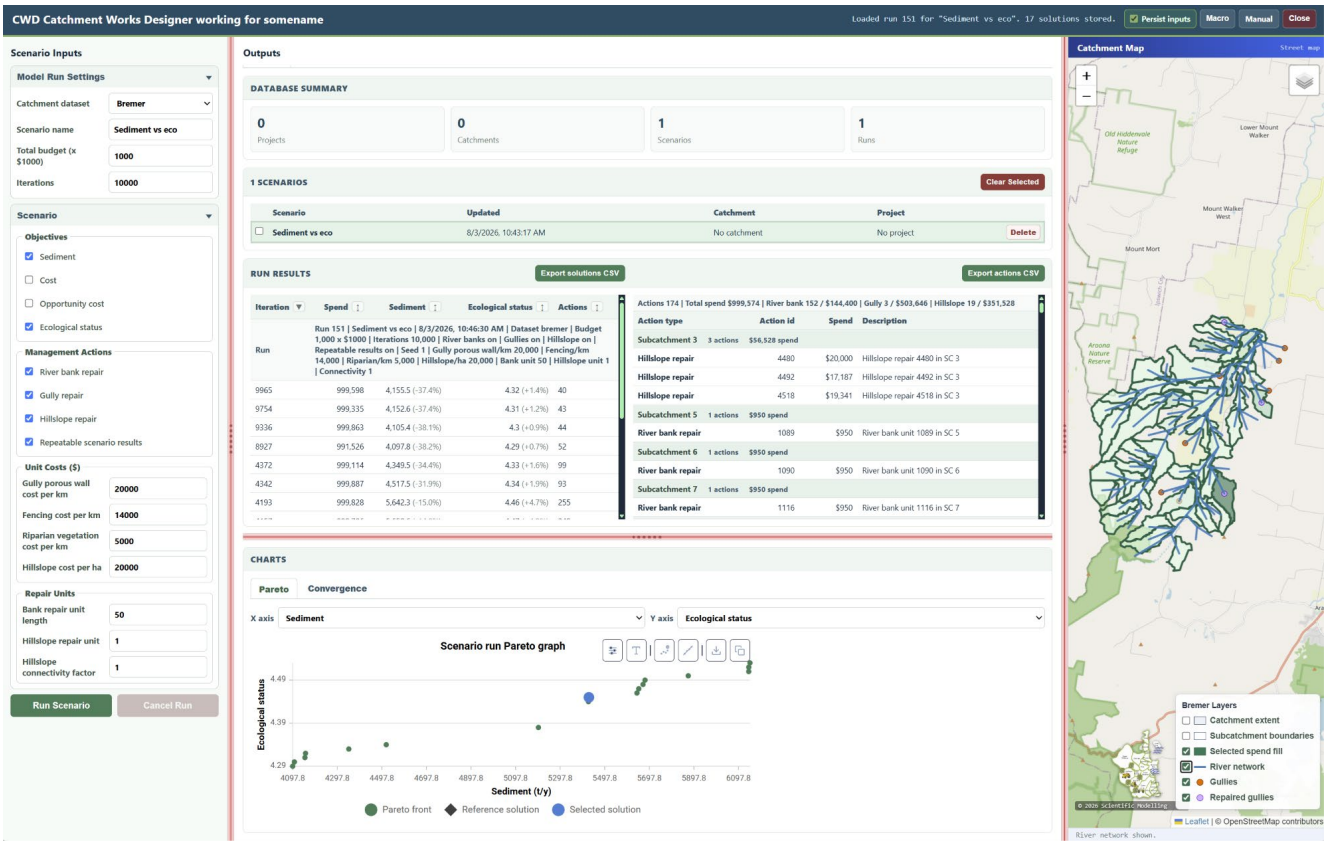


Table of Contents

Abstract....6
1. Introduction....7
1.1 Background....7
1.2 7
1.3 Key capabilities....7
1.4 Scope and current limitations....8
2. Interface Overview....9
2.1 Header controls....9
2.2 Left input pane....9
2.3 Middle output pane....9
2.4 Right map pane....9
3. Workflow....10
3.1 Step 1 - Select a catchment dataset....10
3.2 Step 2 - Define the scenario....10
3.3 Step 3 - Choose objectives and actions....10
3.4 Step 4 - Run the scenario....10
3.5 Step 5 - Inspect outputs....10
3.6 Step 6 - Compare and manage history....10
4. Results Panel....11
4.1 Database Summary....11
4.2 Scenarios list....11
4.3 Run result and actions tables....11
4.3.1 A simple way to inspect a run....11
4.3.2 Actions table....11
4.4 Spend and Cost....12
4.5 Run Log....12
4.6 Data tab....12
5. Charts....13
5.1 Pareto chart....13
5.1.1 Axis selection....13
5.2 Convergence chart....13
5.3 Chart toolbar....13
6. Map....14
6.1 Street map base....14
6.2 Overlay layers....14
6.3 Legend....14
6.4 Map persistence....14
7. Data and Storage....15

7.1 Application folders 15
7.2 Dataset files 15
7.3 New catchment import 15
7.4 Database storage 15
8. Technical Appendices 16
8.1 Appendix A - Input Field Reference 16
8.2 Appendix B - API Reference 17
8.3 Appendix C - Scenario Runner Algorithm 17
8.4 Appendix D - Objective Functions 18
8.4.1 Sediment 18
8.4.2 Cost and Spend 18
8.4.3 Opportunity Cost 18
8.4.4 Ecological Status 18
8.5 Appendix E - Sediment and Management-Action Calculations 18
8.5.1 Hillslope sediment 18
8.5.2 Riparian filter 19
8.5.3 Riverbank sediment 19
8.5.4 Gully sediment 19
8.5.5 Action costs 19
8.6 Appendix F - Map and Chart Implementation Notes 19
8.7 Appendix G - Database and Persistence Model 19
8.8 Appendix H - Validation Context 20
8.8.1 Recommended validation procedure 20
8.9 Appendix I - Known Technical Issues and Interpretation Rules 20
8.10 Appendix J - Maintenance Notes for Developers 20
8.11 Appendix K - Literature and Standards Context 20
8.12 Appendix L - Pareto Graph Technical Notes 21
8.12.1 Interpretation rules for CWD users: 21
9. Technical Appendix: VB.NET vs CWD Objective Comparison Results 23
9.1 1-objective comparison results 23
9.2 2-objective comparison results 23
9.3 3-objective comparison results 23
9.4 Interpretation 23
10. Technical Appendix: Creating Partial USLE Inputs for New Catchments 25
10.1 Purpose 25
10.2 Recommended import workflow 25
10.3 Support for source rasters 25
10.4 Safeguards 25
10.5 Boundary of responsibility 25

11. Technical Appendix: Creating CWD ASCII Grids in QGIS 26
11.1 Purpose 26
11.2 General export workflow 26
11.3 Creating derived factor rasters 26
11.4 Creating the river network and subcatchment linkage 26
11.5 Required checks before import 27
11.6 Recommended file names 27
References 28

Abstract

The catchment resource management process often involves some level of negotiation between the various stakeholders such as industry, agricultural sectors, environmental groups, government and others. In many cases, natural resource management must consider multiple, often competing, objectives.
The often-used methodological approach to support the management process is to somehow optimise the theoretical results of management actions. This invariably involves some scaling between the objectives to optimise a single, compound object. The goal of such approach is to deliver one or more ready-to-go 'best' management actions packages.
The approach taken in the Catchment Works Designer is to support such process by creating trade-offs instead. It aims to support stakeholders to consider the effects of improving one objective on the other objectives. Weighting the importance of objectives is a management responsibility, not a science support one. The approach taken here aims to support the negotiation process by creating and visualising such trade-offs.
The CWD Catchment Works Designer is a web-based catchment-management application for exploring budget-constrained riverbank, gully and hillslope repair scenarios with a range of management actions.
The core of the trade-off generation is a Pareto approach which evaluate the effectiveness of management action packages with respect to the stated objectives. This manual describes the user workflow, the interpretation of output tables, Pareto and convergence charts, map layers, persisted settings, and the model assumptions that affect results.
The application ports the core model concepts from a C#-VB.NET catchment-planning application developed in 2011 by the eWater CRC, in collaboration with the Australian Rivers Institute into a web-based implementation, allowing a universal access for all parties to common data, methods and results. Technical appendices document the input fields, API actions, storage model, model equations, scenario algorithm, map/chart implementation and validation context, and maintenance issues.

Keywords: *catchment, resource management, trade-offs; sediment; gully repair; riverbank repair; hillslope repair; Pareto plot.*

1. Introduction

1.1 Background

This current web-based CDW is a ported version of a project that delivered in 2011. That project was done as part of the eWater Cooperative Research Centre activities in SE Queensland. eWater had a directive to focus significant research resources into the SEQ region to support existing water quality and aquatic ecosystem health initiatives. This directive was a component of the eWater research program to coordinate activities into four major research projects throughout Australia. Integration of the eWater SEQ research program with the Healthy Country project was an excellent opportunity to build on a significant State Government funded initiative, which addresses the erosion processes occurring in the upper catchments of SEQ that are responsible for the decline in both water quality and aquatic ecosystem health. It also dovetailed well with the many activities of the Griffith University Australian Rivers Institute expertise and programs. The project description, implementation and some example results of that project can be found here: [13]

1.2

CWD is intended to support the catchment management process to test how alternative management settings affect selected objectives by assessing, and Pareto-filtering many alternative activity packages.
Scenario runs include an option to use reproducible seeded sampling of management-package composition, so repeated runs with the same data and settings can be compared during validation between the various software versions. Selected objective checkboxes define which objective values are stored, plotted and used for Pareto dominance and trade-off reporting.

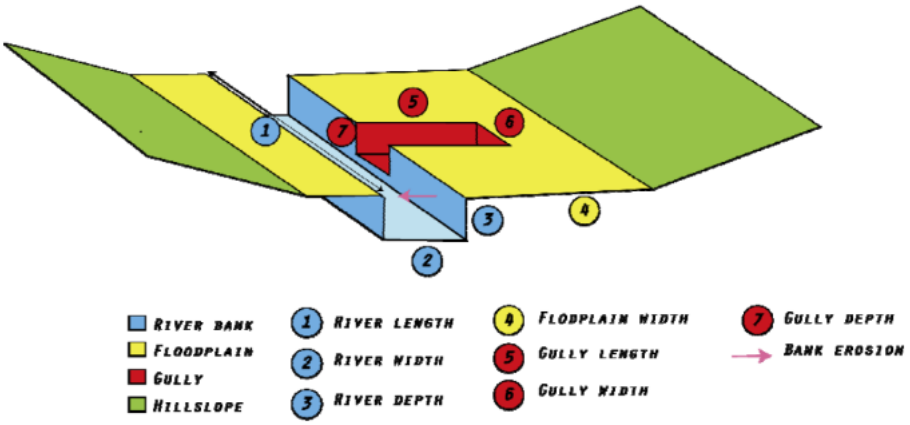


Figure 1 This diagram shows the main elements implemented in the sediment transport model of the CWD application.

1.3 Key capabilities

Currently, in this proof-of-concepts software version, the following activities are supported:

- Load Bremer or Knapps Creek catchment datasets.
- Run riverbank, gully and hillslope repair scenarios under a total budget.
- Persist input values, collapsed-card state and map centre/zoom in the browser.
- Store scenarios, runs, solution rows and run logs in the CWD MariaDB database.
- Display newest scenarios and newest solution rows closest to the top of the output panels.
- Plot a Pareto-style objective chart with selectable X and Y objectives.
- Plot a convergence chart on a separate chart tab.
- Show catchment extent, sub-catchment s, river network, gullies, selected spend fills and repaired gullies on a street map with legend layer toggles.
- Open the CWD manual through the CWD library page when the web library route is available.

1.4 Scope and current limitations

- The scenario runner uses seeded package sampling and Pareto archive updates. It does not use simulated annealing.
- The Iterations input controls the candidate-package loop and is stored with the run for traceability. Annealing is not used.
- Opportunity cost changes only when hillslope repair actions are included in selected solutions.
- Ecological status is tied to forest-cover state in the ported model. If a run only selects gully or riverbank repairs, this objective can remain unchanged.
- No field should silently fall back. Required fields are validated server-side and missing settings are reported as errors.

2. Interface Overview

The CWD page follows the EHMP Random Forest layout pattern: a left input pane, a middle output-and-chart pane, and a right map pane separated by standard splitters. Standard scrollbars are used in scrollable areas so long result lists do not push other panels out of view.

2.1 Header controls

1. Persist inputs saves and restores form inputs and the current map view in the browser.
2. Macro runs the selected catchment smoke-test macro.
3. Manual opens the CWD manual route through the CWD library page.
4. Close saves persisted state and then attempts to close the browser window.

2.2 Left input pane

The left pane contains collapsible cards. Labels and controls are arranged on one line where space permits, to keep the input side compact. The Scenario card contains the dataset, scenario name, budget, iterations and objective checkboxes. The Model Run Settings card contains management-action switches, unit costs and repair-unit settings.

2.3 Middle output pane

The upper middle section uses tabs for Output, Run Log and Data. The Output tab contains the database summary, scenario list and run-results table. The lower middle section has its own chart tabs for Pareto and Convergence. This separation keeps tabular output and chart output from competing for the same vertical space.

2.4 Right map pane

The map pane loads a street-map base layer and then overlays catchment-specific features. The map shows the catchment extent, sub-catchment s, river network, gully markers, selected-spend sub-catchment fills and repaired gully markers. The map legend can be used to show or hide individual overlay layers. The map centre and zoom are saved when persistence is enabled.

Figure 1: CWD uses a three-paneuser interaction layout

3. Workflow

3.1 Step 1 - Select a catchment dataset

Choose Bremer or Knapps Creek from the Catchment dataset dropdown. The selected dataset is used for both map data and model import. When the dataset changes, CWD reloads catchment extent, sub-catchment s, river network and gullies.

3.2 Step 2 - Define the scenario

Enter a meaningful scenario name, total budget and iteration count. The scenario name is stored with the run. The budget constrains which actions can be selected. Iterations is stored for traceability and compatibility with the older model workflow.

3.3 Step 3 - Choose objectives and actions

Select the objectives to display and store. Sediment is always used by the current runner. Cost, opportunity cost and ecological status can be added to output tables and charts. Select which action families are allowed: riverbank repair, gully repair and hillslope repair.

3.4 Step 4 - Run the scenario

Press Run Scenario. The cursor changes to busy while the request is running, and focus moves to the Run Log tab. The run log is updated as the database row is created, catchment data is imported, the model is initialised, candidate packages are evaluated, solution rows are created and the run is completed. When the run finishes, focus returns to the Output tab.

3.5 Step 5 - Inspect outputs

After a scenario run finishes, start with the Output tab. Read the run summary first, then scan the Run Results table to compare solution packages. Select a solution row or Pareto chart point to make it current; the map highlight and Actions table then show what that selected solution would do on the ground.

3.6 Step 6 - Compare and manage history

The Scenarios list is timestamped and ordered newest first. It has a fixed height and vertical scrollbar so that scenario history and run results do not cover each other. Use Clear when the history is no longer needed.

4. Results Panel

4.1 Database Summary

The database summary reports counts for projects, catchments, scenarios and runs. It confirms that the web application can reach the MariaDB CWD database through the PHP database module.

4.2 Scenarios list

The scenario list displays stored scenario names with update timestamps. Newest entries appear closest to the top. The Clear button removes scenario history, run rows, solution rows and run logs through the server-side clear_scenarios action.

4.3 Run result and actions tables

The Output tab contains two linked tables. The Run Results table compares solution packages created by the current run. A scenario is the set of inputs you ran; a solution package is one possible selection of repair actions that fits those inputs. The first row summarises the run number, scenario name, timestamp, dataset, budget, iterations, selected action families and numerical settings. The solution rows then show each retained package by iteration, spend, selected objective values, action count and status. Select a row, or select a point in the Pareto chart, to make that package current. Column headings can be used to sort the rows when you want to look for lower spend, fewer actions or a particular objective value.
The Status column is not a pass/fail judgement. *Reference* marks a useful starting package for inspection: the displayed Pareto solution closest to the observed ideal point using a minimum distance rule. Treat it as a navigation marker, not as the 'best' solution. Selected marks the package currently driving the Actions table and map highlight.

4.3.1 A simple way to inspect a run

For a first pass, use this sequence. First, check the run summary row to confirm that the dataset, budget, objectives and action families are the ones you intended to run. Second, look at the Reference solution as a starting point, but do not stop there. Third, sort or scan the Run Results rows for packages with lower spend, fewer actions or a stronger result for the objective you care about. Fourth, select two or three promising rows or Pareto chart points and compare their Actions tables. Finally, use the map to check whether the proposed work is concentrated in sensible subcatchments or spread across the catchment in a way that needs further review.
When comparing packages, ask practical questions: Is the extra spend buying enough improvement? Does a package rely on many small actions or a smaller number of larger actions? Are the actions mostly river bank, gully or hillslope works? Are they clustered in one subcatchment or distributed across several? These questions help turn the tables and chart into a management conversation rather than a list of numbers.

4.3.2 Actions table

The Actions table answers a practical question: if this solution package was chosen, what work would be included, where would it occur and how much of the estimated spend

would it use? The table refreshes whenever you select a Run Results row or a Pareto chart point.
The summary line gives the total number of actions, total spend, and the count and spend for river bank, gully and hillslope actions. The rows underneath are grouped by subcatchment so that related work is easier to inspect spatially. Each subcatchment group shows its action count and spend, followed by the individual actions in that subcatchment. Each action row lists the action type, action id, spend and description. River bank actions refer to bank repair units, gully actions refer to gully repair work, and hillslope actions refer to hillslope repair units. Use this table with the map: select a solution, note the subcatchments and action types, then check the highlighted locations to understand the pattern of proposed work. Use Export actions CSV to download the same selected-solution action list, including subcatchment totals.

4.4 Spend and Cost

Spend is the estimated money used by a solution package. It is always displayed because users usually need to compare environmental change against budget use. Cost is an optional objective; when the Cost objective is selected, its value currently equals total spending in the ported model. In beginner terms, use Spend to ask how much of the budget the package uses, then use the selected objectives to ask what that spend appears to achieve.

4.5 Run Log

The Run Log tab records the major server-side stages of the run. It is especially useful when checking that no silent fallback occurred. Missing settings, invalid numbers or invalid booleans are expected to fail explicitly.

4.6 Data tab

The Data tab reports catchment map data and model import summaries. It is intended for validation and troubleshooting during the continuing web port.

5. Charts

5.1 Pareto chart

The Pareto chart is a visual way to compare solution packages from the selected run. Each point is one solution package, not one individual action. The X axis and Y axis dropdowns choose which two measures are shown. Selecting a point makes that package current, updates the Run Results selection, refreshes the Actions table and updates the map highlight.
Use the chart to look for trade-offs rather than a single automatic answer. For example, a package with lower spend may also have less sediment reduction, while another package may use more actions to improve one objective. Start with the Reference point, then inspect nearby alternatives that are cheaper, simpler or better on the objective that matters most for the decision.

5.1.1 Axis selection

Use the X axis and Y axis dropdowns to compare two available measures. Typical comparisons are Spend versus Sediment, Spend versus Opportunity cost, or Sediment versus Ecological status. When an objective was not selected for the run, it is not available as a chart axis. After changing axes, re-check which direction represents the more favourable result for the question you are asking; the chart is a comparison aid, not a management decision by itself.

5.2 Convergence chart

The Convergence tab can show newly selected trade-off solutions per iteration block or standardised objective traces on a common scale. The newly selected solutions view is intended to indicate whether additional iterations are still producing new trade-off information.

5.3 Chart toolbar

The chart area uses the EHMP chart helper pattern and toolbar controls copied from the Waterscope chart modules. Toolbar controls support chart interaction such as chart type, colour, copy, print and zoom reset where supported by the Syncfusion chart instance.

6. Map

6.1 Street map base

The map initialises a standard Leaflet map and loads a street-map base layer using the Waterscope Maps helper where available. The fallback is an OpenStreetMap tile layer. Leaflet is used for client-side web mapping and OpenStreetMap provides the base map tiles [1], [10].

6.2 Overlay layers

- Catchment extent: the outer catchment boundary.
- Sub-catchment s: raster-derived sub-catchment polygons, shown with transparent fill and boundary outlines.
- Selected spend fill: green sub-catchment overlays showing relative budget spend in the selected solution.
- River network: downstream links between sub-catchment s.
- Gullies: all mapped gullies in the selected dataset.
- Repaired gullies: gullies that are included in the selected solution's action list.

6.3 Legend

The legend colours match the rendered map markers and overlays. Each legend row has a checkbox that controls layer visibility for catchment extent, sub-catchment s, selected spend fill, river network, gullies and repaired gullies. Standard gullies are orange, repaired gullies are purple, and selected spend fill is green with stronger colour for higher spend in that sub-catchment .

6.4 Map persistence

When Persist inputs is checked, CWD stores centre and zoom by dataset. The persisted view is restored after the dataset layers have loaded so that automatic fitting does not override the user's saved map view.

7. Data and Storage

7.1 Application folders

The CWD source uses the web-port structure agreed for the project: HTML in the root, JavaScript in js, PHP in php, CSS in css, input data in data, and manuals in docs/Manuals. The live WAMP copy uses the corresponding C:\wamp64\www\CWD folder.

7.2 Dataset files

The data directory contains ASCII rasters, raster sidecar files and catchment CSV files. Bremer includes Bremer_Gullies.csv and Bremer_Sub-catchment Data.csv. Knapps Creek includes Gullies_Knapps.csv and Sub-catchment Data_Knapps.csv. Raster layers include sub-catchment s, partial USLE, ground cover, distance to riparian buffer, opportunity cost and related sidecars.

7.3 New catchment import

Use the Data tab New Catchment panel to scan raw catchment folders under data, map the required CSV and ASCII raster files, validate the dataset, and create the catchment manifest. If a prepared Partial_USLE.asc is already available, select it directly. If not, use Create Partial USLE to generate it from R, K, L, and S rasters or constants, then validate before creating the manifest. Detailed QGIS export and Partial USLE preparation guidance is provided in the technical appendices.

7.4 Database storage

CWD stores scenario and run state in the MariaDB database named CWD. The database module is CwoDbUtils.php. It follows the Waterscope database-utility approach by centralising connection, query, insert, scenario, run, solution and run-log operations in one PHP module.

8. Technical Appendices

8.1 Appendix A - Input Field Reference

Input	Type/unit	Purpose
Catchment dataset	Bremer or Knapps Creek	Selects the input data bundle used for map loading and model import.
Scenario name	Text	Stored with the run so scenarios can be identified later.
Total budget	Dollars	Maximum spend allowed by the scenario runner.
Iterations	Integer	Kept for compatibility with the VB.NET scenario concept; the current deterministic runner generates feasible solution points rather than random iterations.
Sediment	Objective	Minimises total sediment at the terminating sub-catchment s.
Cost	Objective	Reports total spend as an optimisation objective when selected.
Opportunity cost	Objective	Reports opportunity cost from repaired hillslope cells.
Ecological status	Objective	Reports ecological status derived from forest-cover proportions; shown as an objective value for charting.
River bank repair	Action switch	Allows riverbank repair units to be generated.
Gully repair	Action switch	Allows gully repair actions to be generated.
Hillslope repair	Action switch	Allows hillslope repair units to be generated.
Gully porous wall cost per km	Dollars/km	Cost component used for gully repair.
Fencing cost per km	Dollars/km	Cost component used for riverbank and gully repair.
Riparian vegetation cost per km	Dollars/km	Cost component used for riverbank and gully repair.
Hillslope cost per ha	Dollars/ha	Cost basis for repaired hillslope units.
Bank repair unit length	Metres	Length represented by one riverbank repair unit.

Hillslope repair unit	Hectares	Target area represented by one hillslope repair unit.
Hillslope connectivity factor	Multiplier	Applies a connectivity penalty to hillslope-action cost when not equal to one.

8.2 Appendix B - API Reference

Action	Method	Purpose
health	GET	Checks database connection and schema version rows.
dashboard	GET	Returns project, catchment, scenario and run counts.
scenarios	GET	Returns the newest scenarios first.
clear_scenarios	POST	Deletes scenario history, runs, solutions and run logs.
run_scenario	POST	Runs the seeded scenario runner and stores the outputs.
latest_run	GET	Returns the newest completed run, solutions and logs.
model_smoke	GET	Executes a basic model smoke test.
model_network_smoke	GET	Executes a network/topology smoke test.
cwo_datasets	GET	Lists catchment dataset options.
bremer_map	GET	Returns GeoJSON-style map data for the selected dataset.
bremer_model_smoke	GET	Imports model data and returns import metrics.

8.3 Appendix C - Scenario Runner Algorithm

The current algorithm is a seeded random-series package sampler intended to be auditable during port validation. It is not a simulated-annealing Designer. The objective of the Pareto approach is to expose trade-offs without forcing unlike objectives into a single weighted score [3].

5. Validate all required input settings; reject missing or invalid values.
6. Create a scenario row and a running run row in MariaDB.
7. Import the selected catchment dataset.
8. Initialise cls_CatchmentMgtModel with cls_Scenario.
9. Prepare allowed management actions.
10. Evaluate the baseline objective values.
11. Generate candidate management packages under the budget using the reproducible random seed.
12. Evaluate the selected objective values for each candidate package.

13. Archive newly accepted trade-off solutions when they are non-dominated or dominate existing archive members.
14. Store solution rows with objective values, compacted action details, run logs and convergence history.
15. Mark Pareto members and the closest-to-ideal Pareto solution, using the C#-style normalised distance rule, then complete the run.

$$\text{score}(a) = (J_s(\text{current}) - J_s(\text{repair } a)) / C(a)$$
$$\text{sum}(C(a_i)) <= B$$

8.4 Appendix D - Objective Functions

The ported PHP model follows the VB.NET naming and objective identifiers. Sediment is ObjectiveId::SedimentObjId, ecological status is ObjectiveId::EcologicalStatusObjId, opportunity cost is ObjectiveId::OpportunityCostObjId, and cost is ObjectiveId::Cost.

8.4.1 Sediment

Sediment is calculated at terminating sub-catchment s. Each terminating sub-catchment includes its local sediment and upstream sediment. This follows the network accumulation concept used in catchment sediment models [14], [19]. Evidence from fallout radionuclide tracing in South-East Queensland water-supply catchments also identifies channel erosion as the dominant sediment source, supporting the emphasis on channel-network rehabilitation [9].

$$J_s = \text{sum}(S_t) \text{ for all terminating sub} - \text{catchment s t}$$
$$S_i = H_i + R_i + G_i + \text{sum}(S_u)$$

8.4.2 Cost and Spend

Spend is model total spending across repaired management actions. When the Cost objective is selected, the cost objective value is currently the same total spending value.

$$\text{Spend} = \text{sum}(C(a_i)) \text{ for repaired actions } a_i$$

8.4.3 Opportunity Cost

Opportunity cost is accumulated only for repaired hillslope cells, using the opportunity-cost raster in the selected catchment dataset.

$$J_oc = \text{sum}(OC_xy) \text{ for repaired hillslope cells } (x, y)$$

8.4.4 Ecological Status

Ecological status is derived from forest-cover proportions in sub-catchment s that are flagged for ecological objective use. The ported model stores the value with a negative sign because the optimisation framework represents maximising objectives in a minimisation-oriented result set.

$$J_e = -\text{sum}(E_i)$$

8.5 Appendix E - Sediment and Management-Action Calculations

8.5.1 Hillslope sediment

The hillslope calculation uses a partial-USLE raster, ground-cover raster, distance-to-riparian buffer raster and cell size. It is related to the Universal Soil Loss Equation family of erosion models [18], but the coefficients shown here are the implemented CWD/VB.NET equations, not a generic RUSLE equation.

$$b = -0.001 \exp(0.053\ G)$$
$$USLE = P * 0.1336 \exp(bD) * 0.5665 \exp(-0.0487G) * cellSize\text{^}2 / 10000$$

8.5.2 Riparian filter

$$p_r = (L_existing_riparian + L_repaired_bank) / L_channel$$
$$F_r = 1 \text{ if } p_r < 0.25;\ F_r = 0.25 \text{ if } p_r > 0.75;\ \text{otherwise } F_r = 1 - p_r$$

8.5.3 Riverbank sediment

$$P_bank = 0.0005 * g * rho_w * Q_bankfull$$
$$R_partial = P_bank * slope * (1 - \exp(-0.015\ W_floodplain)) * depth * rho_s * prop_suspended * L_channel$$
$$R = R_partial * (1 - 0.95\ p_r)$$

8.5.4 Gully sediment

$$G = V_g * 0.6 * rho_s * 0.5 / T_e$$
$$G_repaired = 0.2G$$

8.5.5 Action costs

$$C_bank_unit = ((C_riparian_km + C_fencing_km) / 1000) * L_bank_unit$$
$$C_gully = (C_porous_wall_km / 1000) * L_gully + (L_gully / L_bank_unit) * C_bank_unit$$
$$C_hillslope_cell = (C_hillslope_ha * A_unit_ha) / cells_per_unit$$

8.6 Appendix F - Map and Chart Implementation Notes

CwoMaps.js owns the Leaflet map, layer rendering, interactive legend controls and map persistence. CwoCharts.js owns the Syncfusion chart instances and follows the EHMP SFCharts helper pattern. app.js acts as the page controller, connecting form events, persistence, tabs, splitters, API calls, table rendering, chart updates and map selection.

- Map selection is event-driven: a chart point click emits cwo:solution-selected.
- The selected solution updates table state and map highlight from the compact action list.
- Repaired gully markers are derived from actions of type gully, not from a separate repaired-gully dataset.
- Selected spend fill and repaired gully markers are separate Leaflet overlay layers so the legend can switch them on and off independently.
- The chart data source includes solutionIndex so chart clicks can round-trip to the selected solution.

8.7 Appendix G - Database and Persistence Model

Server persistence and browser persistence serve different purposes. MariaDB stores reproducible scenario/run history. Browser localStorage stores user interface preferences such as input values, input-card collapse state and map view.

16. scenarios stores scenario names and settings JSON.
17. runs stores run status, timestamps and summary JSON.
18. solutions stores objective JSON and compact action JSON.
19. run_logs stores timestamped run progress messages.
20. localStorage key cwo_settings_v1 stores form values when persistence is enabled.
21. localStorage key cwo_input_collapse_v1 stores card collapse state.
22. localStorage key cwo_map_views_v1 stores map view by dataset.

8.8 Appendix H - Validation Context

During porting, CWD outputs have been compared with the VB.NET application and, where useful, with the C# reference project. Recent comparison work used the C# implementation as the reference for ecological-status direction, sediment calculation and Pareto reference-marker interpretation. This type of comparison should continue as more of the scenario-running code is ported.

8.8.1 Recommended validation procedure

23. Run the same dataset, budget, action switches, unit costs and objective checkboxes in VB.NET and CWD.
24. Record the baseline row, selected Pareto marker row, number of solution rows and selected action count.
25. Compare sediment, spend/cost and any selected objective values to displayed precision.
26. Click the final point in the Pareto chart and verify the map highlights the same repaired gullies/sub-catchment s expected from the action list.
27. Investigate any difference as either a data-import issue, model-equation issue, action-ordering issue or display-formatting issue.

8.9 Appendix I - Known Technical Issues and Interpretation Rules

- A flat opportunity-cost column is expected when no hillslope repair actions are selected.
- A flat ecological-status column is expected when selected actions do not alter the forest-cover state used by the objective.
- A selected objective appearing in the table means that the objective is stored, displayed and used for Pareto interpretation in the current run.
- The runner may produce fewer displayed solution rows than the Iterations setting, because rows represent archived trade-off solutions rather than every rejected candidate.
- The result map shows the selected solution, not every solution at once.
- The manual link in index.html should be checked whenever the web-library route or manual file type changes.

8.10 Appendix J - Maintenance Notes for Developers

Keep PHP model classes close to the VB.NET names until validation is complete. This makes cross-reference and debugging faster. JavaScript classes should continue to use the Cwo prefix and EHMP-style OOP modules where practical. Database functions should remain centralised in CwoDbUtils.php unless a larger storage redesign is deliberately approved.

28. When adding a new input field, update index.html, app.js request construction, server validation, scenarioSettingsForStorage, output summary text and Appendix A.
29. When adding a new objective, update cls_Scenario, cls_CatchmentMgtModel, chart axis labels, table rendering and Appendix D.
30. When changing map symbology or layer structure, update CwoMaps.js, the legend controls and the manual at the same time.
31. When changing scenario selection or archive logic, document whether the result is deterministic, seeded stochastic, single-objective or multi-objective.

8.11 Appendix K - Literature and Standards Context

CWD combines catchment erosion calculations, network accumulation, scenario optimisation, web mapping, charting, JSON API transport and relational storage. The

references below are included to anchor those technical areas rather than to imply that every method is implemented exactly as described by each source.

8.12 Appendix L - Pareto Graph Technical Notes

The CWD Pareto chart is a two-objective view of the solution set returned by the selected run. Each plotted point represents one stored solution package, not a single management action. The X and Y axis controls choose which objective values are projected into the chart; common pairings are spend or cost against sediment, sediment against ecological status, and spend against opportunity cost. This is consistent with common multi-objective visualisation practice, where scatter plots are used for lower-dimensional objective-space views [15].

In multi-objective optimisation, a Pareto front is the boundary of non-dominated alternatives in objective space. A solution is non-dominated when no other feasible solution is at least as good on every displayed objective and strictly better on one of them. In a minimisation-oriented CWD view, points lower and farther left are usually preferable for objectives such as sediment, spend, cost and opportunity cost. Ecological status is stored with a sign convention in the ported model, so CWD display code inverts that value for user-facing chart comparison [7], [15].

$$x \text{ dominates } y \text{ iff } \forall i: f_i(x) \leq f_i(y) \text{ and } \exists j: f_j(x) < f_j(y)$$

$$PF = \{ f(x) \mid x \in X \text{ and not exists } y \in X \text{ such that } y \text{ dominates } x \}$$

The front should be read as a trade-off curve or trade-off cloud, not as a single automatic answer. Moving along the front usually exchanges one objective for another: for example, a solution can spend more money to reduce sediment, or retain lower spend with less sediment improvement. The Reference marker identifies the real Pareto solution closest to the observed ideal point after objective normalisation, following the C# reference convention. It is useful for navigation, but it is not a universal decision rule for every management context. Decision-support literature treats visualisation and preference selection as a separate step after multi-objective algorithms produce solution sets [11].

CWD renders all solution points, the Pareto subset, and the reference solution as separate visual series. This follows common practice in optimisation tools: plotting libraries commonly distinguish Pareto-front data from the broader objective database, and interactive tools often use scatter, parallel-coordinate or radar views when users need to inspect more than two objectives [12], [15].

$$P_k = (J_x(s_k), J_y(s_k))$$

For spatial decision problems, Pareto-front interpretation is especially important because locations, budgets and environmental outcomes can conflict. Spatial optimisation research emphasises that decision makers need a distributed set of Pareto-optimal alternatives to understand trade-offs between competing objectives, rather than a single weighted-sum answer that may hide parts of the front [6], [20].

8.12.1 Interpretation rules for CWD users:

1. Check the selected X and Y axes before interpreting the chart, because the same run can look different under another objective pairing.
2. Treat dominated points as useful run history or convergence evidence, but do not prefer them over a non-dominated point unless there is an external reason not represented by the selected objectives.
3. Use chart clicks as a navigation tool: selecting a point updates the solution table selection and map highlight for the corresponding action package.
4. Remember that CWD currently plots the objectives stored for the run. If an objective was not selected or did not vary, it may not appear as an axis option or may form a flat series.

5. Do not confuse this chart with a quality-control Pareto chart, which is normally a sorted bar chart based on frequency or magnitude. CWD uses Pareto in the multi-objective optimisation sense: non-dominated trade-off solutions in objective space.

9. Technical Appendix: VB.NET vs CWD Objective Comparison Results

Date added: 2026-07-29. Scope: VB.NET and CWD only; C# reference-app results are intentionally excluded from this appendix.
Comparison basis: deterministic random-series optimisation runs with simulated-annealing acceptance disabled. The legacy VB.NET class name remains SimulatedAnnealing, but these comparison runs use direct Pareto-dominance archive acceptance/rejection only: candidates are accepted when they dominate existing archive solutions or when they are not dominated or equal to an existing archive solution.

9.1 1-objective comparison results

Sediment: VB.NET solutions 7, best iteration 6, best sediment 3439.4; CWD solutions 7, best iteration 6, best sediment 3439.4. Result: aligned.
Ecological status: VB.NET solutions 55, best iteration 54, best ecological status 4.85; CWD solutions 55, best iteration 54, best ecological status 4.85. Result: aligned.
Opportunity cost: VB.NET solutions 1, best iteration 0, best opportunity cost 0; CWD solutions 1, best iteration 0, best opportunity cost 0. Result: aligned.
Cost: VB.NET solutions 1, best iteration 0, best cost 0; CWD solutions 1, best iteration 0, best cost 0. Result: aligned.

9.2 2-objective comparison results

Sediment + ecological status: VB.NET solutions 6, best iteration 744, best sediment 5452.6, best ecological status 4.41; CWD solutions 6, best iteration 744, best sediment 5452.6, best ecological status 4.41. Result: aligned.
Sediment + opportunity cost: VB.NET solutions 3, best iteration 999, best sediment 5424.2, best opportunity cost 56840; CWD solutions 3, best iteration 999, best sediment 5424.2, best opportunity cost 56840. Result: aligned.
Sediment + cost: VB.NET solutions 43, best iteration 485, best sediment 5606.2, best cost 989470; CWD solutions 43, best iteration 485, best sediment 5606.2, best cost 989470. Result: aligned.

9.3 3-objective comparison results

Sediment + opportunity cost + ecological status: VB.NET solutions 19, best sediment 6354.4, best opportunity cost 119511, best ecological status 4.37; CWD solutions 21, best sediment 6351.9, best opportunity cost 131947, best ecological status 4.36. Result: not aligned.

9.4 Interpretation

The 1-objective and 2-objective VB.NET versus CWD comparisons align on solution counts, best iterations, and best objective values. The 3-objective comparison does not align: CWD produces a different archive size and different best objective values.
Observed cause: the first semantic divergence occurs during the rejected-candidate undo path. VB.NET emits an UndoMismatch trace after undoing altered actions, while CWD continues without an equivalent undo verification trace. The likely CWD repair point is

the budget recovery and undo sequence in CwdScenarioRunner.php, specifically recoverBudgetForAction and undoAlteredActions.

10. Technical Appendix: Creating Partial USLE Inputs for New Catchments

10.1 Purpose

CWD should continue to treat Partial_USLE.asc as the normal model input. The importer can, however, offer a guided preparation path that creates Partial_USLE.asc from separate R, K, L, and S factor rasters before the catchment manifest is saved.

10.2 Recommended import workflow

1. Let the user choose either Use existing Partial_USLE.asc or Create Partial_USLE.asc from R, K, L, and S rasters.
2. If creating it, validate all source rasters against the same ASCII grid contract used by the catchment rasters.
3. Calculate each valid output cell as Partial_USLE = R x K x L x S, preserving NoData where any source cell is NoData.
4. Save the generated raster as Partial_USLE.asc in canonical ArcInfo ASCII Grid format and map it automatically in the manifest.

10.3 Support for source rasters

R rainfall erosivity can be supplied as a raster or as a single constant value applied to the catchment when justified by the data source.
K soil erodibility can be supplied as a raster, or generated by reclassifying a categorical soil raster using a CSV lookup table such as soil_code,k_value.
L slope length should initially support either a supplied raster or a constant value of 1. Pantus et al. [13] describe L = 1 as an accepted simplification for the original case.
S slope factor should normally be supplied from GIS terrain processing based on a DEM. The importer may accept the result and validate it, but should not initially attempt to reproduce full terrain-analysis processing in PHP.

10.4 Safeguards

The builder should check matching ncols, nrows, xllcorner, yllcorner, cellsize, NODATA_value, and canonical cellsize headers before any calculation.
It should reject negative R, K, L, or S values, warn on zero-heavy rasters, report NoData coverage inside the catchment, and report min, max, and mean values for each factor and for the generated Partial_USLE raster.
The manifest should record provenance for generated Partial_USLE.asc, including source filenames, constants, lookup table names, formula, and generation date.

10.5 Boundary of responsibility

The importer should guide and validate raster preparation, but it should not become a full GIS. DEM processing, slope derivation, cost-distance preparation, and specialist spatial analysis should remain in QGIS, ArcGIS, GRASS, SAGA, or equivalent GIS tools. CWD should consume their exported ASCII rasters and report whether they are internally consistent.

11. Technical Appendix: Creating CWD ASCII Grids in QGIS

11.1 Purpose

CWD catchment rasters should be supplied as ArcInfo ASCII Grid files. In GDAL terminology this is the AAIGrid driver, an ASCII interchange format for Arc/Info Grid. CWD expects an .asc text grid with ncols, nrows, xllcorner, yllcorner, cellsize, and NODATA_value header rows, using the canonical cellsize header and NODATA_value -9999 [5].

11.2 General export workflow

1. Load the source raster or derived raster into QGIS.
2. Confirm the layer CRS is the intended catchment CRS. If needed, reproject first so all CWD rasters use the same coordinate system.
3. Align the raster to the catchment grid before export. Use the same extent, origin, pixel size, and number of rows and columns as the reference catchment raster, normally Subcatchments.asc.
4. Right-click the raster layer in the Layers panel, choose Export, then Save As. QGIS documents raster export through Export > Save As and notes that raster outputs can be written to GDAL-supported formats with user-set CRS, extent, resolution, creation options, and NoData values [16].
5. Set Format to Arc/Info ASCII Grid or AAIGrid, set the output filename with an .asc extension, and set NoData to -9999.
6. Use the layer extent and resolution only when they already match the catchment grid. Otherwise use the aligned/intermediate raster prepared in the previous step.
7. Save the file into the new catchment data directory and use the CWD new-catchment validator before creating the manifest.

11.3 Creating derived factor rasters

For a formula-based raster such as Partial_USLE.asc, use Raster Calculator to combine aligned input rasters. QGIS describes Raster Calculator as a raster algebra tool whose output values are computed from an expression using loaded raster layers [17]. For Partial USLE, calculate R x K x L x S, then export the result as ArcInfo ASCII Grid.

For a constant factor raster, use Raster Calculator with an aligned reference raster, for example reference@1 * 0 + value. This creates a full-grid raster with the required extent and cell size.

For categorical source data such as soils or land-use polygons, first rasterize the vector layer onto the same grid as the reference raster. Then reclassify codes to the numeric CWD values, for example soil code to K factor or land-use class to opportunity cost.

For slope-derived rasters, derive slope or terrain factors in QGIS, SAGA, GRASS, or another GIS tool, then align and export the final factor raster. CWD should validate the exported raster; it should not be treated as the terrain-processing tool.

11.4 Creating the river network and subcatchment linkage

Pantus et al. [13] treat the river network as an input used for bank-erosion calculations, riparian-buffer creation, and subcatchment linkage. The practical GIS workflow is to

derive or prepare the stream network and subcatchments from the same DEM or aligned hydrology processing workflow so the resulting network, subcatchment raster, and buffer rasters use the same grid.
For each subcatchment, the CSV must identify the downstream subcatchment. Use -1 for the terminal downstream subcatchment. Pantus et al. [13] note that this downstream linkage can be obtained from GIS hydrology tools.
The same river network is used to calculate ChannelLength as stream length within each subcatchment and ChannelSlope as average river-channel slope within each planning unit. The Bremer implementation used DEM, LiDAR, imagery, and field-derived channel information reported by Olley et al. [8].
Create the riparian buffer from the river network, then rasterize it to the same grid as the subcatchment raster. Pantus et al. [13] used a 25 m riparian buffer and recommend delineating the river network and subcatchments together so they remain spatially consistent.
Before import, check that every subcatchment has a valid Identifier, exactly one DownstreamId path to the terminal outlet, no downstream loops, and non-negative channel length and slope values.

11.5 Required checks before import

Every CWD raster in a catchment must have the same ncols, nrows, xllcorner, yllcorner, and cellsize. The files should use NODATA_value -9999 and should not use dx/dy header rows. GDAL may emit dx/dy instead of cellsize for non-square pixels unless output is forced to square-cell ASCII Grid or the raster is resampled first [5].
Check that NoData does not unexpectedly occur inside the active catchment area, especially in Partial_USLE.asc, Ground_cover.asc, Distance_buffer.asc, Opportunity_cost.asc, and Landholder_willingness.asc.
Check that factor rasters used to create Partial_USLE.asc contain no negative values. Zero values may be valid in some cases, but large zero areas should be reviewed because they can remove hillslope contribution from those cells.

11.6 Recommended file names

Use stable, descriptive filenames: Subcatchments.asc, Partial_USLE.asc, Ground_cover.asc, Riparian_buffer.asc, Distance_buffer.asc, Opportunity_cost.asc, and Landholder_willingness.asc. Source-factor rasters may be kept as R_factor.asc, K_factor.asc, L_factor.asc, and S_factor.asc for provenance.

References

1. Agafonkin, V., 2024. Leaflet documentation: an open-source JavaScript library for mobile-friendly interactive maps. Available at: https://leafletjs.com/

2. Bray, T., 2017. The JavaScript Object Notation (JSON) Data Interchange Format. RFC 8259. Internet Engineering Task Force. Available at: https://doi.org/10.17487/RFC8259

3. Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 6(2), pp.182-197.

4. Fielding, R.T., 2000. Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation. University of California, Irvine.

5. GDAL/OGR contributors, 2026. AAIGrid -- Arc/Info ASCII Grid. GDAL documentation. Available at: https://gdal.org/en/stable/drivers/raster/aaigrid.html

6. Hermoso, V., Pantus, F., Olley, J., Linke, S., Mugodo, J. and Lea, P., 2012. Systematic planning for river rehabilitation: integrating multiple ecological and economic objectives in complex decisions. Freshwater Biology, 57(1), pp.1-9. Available at: https://doi.org/10.1111/j.1365-2427.2011.02693.x

7. MathWorks, 2026. paretoplot - Pareto plot of multiobjective values. MathWorks documentation. Available at: https://www.mathworks.com/help/optim/ug/optim.problemdef.optimizationvalues.paretoplot.html

8. Olley, J., Ward, D., Pietsch, T., McMahon, J., Laceby, P., Saxton, N., Rickard, B., Rose, C. and Pantus, F., 2009. Rehabilitation priorities Knapp Creek. Report to Healthy Waterways for the Healthy Country Project, Brisbane. 42 pp.

9. Olley, J., Burton, J., Smolders, K., Pantus, F. and Pietsch, T., 2013. The application of fallout radionuclides to determine the dominant erosion process in water supply catchments of subtropical South-east Queensland, Australia. Hydrological Processes, 27(6), pp.885-895. Available at: https://doi.org/10.1002/hyp.9422

10. OpenStreetMap contributors, 2026. OpenStreetMap geographic data and tile attribution. Available at: https://www.openstreetmap.org/copyright

11. Osika, Z., Zatarain Salazar, J., Roijers, D.M., Oliehoek, F.A. and Murukannaiah, P.K., 2023. What lies beyond the Pareto front? A survey on decision-support methods for multi-objective optimization. Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23, pp.6840-6848. Available at: https://doi.org/10.24963/ijcai.2023/755

12. ParMOO, 2026. The Interactive Visualization (viz) Library. ParMOO documentation. Available at: https://parmoo.readthedocs.io/en/latest/modules/viz.html

13. Pantus, F., Hermoso Lopez, V., Lea, P., Olley, J., Linke, S. and Mugodo, J., 2011. Catchment Rehabilitation Planner. Report to Healthy Waterways Partnership Ltd., Australia. 75 pp. ISBN: 978-1-921760-46-4.

14. Prosser, I.P., Rustomji, P., Young, W.J., Moran, C.J. and Hughes, A.O., 2001. Constructing river basin sediment budgets for the National Land and Water Resources Audit. Canberra: CSIRO Land and Water.

15. pymoo, 2026. Scatter plot. pymoo documentation. Available at: https://pymoo.org/visualization/scatter.html

16. QGIS Documentation Team, 2026a. Creating new layers from an existing layer: raster export. QGIS User Manual. Available at: https://docs.qgis.org/3.10/en/docs/user_manual/managing_data_source/create_layers.html

17. QGIS Documentation Team, 2026b. Raster calculator. QGIS User Manual. Available at: https://docs.qgis.org/testing/en/docs/user_manual/processing_algs/qgis/rasteranalysis.html

18. Renard, K.G., Foster, G.R., Weesies, G.A., McCool, D.K. and Yoder, D.C., 1997. Predicting Soil Erosion by Water: A Guide to Conservation Planning with the Revised Universal Soil Loss Equation (RUSLE). USDA Agriculture Handbook No. 703.

19. Wilkinson, S.N., Prosser, I.P., Rustomji, P. and Read, A.M., 2009. Modelling and testing spatially distributed sediment budgets to relate erosion processes to sediment yields. Environmental Modelling & Software, 24(4), pp.489-501.

20. Xiao, N., Bennett, D.A. and Armstrong, M.P., 2008. Seeking the Pareto front for multiobjective spatial optimization problems. International Journal of Geographical Information Science, 22(5), pp.507-526. Available at: https://doi.org/10.1080/13658810701492365