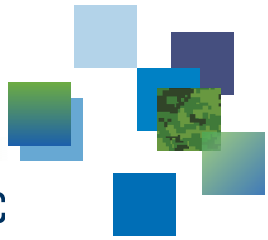
Defence Research and Development Canada
Recherche et développement pour la défense Canada



DRDC | RDDC
technologysciencetechnologie

Joint Exercise Training Allocation decision support software: Instruction manual and database design

Matthew R. MacLeod
Mark Rempel
DRDC – Centre for Operational Research and Analysis

Defence Research and Development Canada
Reference Document
DRDC-RDDC-2018-D054
June 2018

Canada

IMPORTANT INFORMATIVE STATEMENTS

This document was reviewed for Controlled Goods by DRDC using the Schedule to the *Defence Production Act*.

Disclaimer: This publication was prepared by Defence Research and Development Canada an agency of the Department of National Defence. The information contained in this publication has been derived and determined through best practice and adherence to the highest standards of responsible conduct of scientific research. This information is intended for the use of the Department of National Defence, the Canadian Armed Forces (“Canada”) and Public Safety partners and, as permitted, may be shared with academia, industry, Canada’s allies, and the public (“Third Parties”). Any use by, or any reliance on or decisions made based on this publication by Third Parties, are done at their own risk and responsibility. Canada does not assume any liability for any damages or losses which may arise from any use of, or reliance on, the publication.

Endorsement statement: This publication has been published by the Editorial Office of Defence Research and Development Canada, an agency of the Department of National Defence of Canada. Inquiries can be sent to: Publications.DRDC-RDDC@drdc-rddc.gc.ca.

© Her Majesty the Queen in Right of Canada, Department of National Defence, 2019

© Sa Majesté la Reine en droit du Canada, Ministère de la Défense nationale, 2019

Abstract

Canadian Joint Operations Command (CJOC) Joint Training Authority (JTA) staff requested that the CJOC Operational Research and Analysis (OR&A) team design and implement a transparent, repeatable, evidence-based approach to support the development and refinement of a five-year rolling exercise program consistent with government policy and force posture direction. This reference document describes elements of the technical design and practical use of the decision support software CJOC OR&A built to respond to this problem statement.

Résumé

Le personnel du responsable de l'instruction interarmées (RII) du Commandement des opérations interarmées du Canada (COIC) a demandé à l'équipe d'analyse et de recherche opérationnelle (ARO) de concevoir et de mettre en œuvre une méthode transparente, reproductible et fondée sur les faits pour appuyer l'élaboration et la mise au point d'un programme d'exercice quinquennal conforme à la politique gouvernementale et à l'orientation en matière de posture des forces. Le présent document de référence décrit les éléments de la conception technique et de l'utilisation pratique du logiciel d'aide à la décision que l'équipe d'ARO du COIC a créé pour répondre à cet énoncé de problème

Table of contents

List of figures

List of tables

Acknowledgements

The authors would like to acknowledge Raman Pall for his work on developing the initial spreadsheet tools and data collection process. The contributions of Joint Training Authority (JTA) staff, especially Wayne Douglas, Roy Forestell, Ivy Miezitis, and Ryan Kenick have been essential to gathering and preparing the data.

1 Introduction

Canadian Joint Operations Command (CJOC) Joint Training Authority (JTA) staff have requested that the CJOC Operational Research and Analysis (OR&A) team design and implement a transparent, repeatable, evidence-based approach to support the development and refinement of a five-year rolling exercise program consistent with government policy and force posture direction. The following problem statement was developed and subsequently endorsed by Commander CJOC [1]:

> To provide a rigorous means of selecting a set of joint exercises that can be conducted in a given period optimally aligned with government policy and force posture direction, subject to the constraints required of the JTA:
>
> - all contingency plans (CONPLANs) (for which opportunities exist) must be exercised at a specified frequency;
> - all tasks in the joint task list (for which opportunities exist) must be provided an opportunity to be validated at a specified frequency;
> - all geographic regions (for which opportunities exist) must have at least one exercise conducted within their boundaries at a specified frequency; and
> - the sum of the selected exercises' costs in each fiscal year will not exceed the available Joint Exercise Training Account (JETA) budget in that fiscal year.

This Reference Document describes elements of the technical design and practical use of the decision support software CJOC OR&A built to respond to this problem statement. More information on the application and purpose of the software can be found in an earlier Scientific Letter [2], and in a Scientific Report currently being drafted [3].

1.1 Brief background

To motivate the creation of this document, a short history of the development of the software is required. The initial versions and structure were built by one of the authors (Mr. Rempel) and Mr. Raman Pall from late 2016 through early 2017. In August 2017 Mr. Pall left the organization and the other author (Mr. MacLeod) joined the team, and was given primary responsibility for maintaining and developing the tool. As of this writing, Mr. MacLeod is about to depart on extended leave, as the CJOC JTA prepares for next year's JETA funding cycle, with the possibility that another scientist may become involved in the interim. As such, the authors felt it was important to document the current state of the software and how it is being used, especially as the JTA had made significant changes to the data structure, and Mr. MacLeod had made significant modifications to the processing. It is also being used to provide a reference for implementation information that would be too detailed for inclusion in [3] and other scientific publications.

1.2 Contributions

The report will document three main contributions of the underlying work done to support the analysis. Underpinning the entire system is the design of database, which was primarily developed in early 2017. This contribution will be described in Section 3. The bulk of the document, however, will focus on how to use the scripts that execute the analysis, the development of which over the Fall–Winter of 2017–2018 involved two main developments.

The first of these developments was the ability to import data directly from the spreadsheets developed by the client and write it into the database in a single script. This minimizes the chance of data transcription errors, and also allows new versions of the client data to be imported rapidly. The main risk to this approach is that the client may change the structure of their spreadsheets,[1] or provide data that is not sufficiently consistent for the import scripts to cope with.

The second of these developments was to perform the data manipulation and generation of plots at the analysis phase by pulling the data tables into *R* and using the `dplyr` package, rather than passing Structured Query Language (SQL) queries to the database that returned results that could be plotted directly. This allowed the code to be written in an arguably more human-readable form, and also provided greater flexibility in adjusting the data in ways that would support the output. The main downside is a potential performance penalty, but given the relatively small dataset (hundreds of rows with tens of variables), this is not practically noticeable.

2 Processing the data

The relevant files for this activity are stored on the DRDC – Centre for Operational Research and Analysis Defence Research Establisment Network (DRENET) common drive:
`\\CJOC ORA Team\joint_exercise_portfolio_optimization\`

Important sub-directories are:

- `data`: data provided by the client, as well as a file containing conversions from the criterion levels to numerical values. When new data files are obtained, new sub-directories are created to hold them within this sub-directory, with naming convention `YYYYMMDD`.
- `model`: contains deterministic and stochastic versions of the mathematical model in sub-directories so named. The stochastic model has not been updated recently. The deterministic model has been updated several times, with the most recent being `jeta_model_deterministic_v10.mod`.
  Also of interest may be `jeta_model_deterministic_v10_TJ.mod`, which includes a constraint to force the inclusion of a specific exercise.

[1] Indeed, as of this writing this is underway for the next fiscal year (FY) cycle.

- `results`: contains spreadsheets and graphics storing the results of optimization runs, as well as any exploratory data analysis. When new data is obtained, sub-directories are created to store the associated output, with the same naming convention as above (`YYYYMMDD`).
- `scripts`: contains the *R* scripts used to process the data and results. The most current versions of this writing are stored in the subdirectory `JETA-R`, with an *R Studio* project file `JETA-R.Rproj`. The three key scripts which should be read in conjunction with this report are:
  - `ConvertInputData.R`—this script reads in data from the JTA provided files and writes it into an SQL database.
  - `processResults.R`—this script takes the results of an optimization model run and writes them into the database.
  - `exploratoryResults.R`—this script generates analysis tables and plots based on the contents of the database.

One must also have an Open Database Connectivity (ODBC) data source named `jetaDb` to store the data and results. Setup of this data source is beyond the scope of this document.

2.1 Preparing the data

As of this writing, the client is providing data on each exercise in two separate files. One contains the data on which Joint Mission Essential Tasks (JMETs) each exercise offers the opportunity to validate,[2] and the other contains all other associated data.[3] The JMET spreadsheet is assumed to have a tab named `FY 18-19`, and the other a tab named `L1 JETA Exercises`, each with the relevant data. Importantly, the names used in the `Exercise` column in the JMET spreadsheet must match those in the `Exercise name` column in the other spreadsheet, for the data to be correctly associated. Newly received data should be placed in a new subdirectory under `data`, named with that day's date (which may or may not correspond to the date in the filename provided by the client).

Note that currently the data ingestion cannot handle duplicate exercise names. This may occur where multiple level 1s (L1s) make separate budgetary requests for different activities covered under the same umbrella exercise (e.g., VIGILANT SHIELD 19). The matching of the lines of data in the two spreadsheets relies on them having unique names that directly correspond. While it would be possible to also match on the L1 fields in the two spreadsheets, this is an added layer of complexity, and simply moves the problem downstream to the output stage. It is recommended that in these situations one of the exercises be renamed.

[2] Currently maintained by Roy Forestell, with a filename of the form `YYYYMMDD-U-3000-CJOCHQ_J7_JTA-JETA_L101_JMETs_V2.xlsx`, stored in the JTA sharepoint in the Canadian Armed Forces (CAF) Joint Task List section.

[3] Currently maintained by Wayne Douglas, with a filename of the form `YYYYMMDD-U-3000-CJOCHQ_J7JTA-JETAandL101_Ex_List.xlsx`, stored in the JTA sharepoint under the Joint Training Advisory Group (JTAG) Conference Slides section.

If there are duplicates, this will become clear when the optimization model is run, with an error concerning a 'duplicate tuple.'

A third spreadsheet, `Conversions.xlsx` was developed to allow the criteria levels and weights to be coded as per the value model (described in more detail in [3]). It contains a single tab for each criterion (using their short names as specified in the database design in Section 3), with a column for the plain-language name of the criteria levels, and one for their corresponding weight. Finally, there is a tab called `weighting` that defines the overall weight of each criterion. It is important that the plain language criteria levels (e.g., 'Negligible') match those used in the client provided spreadsheet, as they are used as look-up values.[4] If there are no updates to be made to this spreadsheet, it should simply be copied from the next most recent data directory.

2.2 Importing the data

The next major step is to import the data from the spreadsheets using the script `ConvertInputData.R`. *Before doing anything else*, one should update the variables `dataDir`, `exFile`, and `jmetFile` to correspond to the new date of the dataset. It is also possible to update the sheet names within the provided files, if necessary. One can also adjust the budgets used for each of the fiscal years to be considered by updating the variable `budgetTable`. The authors followed the convention that had been put in place previously, where a FY is defined by the calendar year in which it ends (e.g., FY 18/19 is referred to as '2019').

With that in place, running this script will process the spreadsheets provided into the database. Several caveats should be made regarding the current state of the code, most of which are related to having to adapt the old code to use only a single year's worth of data. These are flagged with appropriate comments in the script file as well:

- The database is not so much updated as rebuilt each time the data is imported. More technically, each table is dropped and then replaced by a new table. This may fail if the database is under active use (e.g., if a query is active in *Access*).
- Several 'magic' values are defined at the top of the file, to make explicit that they are currently being used as constants. Since only one year of data at a time is being provided by the JTA, every exercise can be assumed to start in that year (currently 2018–2019, known as 2019). This is also treated as the 'centroid year' (i.e., the centre of the window in which the constraints must be met). Since the other parts of the model are only currently set up to work with a single dataset in the database, a fixed identification number is assigned to the dataset. Most notably, a magic number[5] is

[4] It was noted in developing the new scripts that some typos had been codified in the old input process, which convoluted the associated look-up tables.

[5] This was set to 1015 for consistency with the old input method. This could probably be set to something more distinctive, in which case the optimization model file would have to updated to match. Alternatively, 'None specified' could be removed from the `conplanValidateTable`, so it is never required to be validated.

set for the value 'None Specified' for the CONPLAN attribute. This is required so that the optimization model does not require that an exercise with 'None specified' be included in the selected portfolio.

- As of this writing the column 'L1' is being dropped when `exerciseTable` is being created, as it is not being used as a decision variable. If at a later stage the client wishes to filter the results by 'L1', the code may need revisions to ensure this column is preserved throughout (and the database structure will need revision).

- The values generated by the script for entry into `conplanValidateTable`, `geoOccurrenceTable` and `taskValidateTable` set the constraints such that the 'magic' year is the centre of all the constraint windows, and so the model will attempt to meet all constraints in that year. This will need to be updated if different windows or frequencies are desired, which will also need a new input format to be defined.

- The way the total exercise value is calculated treats the underlying data as a matrix that exists in specific numbered columns in the exercise table, so is potentially fragile to new columns being added to the input file. The authors could not find a clear way to multiply columns in a data frame by weights, and so resorted to using matrix math to multiply them through.

- The file `exFile` is read from twice (the second time to read in the cost data). As reading from the *Excel* files is the slowest operation conducted by the script, it would likely be more efficient to read all the data from the file once, and break it up into separate data frames in memory.

- All activities are currently given the 'magic' start year, as they are all assumed to be occurring in the one year for which cost data has been provided. This code will need to be adjusted once the data format for multiple years is agreed. It is assumed an activity (whether planning or execution) exists if and only if there is a JETA money request associated with it.

- The structure of the JMET spreadsheet precludes it being read in with meaningful column names for some of the left-most columns, which makes it potentially fragile to changes in the first few columns (as they are dropped based on position, not by name). That said, the JMET names are created directly from the data file, so if these are changed, or if additional columns are added in the same structure, the script should be robust to this. The names are subjected to some cleaning to remove newline characters and the like.

- The JMET data table is 'left joined' to the `exerciseTable`. The upshot of this is that any extra exercises in the JMET data[6] will be dropped, while exercises in the other dataset that do not have a match will be kept. It is here that any differences in exercise naming will become an issue.

[6] e.g., L101-funded exercises, or former exercises now deemed to be operations.

At this stage all the data necessary to run the optimization model will have been loaded into the database.

2.3 Running the optimization model

Prior to running the model, the two `printf` statements in the final lines of the `.mod` file will need to be updated to correspond to the result subdirectory for the correct date (this subdirectory should also be created, if it hasn't already). They should also be updated to correspond to the version of the model, if a new one has been created. The result filenames are also currently created to include the budget for 2018–2019, to prevent the overwriting of result files by subsequent runs with different budgets (the most common comparison requested by the client). If there is no budget for this year in the database, this will have unpredictable results.

Assuming that the *GLPK* solver has been installed by the user and is in the path, the optimization model can be run from the `model\deterministic` directory using a command like: `glpsol -m jeta_model_deterministic_v10.mod`, where the filename is adjusted as needed for new versions of the model. Remember that errors concerning a 'duplicate tuple' can generally be traced to duplicate exercise names in the provided data.

In general, the model is currently setup to only require that the solution meet constraints that are possible to meet with the given data; e.g., if a CONPLAN does not appear in the dataset, it is not required to be exercised. The exception to this is the budget constraint: therefore a failure to find a feasible solution is almost certainly due to the budget provided.

A common request from the JTA has been to generate portfolios with one or more specific exercises forced into the solution, to compare against the solution generated by the default model.[7] At present this can only be done by manually adding constraints to the optimization model. An example can be found in:

`jeta_model_deterministic_v10_TJ.mod`

which contains a constraint of the form:

```
subject to exercise_name:
x[1031,2019] == 1;
```

This constraint forces the exercise with `exercise_id` of 1031 to be included in 2019's budgeting. If exercises are added or removed from the input data, the `exercise_id` of exercises will change, so this is a very fragile process. After running `ConvertInputData.R` the user can either open `exerciseTable` in *R* or open their `jetaDb` in *Access* to look up the `exercise_id` they need to set one or more such constraints.

[7] This is usually due to the default solution not including one or more of the most expensive exercises.

Assuming a feasible solution has been found, the results will be written into a `csv` file in the `results` directory, named according to the 2019 budget and the version of the model used.

As a final note, the 2019 budget is also currently being written into the `result_id` column, to allow result sets with different budgets to be distinguished in the analysis phase (e.g., results with the same budget but other different constraints, or with different budgets in the out years). There may be a more robust way to construct unique identifiers, but the GNU Mathematical Programming Language (GMPL) language is not overly flexible. In older versions of the scripts this column was simply filled with the value 1.

2.4 Loading the results into the database

While the optimization model reads from the database, it is not capable of writing the results back to the database.[8] Therefore, a separate script called `processResults.R` is used to load the results into the database, so that they can be used for analysis along with the input data.

As is a recurring theme, *before doing anything else*, one should update the `dir` and `resultsFileName` variables to correspond to the latest results directory and file name. After that, the script can be sourced.

Unlike the data import, this script has been configured to *add* results to the database, rather than replace them, to enable the results of different budget levels (as presently distinguished by the `result_id` column) to be directly compared. However, if multiple runs are made the user will likely eventually want to clear older or mistakenly generated results out, which can be done by uncommenting the `sqlDrop` command.

The script also runs a number of SQL queries, the results of which are loaded into data frame objects labelled `q1` through `q4`, as documented in the file. These have not been updated in some time, so should be validated before being relied upon.

2.5 Plotting the results

Plotting of the results is currently done in a script called `exploratoryAnalysis.R`. This is somewhat misnamed, as it now contains a mix of exploratory analysis of the dataset as well as plots related to the results. As `processResults.R` modifies the database when run, it is not desirable to include plotting code that may be tweaked and run multiple times therein.

Once again, *before doing anything else*, the variable `dir` should be modified to point to the most recent results directory. The script can then be sourced, which will create a number of graphic files in the `results` directory, while printing some information to the screen.

[8] The authors are exploring the use of the *Rglpk* package to allow *R* to directly call the optimization model and receive the results, which should allow a cleaner resolution of this issue.

The script is currently in a transitory phase, with the newer code for plotting results from multiple budget levels against each other at the top. As described in section 1.2, it reads many of the tables back out of the database in order to work with them using `dplyr` and related *R* packages. It makes heavy use of the `filter`, `select`, `group_by` and various `join` functions to manipulate the data frames into forms that can be used to produce the desired plots using the `ggplot2` package. Converting variables that are defined as strings into `factor` variables, and ensuring that the levels within the `factor` are in the desired order for output, is the other major feature of this code.

Below the newer code, there is a comment indicating that the code '`Gets old school past this point`.' This code is left in the file primarily for reference, and much of it is written to deal with data sets that include multiple years of exercise data—which should become newly relevant in the future. Unlike the new code which relies on `dplyr`, the older code is built mostly around passing SQL queries to the database, and plotting the returned tables fairly directly with `ggplot2`. To respond to client requests for modifying the output in Fall 2017, several less than ideal modifications were hastily made to this code to allow e.g., the colour coding of different exercise types, that required the introduction of additional tables into the database. Some of this code will therefore not work as expected without those manually created tables.

2.6 Guidance on future changes

Based on what is known on future planned changes to data collection, the front-end of `ConvertInputData.R` will require changes to read in next year's exercise data. Currently data is read from the JTA provided spreadsheets with three statements beginning with the following:

```
rawCosts <- read_xlsx(exFile, exSheet) %>% select(ex_name = ‘Exercise
name‘, costPlan = ‘CTES Data FY18/19 Planning ($CAD)‘, ...)

exerciseTable <- read_xlsx(exFile, exSheet) %>% select(ex_name =
‘Exercise name‘, focus = ‘Exercise Focus‘, ...)

jmetTable <- read_xlsx(jmetFile, jmetSheet, skip=2) %>% select(-X__1,
-X__3, -X__4,-X__5) %>% filter (!is.na(X__2))
```

As noted in Section 2.2, it is straightforward enough to adjust the filenames defined as `exFile` and `jmetFile`. What will be more fragile is if the structure of the input files themselves changes.

The command to read in from `exFile` depicts how it is relatively straightforward to read in from an *Excel* file with clear column names in the first row; e.g., the column with title '`Exercise name`' can be directly assigned into a data frame column entitled `ex_name`, to match the names expected in the database. This code should be fairly easy to adapt to a new sheet, assuming titles of data fields remain in the first row.

Reading from the current JMET spreadsheet is slightly less straightforward. The option `skip = 2` is used on initial read, as there is an overall title spanning row 1 of the spreadsheet, and grouping titles of 'Operational Functions' in row 2 above the actual JMET names themselves in row 3. As the titles in cells in columns A through E are in merged cells spanning rows 2 through 4, they do not get picked up when using row 3 as the column title row. These columns are instead assigned titles of the form `X__n`, four of which are not needed by the scripts and are immediately dropped using the `select` command. The exercise name is currently in column `X__2`, which later gets replaced with `ex_name` in the same command that replaces the textual name of each JMET with its assigned `task_id`:

```
colnames(jmetTable) <- c("ex_name",taskTable$task_id[match(taskNames,
taskTable$task_name)])
```

By adjusting the commands above one should be able to adapt `ConvertInputData.R` to enable continued used of the rest of the *R* scripts mostly unchanged. One potential caveat is that the JTA staff has discussed using separate input sheets per L1 organization, which may require additional code to first append the input together. While it is currently commented out of the code, there is logic in the code to deal with situations where not all of the exercises in the dataset are requesting JETA funding, by adding an additional true/false column to `exerciseTable`.

3 Relational database

This section describes the structure of the relational database developed for the JETA decision support tool, as well as some ancillary tables. This is succinctly conveyed by Figure 1 and Tables 1 through 12,[9] but the text of this section will call attention to a few particularities. Most notably, the database structure was developed to address a multi-year decision problem with stochastic elements, whereas the tool is currently only being applied to answer questions about a single year with deterministic costs. These are left as-is to allow for future growth.

One of the elements that recurs throughout the database is the `dataset_id` key. This would allow the storage of multiple different input datasets in the database at once. In practice the tool has not been built to allow for this, nor have questions from the JTA staff required its use—there has only ever been one up-to-date set of data inputs, with the primary exception of the budget level (which is more of a variable constraint than data). That said, this key could be useful for testing, although the current method of completely dropping and adding tables upon data input would have to be reconsidered, with some allowance for non-destructive editing of existing data.

[9] Table 13 documents one additional table that is not part of the relational model, but is stored in the database for convenience.

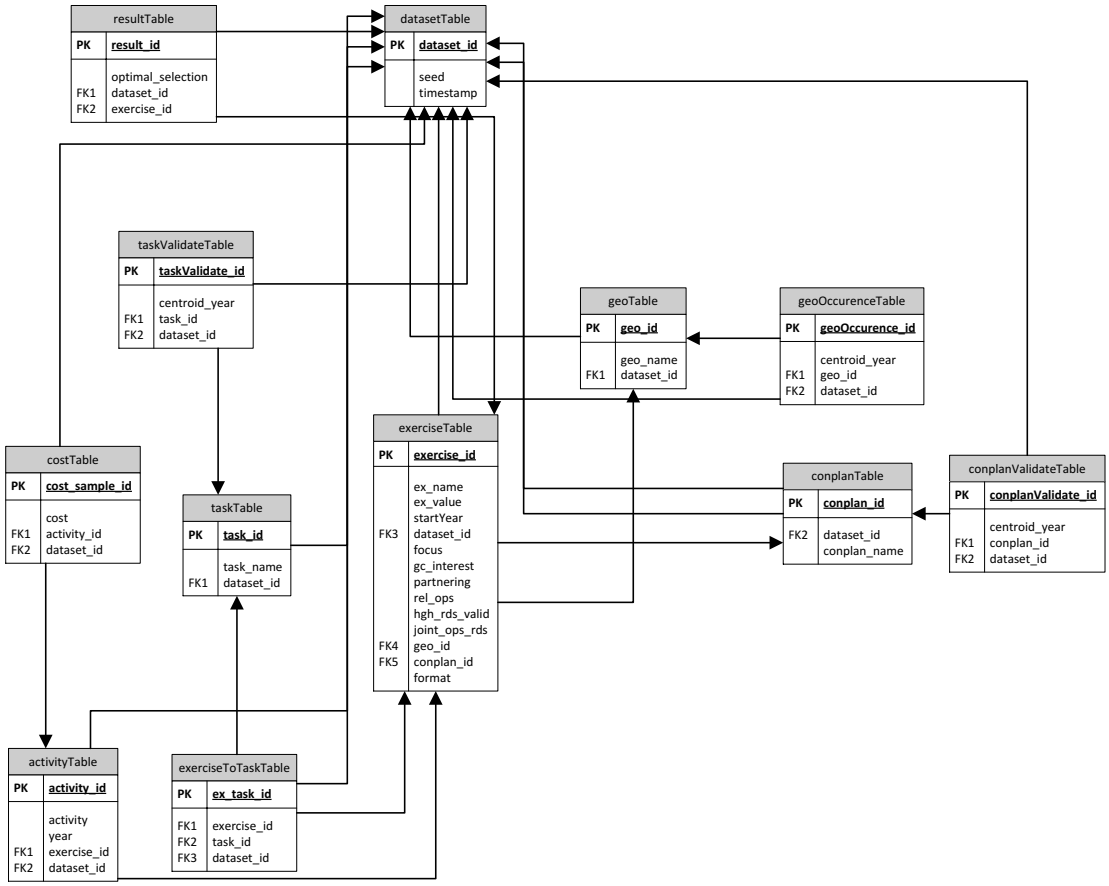


Figure 1: *Relational database model.*

As noted in Section 2.6, at times the input data provided by the JTA has contained information on some exercises that were not actually requesting JETA funding. These need to be excluded from the optimization process, and so an extra column was added to `exerciseTable` (see Table 1) called `JETAfunded`, which was '1' if the exercise was requesting JETA funds and '0' if false. The code to supports this remains in `ConvertInputData.R`, but is commented out, and so is not represented in Figure 1 or Table 1.

The `costTable` (see Table 7) originally made provision for multiple 'samples' of a cost to be assigned to an activity, to allow for uncertainty in costing. This has not been used, as stakeholders are making fixed budget requests (although CJOC did provide some ranges on some estimates for the work done in Fall 2017 [2]). Relatedly, the `seed` column in `datasetTable` (see Table 11) is currently set to a constant value of 1, as random samples of activity costs are not being generated.

Currently the `result_id` column in `resultTable` (see Table 12) is being populated with the budget level for the upcoming financial year, as this is the primary variable being used to compare different planning scenarios. As noted in Section 2.4, at present result sets are being added to the database for each new run of `processResults.R`, unless a line is commented out to drop the existing results. Care should therefore be taken if multiple scenarios are, for instance, run with the same budget level but with different constraints enabled, as the rows of the database will no longer be unique.[10]

While not strictly forming part of the relational model, `budgetTable` (see Table 13) is used to facilitate loading of the budget constraints into the optimization from `ConvertInputData`, where it is generated using a line of code like the following:

```
budgetTable <- data.frame(year=c(2018,2019,2020), budget=c(25000000,
40000000,25000000)) %>% add_column(budget_id = 1001:(1000+nrow(.)))
```

and directly added to the database.

3.1 Potential future changes

The database as currently designed contains primarily—if not exclusively—the data required to run the optimization model. However, the JTA staff frequently requests plots filtering or highlighting exercises based on factors not used by the model: e.g., whether an exercise is physical or virtual, whether it is focused on international engagement, or which organization is leading it. These additional fields could be worked directly into the base model, or alternatively (as the author has done to this point), custom tables which can temporarily be added to the analyst's local database with the required fields along side an `exercise_id` and/or name, and joined to the results using `dplyr` in *R*.

Another aspect not directly present in the current model is the ability to compare against portfolios manually generated by the JTA staff. The author manually created a `coaTable` (COA = course of action) to store this table, although the design was not finalized or compatible with the current input. If these questions recur, it would be worth considering how to represent these portfolios in the database.

Finally, the use of the primary key of `resultTable` to store budget values should be resolved, which may involve defining `budgetTable` in a more structured a way. A more general solution to this problem would be allowing the constraints that the JTA staff wish to vary (e.g., also constraints on forcing the inclusion of exercises) to be defined as data input, rather than directly coded into the models and/or scripts.

[10] This is an abuse of the database design, as a primary key should be completely unique for each row. One could consider using the `dataset_id` to store this information instead, but as that value is a key across all other tables, this would have further ramifications across the database design.

Table 1: *Table definition — exerciseTable.*

Column name	Key	Format	Description
exercise_id	Primary	Numeric	Unique identifier of candidate joint exercise
ex_name		Character	Name of the joint exercise
ex_value		Numeric	Value of candidate joint exercise (see [3])
startYear		Numeric	Year the candidate joint exercise begins consuming JETA funds
dataset_id	Foreign	Numeric	Unique identifier of the data set
focus		Character	Level assigned to the exercise focus criterion
gc_interest		Character	Level assigned to the Government of Canada (GC) interest criterion
partnering		Character	Level assigned to the partnering with external actors criterion
rel_ops		Character	Level assigned to the relationship to current operations criterion
hgh_rds_vld		Character	Level assigned to the high readiness validation opportunity criterion
joint_ops_rds		Character	Level assigned to the potential to improve or enhance joint operational readiness criterion
geo_id	Foreign	Numeric	Unique identifier of the geographic region that the candidate joint exercise will be conducted within
conplan_id	Foreign	Numeric	Unique identifier of the CONPLAN that the candidate joint exercise will exercise
format		Character	Format of the joint exercise

Table 2: *Table definition — geoTable.*

Column name	Key	Format	Description
geo_id	Primary	Numeric	Unique identifier of the geographic region
geo_name		Character	Name of the geographic region
dataset_id	Foreign	Numeric	Unique identifier of the data set

Table 3: *Table definition — geoOccurenceTable.*

Column name	Key	Format	Description
geoOccurence_id	Primary	Numeric	Unique identifier of the required occurrence of a joint exercise in a geographic region
centroid_year		Numeric	Fiscal year that is the centroid of a time period in which a joint exercise must occur within the geographic region
geo_id	Foreign	Numeric	Unique identifier of the geographic region
dataset_id	Foreign	Numeric	Unique identifier of the data set

Table 4: *Table definition — conplanTable.*

Column name	Key	Format	Description
conplan_id	Primary	Numeric	Unique identifier of the CONPLAN
dataset_id	Foreign	Numeric	Unique identifier of the data set
conplan_name		Character	Name of the CONPLAN

Table 5: *Table definition — conplanValidateTable.*

Column name	Key	Format	Description
conplanValidate_id	Primary	Numeric	Unique identifier of the required occurrence of a joint exercise to validate a CONPLAN
centroid_year		Numeric	Centroid of a time period in which a joint exercise must occur to provide an opportunity to validate a CONPLAN
conplan_id	Foreign	Numeric	Unique identifier of the CONPLAN
dataset_id	Foreign	Numeric	Unique identifier of the data set

Table 6: *Table definition — activityTable.*

Column name	Key	Format	Description
activity_id	Primary	Numeric	Unique identifier of the activity
activity		Character	Type of activity ('Execution' or 'Planning')
year		Numeric	Fiscal year that the activity occurs
exercise_id	Foreign	Numeric	Unique identifier of the joint exercise
dataset_id	Foreign	Numeric	Unique identifier of the data set

Table 7: *Table definition — costTable.*

Column name	Key	Format	Description
cost_sample_id	Primary	Numeric	Unique identifier of the sample of the activity's cost
cost		Numeric	Cost sample
activity_id	Foreign	Numeric	Unique identifier of the activity
dataset_id	Foreign	Numeric	Unique identifier of the data set

Table 8: *Table definition — taskTable.*

Column name	Key	Format	Description
task_id	Primary	Numeric	Unique identifier of the joint task
task_name		Character	Name of the joint task, including its index (e.g., JT 1.7)
dataset_id	Foreign	Numeric	Unique identifier of the data set

Table 9: *Table definition — taskValidateTable.*

Column name	Key	Format	Description
taskValidate_id	Primary	Numeric	Unique identifier of the requirement to provide an opportunity to validate the joint task
centroid_year		Numeric	Fiscal year that is the centroid of a time period in which a joint exercise must occur to provide an opportunity to validate the joint task
task_id	Foreign	Numeric	Unique identifier of the joint task
dataset_id	Foreign	Numeric	Unique identifier of the data set

Table 10: *Table definition — exerciseToTaskTable.*

Column name	Key	Format	Description
ex_task_id	Primary	Numeric	Unique identifier of the joint task to be conducted within the joint exercise
exercise_id	Foreign	Numeric	Unique identifier of the joint exercise
task_id	Foreign	Numeric	Unique identifier of the joint task
dataset_id	Foreign	Numeric	Unique identifier of the data set

Table 11: *Table definition — datasetTable.*

Column name	Key	Format	Description
dataset_id	Primary	Numeric	Unique identifier of the dataset
seed		Numeric	Seed used to generate the samples of activity costs
timestamp		Numeric	Time the dataset was created

Table 12: *Table definition — resultTable.*

Column name	Key	Format	Description
result_id	Primary	Numeric	Unique identifier of the decision to select or not select a joint exercise
optimal_sel		Numeric	Decision to select or not select a joint exercise; 1 represents select and 0 represents not select
dataset_id	Foreign	Numeric	Unique identifier of the dataset
exercise_id	Foreign	Numeric	Unique identifier of the joint exercise

Table 13: *Table definition — budgetTable.*

Column name	Key	Format	Description
budget_id	Primary	Numeric	Budget for the given year
year		Numeric	Fiscal year to which the budget is associated
budget		Numeric	Budget in dollars

References

[1] Pall, R., Rempel, M., and Roi, M. (2017), Joint exercise portfolio optimization: Problem description, methodology, criteria, and constraints. Presented to Commander CJOC at Future Ops & Plans (5 July 2017).

[2] MacLeod, M. R. and Rempel, M. (2017), Using optimization to support the Joint Exercise Training Account business planning, (DRDC-RDDC-2017-L329) DRDC – Centre for Operational Research and Analysis.

[3] MacLeod, M. R., Rempel, M., and Roi, M. (2018), Joint exercise portfolio optimization: problem definition and formulation, (unpublished draft) DRDC – Centre for Operational Research and Analysis.

List of symbols, abbreviations, and initialisms

CAF Canadian Armed Forces

CJOC Canadian Joint Operations Command

COA course of action

CONPLAN contingency plan

DRENET Defence Research Establisment Network

FY fiscal year

GC Government of Canada

GMPL GNU Mathematical Programming Language

GNU GNU's not Unix

JETA Joint Exercise Training Account

JMET Joint Mission Essential Task

JTA Joint Training Authority

JTAG Joint Training Advisory Group

L1 level 1

ODBC Open Database Connectivity

OR&A Operational Research and Analysis

SQL Structured Query Language

DOCUMENT CONTROL DATA

*Security markings for the title, authors, abstract and keywords must be entered when the document is sensitive

1. ORIGINATOR (Name and address of the organization preparing the document. A DRDC Centre sponsoring a contractor's report, or a tasking agency, is entered in Section 8.)

DRDC – Centre for Operational Research and Analysis
Carling Campus, 60 Moodie Drive, building 7S.2, Kanata, ON K1A 0K2, Canada

2a. SECURITY MARKING (Overall security marking of the document, including supplemental markings if applicable.)

CAN UNCLASSIFIED

2b. CONTROLLED GOODS

NON-CONTROLLED GOODS
DMC A

3. TITLE (The document title and sub-title as indicated on the title page.)

Joint Exercise Training Allocation decision support software: Instruction manual and database design

4. AUTHORS (Last name, followed by initials – ranks, titles, etc. not to be used. Use semi-colon as delimiter)

MacLeod, M. R.; Rempel, M.

5. DATE OF PUBLICATION (Month and year of publication of document.)	6a. NO. OF PAGES (Total pages, including Annexes, excluding DCD, covering and verso pages.)	6b. NO. OF REFS (Total cited in document.)
June 2018	19	3

7. DOCUMENT CATEGORY (e.g., Scientific Report, Contract Report, Scientific Letter)

Reference Document

8. SPONSORING CENTRE (The name and address of the department project or laboratory sponsoring the research and development.)

DRDC – Centre for Operational Research and Analysis
Carling Campus, 60 Moodie Drive, building 7S.2, Kanata, ON K1A 0K2, Canada

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)	9b. CONTRACT NO. (If appropriate, the applicable contract number under which the document was written.)
06ac	

10a. DRDC DOCUMENT NUMBER	10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)
DRDC-RDDC-2018-D054	

11a. FUTURE DISTRIBUTION WITHIN CANADA (Approval for further dissemination of the document. Security classification must also be considered.)

Public release

11b. FUTURE DISTRIBUTION OUTSIDE CANADA (Approval for further dissemination of the document. Security classification must also be considered.)

Public release

12. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Use semi-colon as a delimiter.)

Data Analytics; Visual Analytics; Budgeting; Decision Support; Joint Training

13. ABSTRACT/RÉSUMÉ (When available in the document, the French version of the abstract must be included here.)

CJOC JTA staff requested that the CJOC OR&A team design and implement a transparent, repeatable, evidence-based approach to support the development and refinement of a five-year rolling exercise program consistent with government policy and force posture direction. This reference document describes elements of the technical design and practical use of the decision support software CJOC OR&A built to respond to this problem statement.

Le personnel du responsable de l'instruction interarmées (RII) du Commandement des opérations interarmées du Canada (COIC) a demandé à l'équipe d'analyse et de recherche opérationnelle (ARO) de concevoir et de mettre en œuvre une méthode transparente, reproductible et fondée sur les faits pour appuyer l'élaboration et la mise au point d'un programme d'exercice quinquennal conforme à la politique gouvernementale et à l'orientation en matière de posture des forces. Le présent document de référence décrit les éléments de la conception technique et de l'utilisation pratique du logiciel d'aide à la décision que l'équipe d'ARO du COIC a créé pour répondre à cet énoncé de problème