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

DRDC | RDDC
technologysciencetechnologie

GALO—Genetic Algorithm for Layout Optimization

A technical documentation for modellers and developers

Jonathan Lin
University of Waterloo

Wenbi Wang, PhD
DRDC – Toronto Research Centre

Defence Research and Development Canada
Reference Document
DRDC-RDDC-2018-D139
January 2019

Canada

IMPORTANT INFORMATIVE STATEMENTS

This document was reviewed for Controlled Goods by Defence Research and Development Canada (DRDC) using the Schedule to the *Defence Production Act*.

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.

Abstract

GALO (Genetic Algorithm for Layout Optimization) is a desktop application tool that uses a genetic algorithm to determine the optimal layout of a group of operators in a workplace. In GALO, a workplace is modelled as a rectangular grid that is comprised of many cells. Each cell is defined as an open workspace that can be assigned to an operator, a barrier that functions as a spatial obstruction, or an unassignable space that could not be designated as a workspace. An optimal layout is defined as the one that best supports inter-operator communication (or interaction). Communication requirements are modelled as weighted directional connections. For each workplace model, after inter-operator communication requirements are specified, GALO has the ability to generate the optimal layout configuration based on a genetic algorithm. This Reference Document provides an instruction guide for future users of the tool, and a technical documentation of the program for future software developers.

Significance to defence and security

Command and Control spaces like an Operations room are complex work environments and critical facilities for a military operation. Their physical layout is an important design consideration since the position and orientation of operators affect team interaction and therefore operational effectiveness. GALO provides an algorithmic solution to identify optimal workplace layout arrangements.

Résumé

GALO (*Genetic Algorithm for Layout Optimization*) est une application bureautique qui utilise un algorithme génétique pour déterminer l'aménagement optimal du lieu de travail d'un groupe d'opérateurs. GALO permet de modéliser un lieu de travail sous la forme d'une grille rectangulaire constituée de nombreuses cellules. Chaque cellule représente un espace de travail ouvert pouvant être attribué à un opérateur, une barrière faisant office d'obstacle spatial ou bien un espace ne pouvant pas être attribué ni désigné comme zone de travail. On définit l'aménagement optimal comme étant celui qui favorise le plus la communication (ou les interactions) entre les opérateurs. Les exigences de communication sont modélisées sous la forme de liaisons directionnelles pondérées. Une fois qu'on a précisé les exigences de communication entre les opérateurs pour chaque modèle de lieu de travail, GALO a la capacité de déterminer l'aménagement optimal en fonction d'un algorithme génétique. Le présent document de référence renferme un guide d'instructions pour les futurs utilisateurs de l'outil, ainsi que des renseignements techniques sur le programme à l'intention des futurs développeurs de logiciels.

Importance pour la défense et la sécurité

Les zones de commandement et contrôle, comme la salle des opérations, constituent des environnements de travail complexes qui comportent des installations indispensables aux opérations militaires. L'aménagement physique des lieux est un aspect important de la conception d'un espace de travail, car la position et l'orientation des opérateurs ont une incidence sur les interactions entre les membres de l'équipe et par conséquent sur l'efficacité opérationnelle. GALO constitue une solution algorithmique pour déterminer l'aménagement optimal d'un lieu de travail.

Table of contents

List of figures

1 Introduction

A workplace layout optimisation tool entitled GALO (Genetic Algorithm for Layout Optimization) has been developed by the Defence Research and Development Canada (DRDC), under project 01ab *Royal Canadian Navy (RCN) crewing and human factors*. The tool is specifically created to design and evaluate the floor plans of collaborative work environments such as a military command centre or an operations room. Specifically, GALO represents workspace at a highly abstract level and uses a genetic algorithm to automatically search for desirable placement of operators (and their workstations) in a finite set of optional spaces. This tool was further expanded from a prototype previously developed and validated in-house [1, 2].

This Reference Document provides a technical record of GALO. It was written with two target audience in mind. Section 2 provides instructions for future users of this tool, that is, modellers or analysts who use the tool for layout design and analysis. Detailed step-by-step descriptions are provided to explain the typical processes used to create a GALO model and identify layout solutions using the genetic algorithm. In the following Section 3, a technical documentation of the tool was composed to describe the underwiring of GALO from the software programming point of view. Such information will assist future developers of this tool when further expansion of its capability is required. Information provided in this document is based on GALO Version 4.4.

2 A step-by-step instruction on how to use GALO

Section 2 describes the typical process to use GALO for layout analysis. It is written for future modellers of this tool.

2.1 Software installation

GALO is a Microsoft Windows application. Its operation requires the Vista or a later version of the Windows operating system, and Microsoft .NET Framework 4.5 or later. The software can be installed on a computer in two ways: automatically using an installation wizard or manually.

If the user prefers an automatic installation, double clicking the file "GALO 4.4.msi" will open the installation wizard (as shown in Figure 1).

Figure 1: *GALO setup wizard dialog.*

The user can step through the wizard's interface by clicking the "Next" button. Installation options like GALO's file directory can be specified during this process. For the current version of the program, it is strongly recommended to use the default directory suggested by the wizard. If the user's account does not have administrator privilege, an administrator password will be asked during the installation process.

Once the installation is successfully completed, a dialog will pop up, notifying the user that GALO 4.4 has been successfully installed (as shown in Figure 2). A shortcut to GALO will appear on the user's desktop.

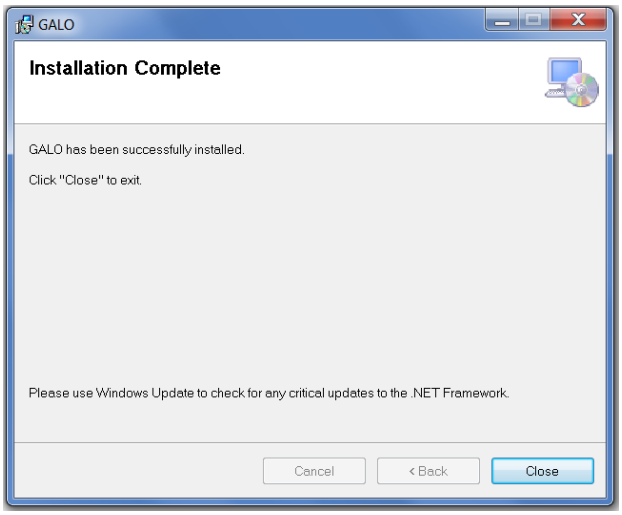

Figure 2: *GALO setup wizard successful installation dialog.*

Alternatively for a manual installation, the user should create a folder with an appropriate name in the directory of choice. The user needs to copy over the following three files: "GALO.exe," "GA_C.exe," "dijkstra.exe." All three files must be in the same folder for GALO to function. GALO can then be run by double clicking "GALO.exe." It is recommended that the user create a shortcut on their desktop for "GALO.exe" to facilitate easy access to the program.

2.2 Overview of user interface and GALO modelling process

GALO has a simple Graphical User Interface (GUI), as shown in Figure 3. It is comprised of three panes: a navigation pane on the left, the main user interface on the top right, and two control buttons at the bottom right for stepping through the modelling process.

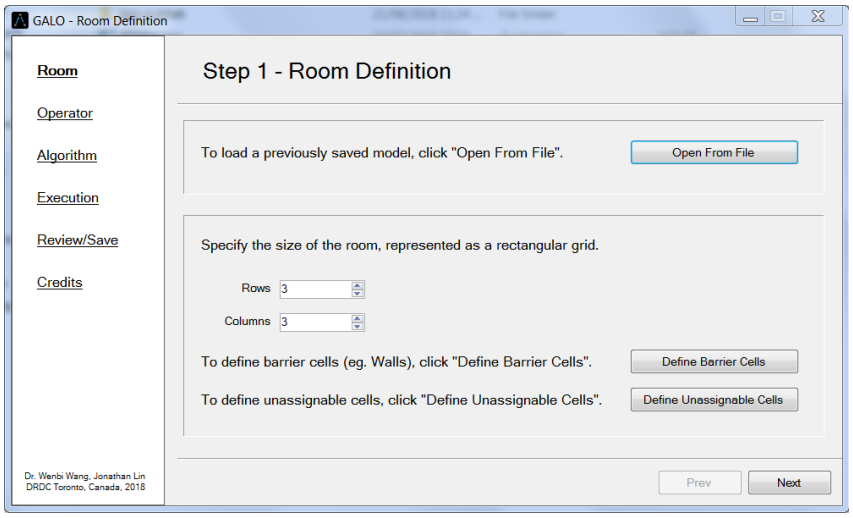

Figure 3: *GUI for room definition.*

Five steps are required to conduct a layout analysis in GALO, namely room definition, operator definition, algorithm parameter setup, model execution, and results review. In the rest of the section, we will explain each step in more detail.

2.3 Room definition

The first step to create a GALO model is to define the room, hereby referred to as the workplace. The objective of GALO is to optimize the placement of operators in a workplace. A screenshot of the GUI for this step is shown in Figure 3.

If there exists a previously saved GALO model, it is possible to load it in this step by clicking on the button "Open From File." A dialog will open where the user can navigate to and select their previously saved model, as shown in Figure 4.

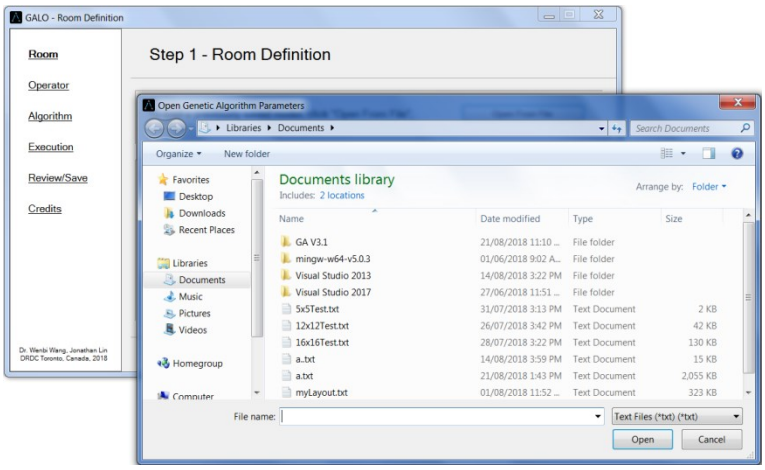

Figure 4: *Dialog for loading a previously saved GALO model.*

For a new model, the user first needs to define the size of the room by specifying the number of rows and columns in this workplace. A workplace is represented abstractly as an $m \times n$ grid (e.g., as illustrated in Figure 5). Each grid location, hereby referred to as a cell, can be defined as an open cell where an operator can be placed, a barrier cell which causes spatial obstruction, or an unassignable cell where an operator cannot be placed but which does not cause any obstruction. Common architectural features of a room can be represented by the latter two types of cells, such as walls or walkways.

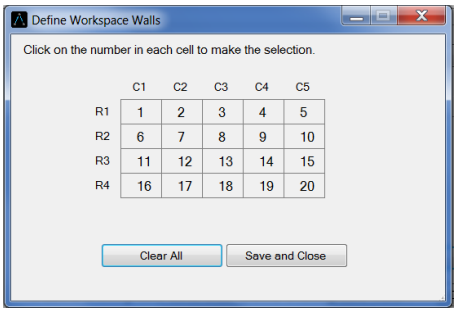

Figure 5: *A grid-like workspace model.*

Barriers and unassignable cells need to be defined explicitly in GALO by clicking “Define Barrier Cells,” and “Define Unassignable Cells” respectively. Clicking either of these buttons will open a dialog showing the labelled workplace. If the barrier cell dialog is open, clicking a cell’s numerical label will define that cell to be a barrier cell. The same process holds for unassignable cells. Barrier cells are represented with black contours, while unassignable cells are represented with red contours, as illustrated in Figure 6. Clicking “Clear All” in the barrier cell dialog will revert all barrier cells to open cells, and likewise for unassignable cells.

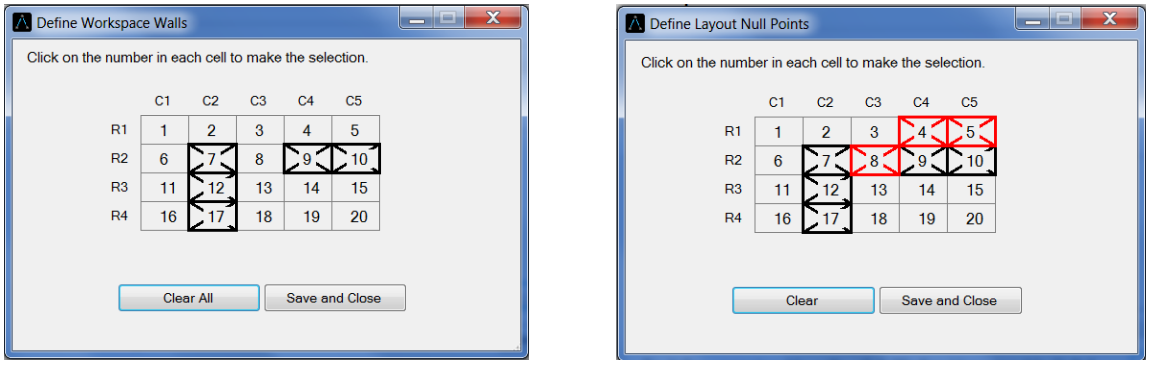

Figure 6: *Barrier cells and unassignable cells.*

Once Step 1 is completed, clicking “Next” or the appropriate link on the side panel will move to the next step.

2.4 Operators and communication requirements

The second step in GALO modelling is to define the number of operators and their communication requirements.

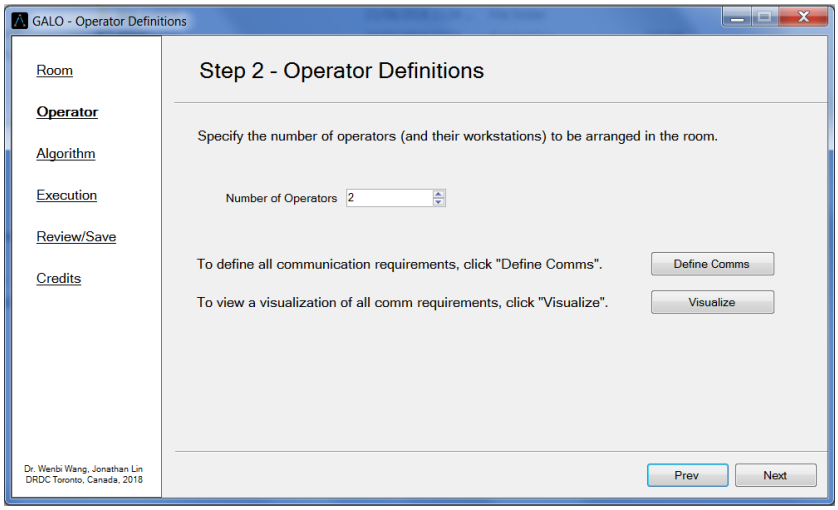

Figure 7: *GUI for room definition.*

GALO represents workers in a workplace as operators, which can be placed in open cells. As shown in Figure 7, the user needs to enter the number of operators into the box labelled “Number of Operators.”

The number of operators must be at least one and is capped at the number of open cells in the workplace previously defined in Step 1.

An example of a team of operators and their communication requirements is shown in Figure 8.

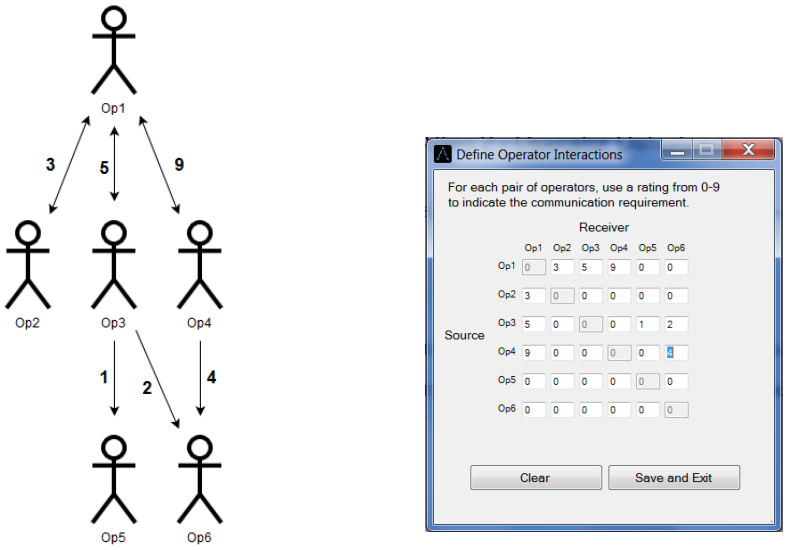

Figure 8: *Operator communication requirements example.*

In a GALO model, inter-operator communication requirements are quantified as a set of weighted directional connections. Specifically, each operator is treated as both a source and a receiver for information exchange. Between each pair of operators, communication requirements can be represented by using a weight between zero (0) and nine (9). A weight of zero (0) indicates no information exchange between the pair. A non-zero weight indicates such requirement exists, and a larger weight represents a higher priority for communication. On the GUI, clicking on "Define Comms" brings up a matrix where communication requirements can be defined. One example is shown in Figure 8. In general, the box at the ith row and jth column represents the weight of the connection from operator i (as the information source) to operator j (as the information receiver). Clicking the "Clear" button on this interface will reset all weights to 0.

An implicit assumption in GALO is that each cell can accommodate a single operator. However, in practice, it is not uncommon that some operators may require a larger floor footprint than others, because for example they have a larger workstation. There are different ways to model such requirements in GALO. One solution is to use a hypothetical "super-operator" concept for representing those who require a larger area than one cell. In GALO, a super-operator is comprised of a group of operators that are interconnected with links of weight 9. When the genetic algorithm is applied, all components of a super-operator will likely be placed together in close proximity. Figures 9 and 10 describe the concept by showing two examples of super-operators with space requirement of 2 and 4 regular cells.

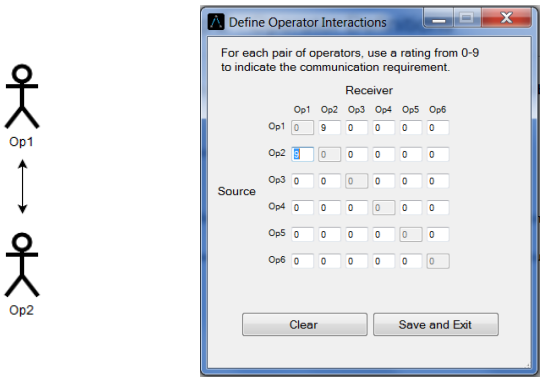

Figure 9: 2 × 1 *superoperator example.*

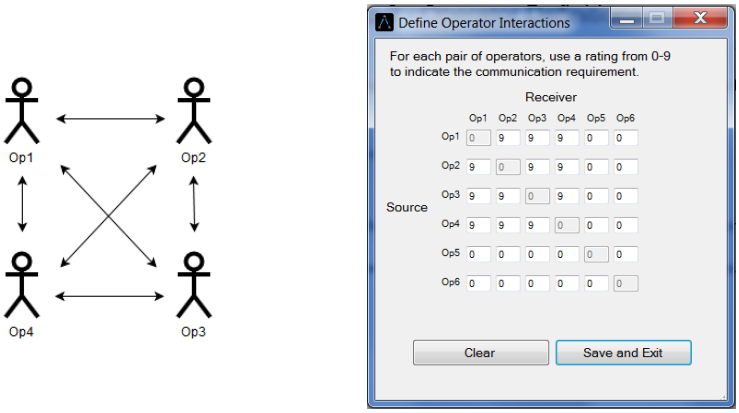

Figure 10: 2 × 2 *superoperator example.*

After inter-operator communications have been defined, clicking "Save and Exit" to close the dialog interface. If the user hopes to review all inputs, the program provides a graphical visualization function. Clicking the "Visualize" button will bring up a circular arrangement of all operators with all communication requirements represented as directional links (as illustrated in Figure 11).

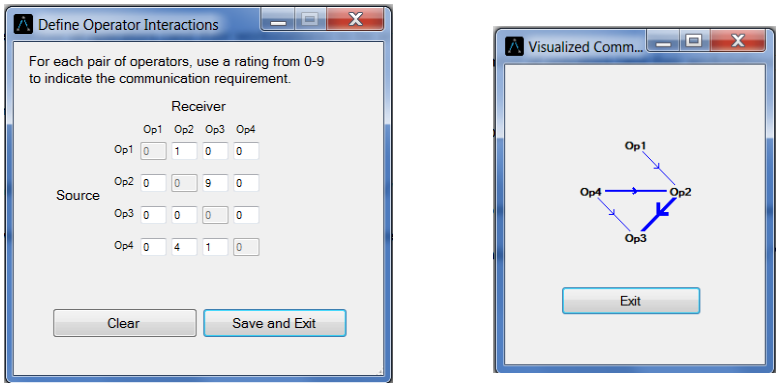

Figure 11: *Visualization of communication requirements.*

Once Step 2 is completed, clicking "Next" or the appropriate link on the side panel will move to the next step.

2.5 Simulation parameters

The third step in GALO modelling is to specify the four simulation parameters for the genetic algorithm: population size, probability of swap, maximum number of generations, and number of independent simulation runs, as shown in Figure 12.

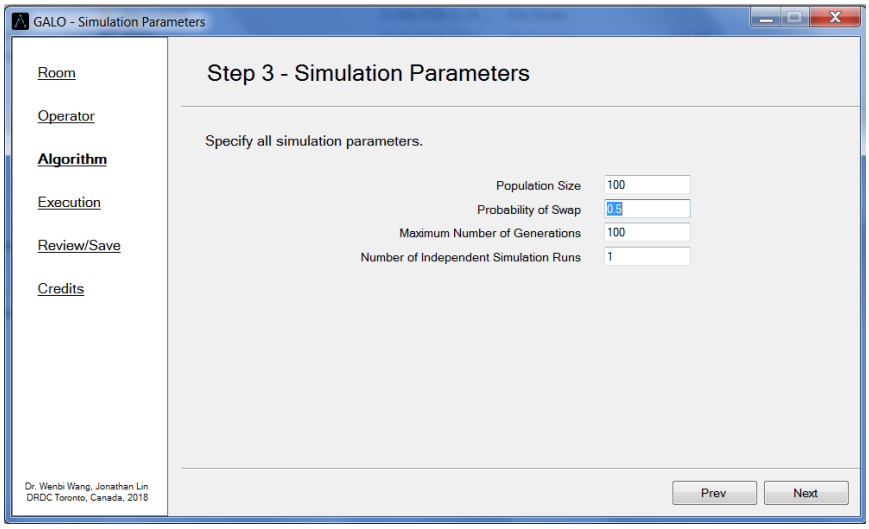

Figure 12: *GUI for simulation parameter setup.*

Population size is the number of candidate solutions that are generated by the genetic algorithm. Having a higher population size increases the probability of an optimal solution being found, at the cost of a longer simulation run time. The population size must be at least 1.

Probability of swap is the likelihood that a solution will undergo a genetic operation (swap) during the simulated evolution. A larger value of this parameter tends to increase the algorithm's search speed in the solution space, which improves the chance of discovering an optimal solution. However, such a setting has a negative consequence as well since it also increases the chance of destroying a quality solution.

Maximum number of generations controls the time frame of the simulated evolution. Its value must be at least 1. In GALO, it serves as a stop-rule for terminating the algorithm's execution. A large parameter setting increases the number of possible solutions explored, which improves the chance of discovering an optimal solution, at the cost of a longer simulation run time.

Number of independent simulation runs is the number of times the simulation will run with the three parameters above. With more simulation runs, there is a higher chance of discovering alternative optimal solutions, at the cost of a longer simulation run time.

For an in-depth explanation of these parameters, readers are referred to [1].

Once Step 3 is completed, clicking "Next" or the appropriate link on the side panel will move to the next step.

2.6 Simulation execution

The fourth step of GALO modelling is to review all user specified parameters and run the simulation.

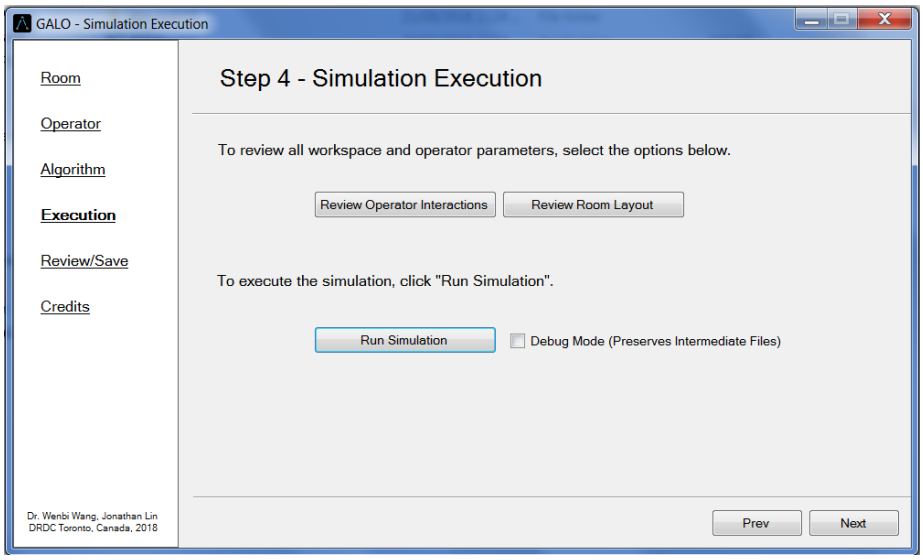

Figure 13: *GUI for simulation execution.*

The user can click “Review Operator Interactions” and “Review Room Layout” to examine model parameter setup, as shown in Figure 13.

Once satisfied, the user can click “Run Simulation” to begin the simulation. Selecting “Debug Mode” before running the simulation will preserve all intermediate text files used by the program. When the simulation begins, several dialogs and a progress bar will appear.

After the simulation is complete, an output window will be displayed with a graphical representation of the most optimal solution generated, as shown in Figure 14. The optimal solution generated by each independent simulation run is referred to as a “winner,” and the most optimal solution overall is referred to as a “true winner.” If more than one true winner is generated, the user can browse through them using the “Prev” and “Next” buttons. The true winner fitness score, number of unique true winners, number of suboptimal winners, number of simulations that produced a winner, and the percentage of simulations that produced true winners are also shown.

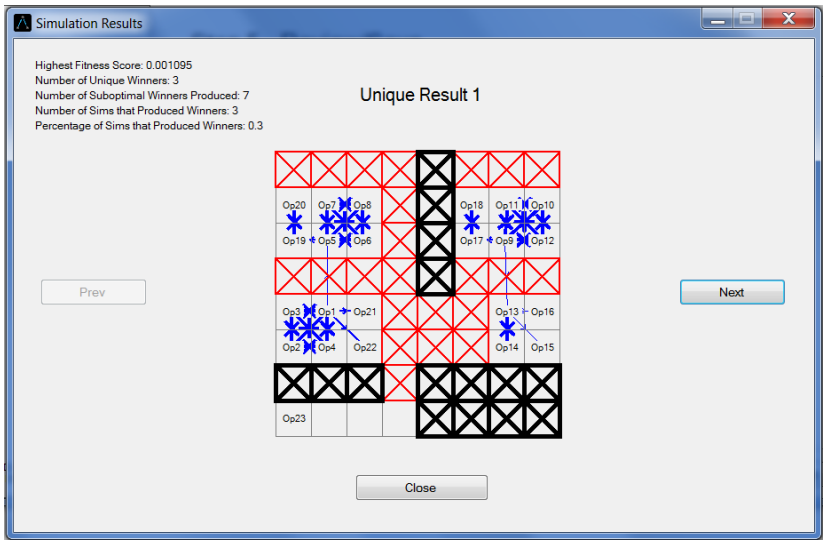

Figure 14: *Simulation output dialog.*

When the simulation is complete and the output window is closed, Step 5 of GALO will automatically be shown. Alternatively, before running the simulation, the user can click "Next" or click the appropriate link on the side panel to move to the next step.

2.7 Viewing results and saving custom models

The fifth and final step of GALO modelling is to review the simulation results and diagnostic information.

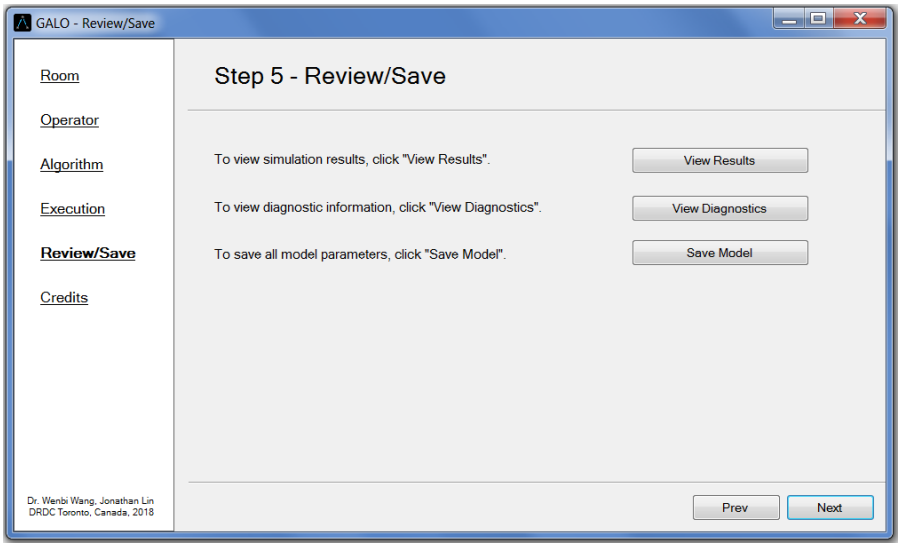

Figure 15: *GUI for reviewing and saving results.*

If a simulation has been completed since opening GALO, clicking "View Results" will bring up the output window described above, with the results of the most recent simulation.

Clicking "View Diagnostics" will bring up a chart showing the progression of the best and average solution's fitness over all generations, as well as the true winner fitness score (see Figure 16). If more than one unique winner is generated, the user can review each winner's diagnostic information using the "Prev" and "Next" buttons.

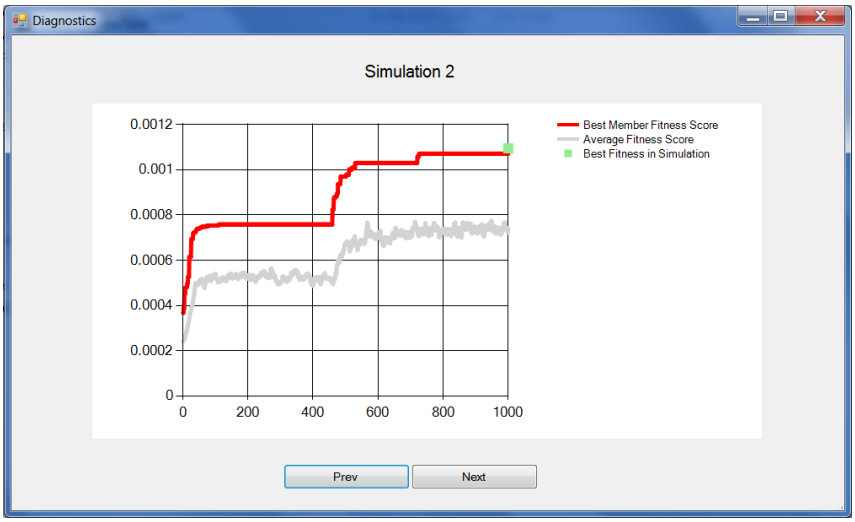

Figure 16: *Diagnostic dialog.*

In this step, the user can also save the current model that includes both workspace and operator definition. Clicking "Save Model" opens a dialog for the user to select a location and name for the model file (see Figure 17). Once created, the model can be loaded into GALO again in the future for further analysis.

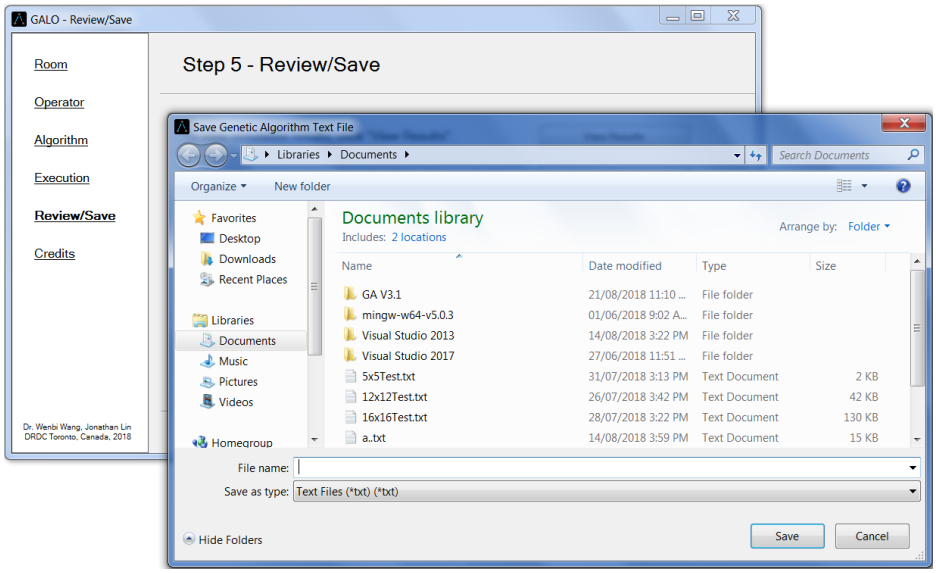

Figure 17: *Save model dialog.*

Step 5 is the last step of a typical GALO analysis. Clicking "Next" on the GUI (or the appropriate link on the side panel) will display the credits, which shows developer names and contact information, as shown in Figure 18.

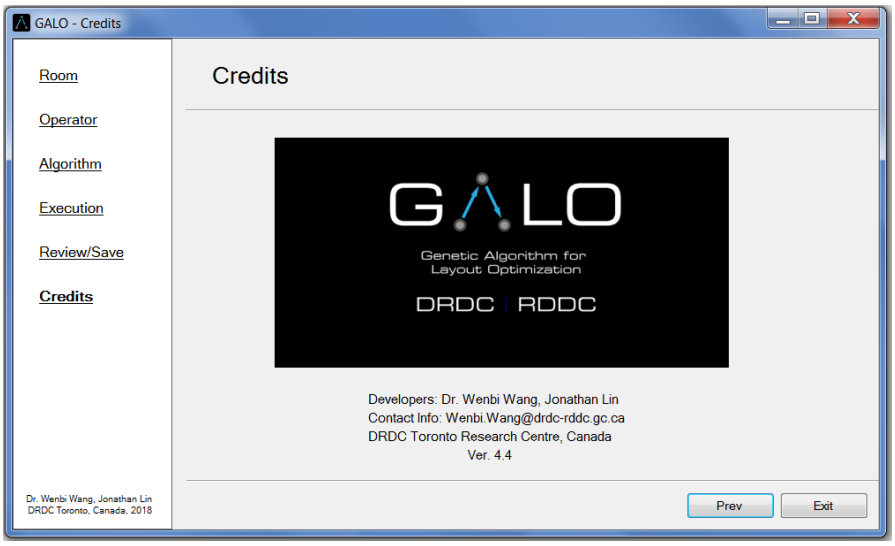

Figure 18: *GUI displaying credits and contact information.*

3 A technical documentation of the GALO program

Section 3 explains the technical aspect of the GALO program. It is written as a technical documentation for future software developers.

3.1 Problem abstraction and simulation theory

The main utility of GALO is to optimize the spatial arrangement of operators in a two dimensional (2D) workspace. However, internally the genetic algorithm optimizes the placement of n number of operators on a 1D line with n slots. Thus, the real life workspace problem is abstractly represented in two models, one which can be defined by the user with the GALO GUI, and the other that is interpreted by the genetic algorithm.

Figure 19: *Abstract representations of real life workspaces.*

The GALO model represents a real life workspace as an $m \times n$ rectangular grid, with each cell being either an open cell, a barrier, or an unassignable cell. The GALO model also represents communication requirements as weighted directional connections between operators. As shown in Figure 19, this level of abstraction is used in the GUI, and serves as an intermediate between real life workspaces and the model used by the genetic algorithm.

The genetic algorithm model represents a workspace as a 1D line with n slots and n operators, where n is the number of open cells in the GALO model. Unlike the GALO model, the genetic algorithm model places an operator in every slot, and represents empty space as an operator without any connections. Such differences are illustrated by an example in Figure 20.

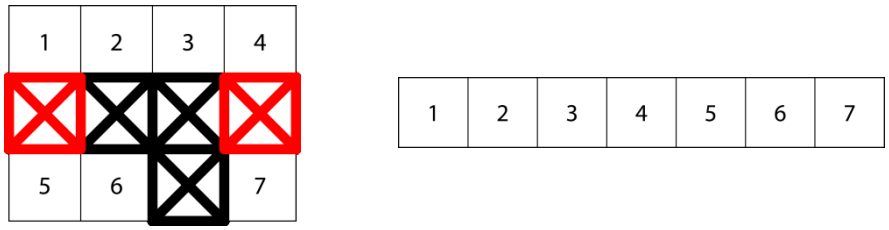

Figure 20: *GALO model and genetic algorithm model of a sample workspace.*

In the GALO model, every pair of slots in the line has a numerical weight from 0 to 999 stored in an adjacency matrix, which represents the distance from those two cells. Similarly, every pair of operators has a numerical weight from 0 to 9 stored in the weight matrix, which represents the relative importance of that connection, with 0 representing no connection. We will elaborate on these two importance matrices in the following subsections.

3.1.1 Adjacency matrix

The effectiveness of the placement of two operators in a workplace depends on the shortest path between them.

A path in the GALO model is made of multiple adjacent or diagonal "hops," as shown in Figure 21. An adjacent hop is defined to have a weight of 2, while a diagonal hop is defined to have a weight of 3, so as to approximate the ratio of the adjacent Euclidian distance to the diagonal Euclidian distance.

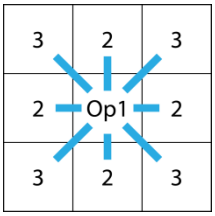

Figure 21: *All possible hops with respective weights.*

The shortest path distance between two cells is defined as the length of the shortest path from one cell to the other. One example is shown in Figure 22, where the shortest path distance between two cells (i.e., cell 6 and cell 7) is 13.

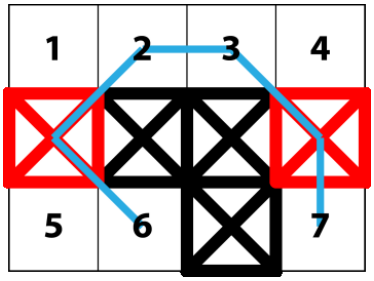

Figure 22: *An example of the shortest path between cell 6 and cell 7.*

The shortest path distances of all pairs of open cells are stored in an adjacency matrix. An adjacency matrix is a square matrix of size $n \times n$, where C_i is the ith cell, a_{ij} is the shortest path from cell i to cell j, and n is the number of open cells in the GALO model.

$$\begin{array}{c|ccccc} & \boldsymbol{Cell_1} & \boldsymbol{Cell_2} & \boldsymbol{Cell_3} & \cdots & \boldsymbol{Cell_n} \\ \boldsymbol{Cell_1} & a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ \boldsymbol{Cell_2} & a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \boldsymbol{Cell_3} & a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{Cell_n} & a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{array} \quad (1)$$

If i and j are the same, the shortest path distance is 0. If cells i and j have no valid path between them, the shortest path distance is 999. From these properties, an adjacency matrix is always symmetric.

3.1.2 Communication weight matrix

The importance of a connection (i.e., a communication link) between two operators is represented as a weight from 0 to 9, where 0 represents no connection and 9 represents a most critical connection. The weight of every connection between a pair of operators is stored in a communication weight matrix.

This matrix is a square matrix of size $n \times n$, where n is the number of operators. As the genetic algorithm model places an operator in every open cell, for any workspace, the size of its weight matrix and adjacency matrix is the same.

$$\boldsymbol{W} = \begin{bmatrix} & \boldsymbol{Cell_1} & \boldsymbol{Cell_2} & \boldsymbol{Cell_3} & \cdots & \boldsymbol{Cell_n} \\ \boldsymbol{Cell_1} & w_{11} & w_{12} & w_{13} & \cdots & w_{1n} \\ \boldsymbol{Cell_2} & w_{21} & w_{22} & w_{23} & \cdots & w_{2n} \\ \boldsymbol{Cell_3} & w_{31} & w_{32} & w_{33} & \cdots & w_{3n} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{Cell_n} & w_{n1} & w_{n2} & w_{n3} & \cdots & w_{nn} \end{bmatrix} \tag{2}$$

w_{ij} represents the weight (0–9) of a connection between the ith and jth operators. If there is no connection between these two operators, w_{ij} is 0.

3.1.3 Simulation theory

The process from converting a real life workspace to a simulated result is illustrated in Figure 23 and described as follows:

The user represents a real life workspace as a GALO model using the GUI. This includes defining the room dimensions, selecting barrier and unassignable cells, specifying the number of operators, and inter-operator communication requirements. When the simulation begins, this GALO model is converted to inputs for a genetic algorithm which internally represents the workspace as a one-dimensional, single row of spaces (with $m \times n$ slots), an adjacency matrix to represent physical proximity between each pair of cells/slots, and a weight matrix for inter-operator communication requirements.

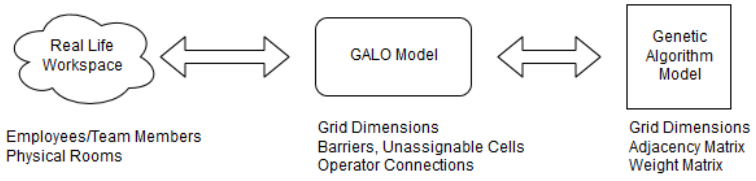

Figure 23: *Levels of abstraction of real life workspaces.*

The genetic algorithm uses the above inputs to generate optimal layouts. For each simulation run, a set of operator-cell arrangement is randomly created at the beginning. The effectiveness of each arrangement is quantified as its "fitness" using a fitness function.

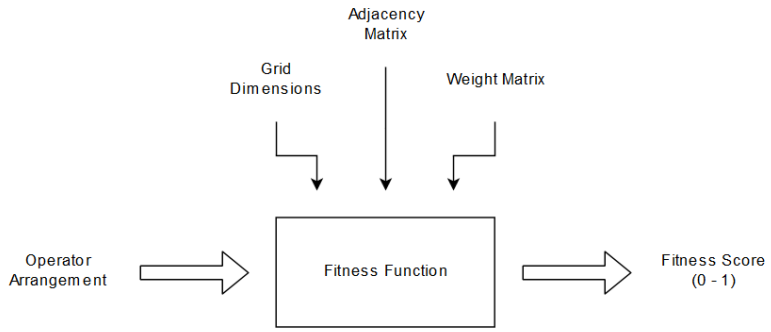

Figure 24: *Fitness function overview.*

Over successive generations, arrangements undergo random mutations, while high fitness arrangements are preserved.

The fitness of an arrangement (denoted as F) is the inverse of the sum of the fitness of every operator pair (denoted as f). The fitness of an operator pair is the shortest path distance between their positions multiplied by their connection weight. If an operator pair has no connection, its connection weight is 0 and thus its fitness f does not affect the overall fitness score F.

$$F = \frac{1}{\sum_{i=1}^{n}\sum_{j=1}^{n} f(i,j)} \tag{3}$$

$$f(i,j) = a_{p(i),p(j)} \cdot w_{i,j} \tag{4}$$

Arrangements with operators placed in closer proximity minimize the shortest path distances between connected operators, maximizing the arrangement's fitness. Similarly, arrangements with higher weight connected operators in closer proximity will have higher fitness scores.

Every independent simulation run produces a winning operator arrangement with its fitness score. Multiple independent simulation runs may produce identical winning arrangements. The arrangements with the highest fitness scores are found and converted back to GALO models before being displayed graphically to the user.

3.2 Program architecture

GALO consists of three executable programs: GALO.exe which is the main program, Dijkstra.exe and GA_C.exe which are two subroutines called upon by GALO.exe. GALO.exe is a windows forms application written in C# using MS Visual Studio 2017. The two subroutines are compiled from source code written in C. At runtime, these three executables communicate with each other using temporary text files, as shown below in Figure 25.

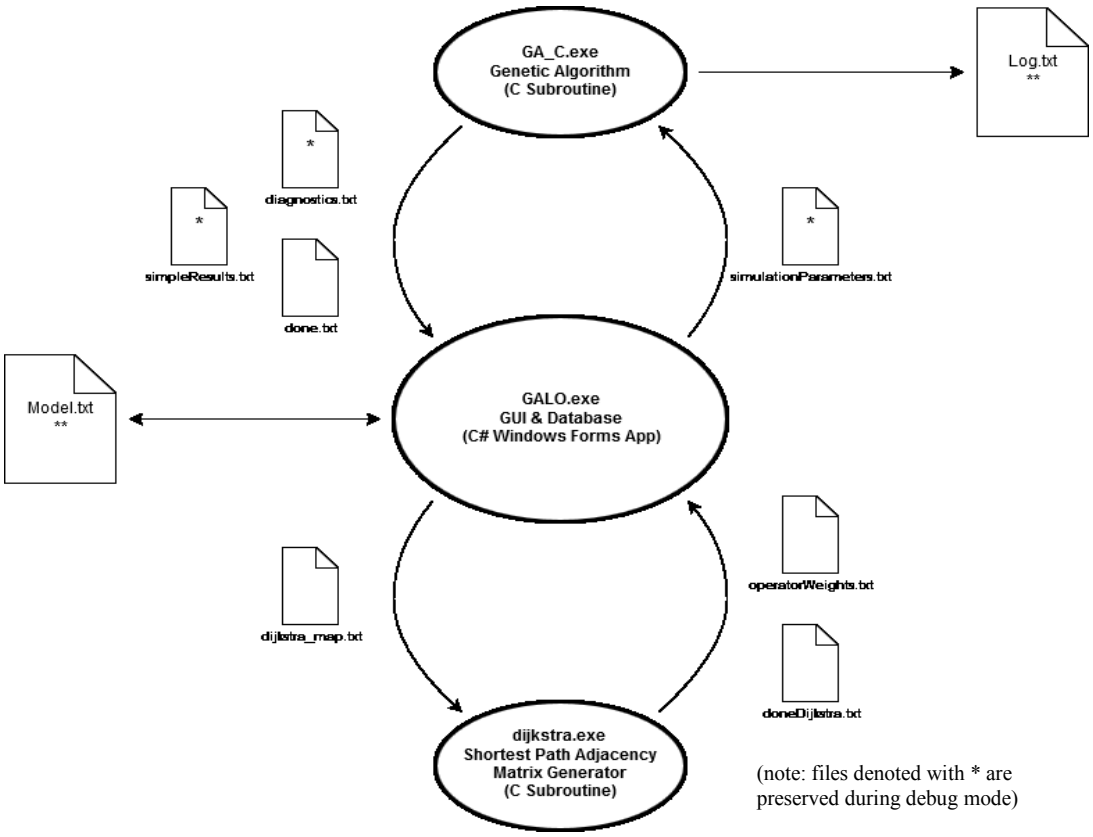

Figure 25: *GALO program architecture overview.*

GALO.exe serves as the main entry point to the program with GUIs and functionalities for entering and storing all simulation parameters. GALO.exe also converts user specified models into genetic algorithm models for GA_C.exe to read. The user has the option to load a pre-saved model, shown as Model.txt. Once a simulation is executed, the dijkstra_map.txt file is created to store the room dimensions, barrier cells, and unassignable cells.

The dijkstra.exe subroutine is then executed, which reads the data from dijkstra_map.txt and generates a weight matrix in a new file called operatorWeights.txt. Upon completion, the file dijkstra_map.txt will be deleted and another file doneDijkstra.txt will be created.

Once GALO.exe detects the existence of doneDijkstra.txt, it then reads the weight matrix from operatorWeights.txt and logs a complete set of parameters for GA_C.exe in a file called simulationParameters.txt. The temporary file operatorWeights.txt is removed at the end of the process. GALO.exe then removes the temporary file operatorWeights.txt and loads the subroutine GA_C.exe.

GA_C.exe reads the complete set of parameters from simulationParameters.txt, subsequently deleting this file, and executes the simulation. Once the simulation is complete, a detailed log is created with a file name and at the location specified by the user. Fitness scores and operator positions for each simulation run are written in simpleResults.txt, diagnostic information for each run is written in diagnostics.txt. At its end, GA_C.exe will produce an empty file named done.txt to indicate its completion.

Upon detection of done.txt, GALO.exe reads the simulation results and diagnostic info before deleting simpleResults.txt, diagnostics.txt, and done.txt. These results are displayed in the GUI. At this point in the main program, the user has the option to save the model for future use.

3.2.1 Main program (GALO.exe)

GALO.exe is a WinForms application written in C# using the .NET Framework 4.7.2. GALO.exe serves as a user interface and a database for all parameters used by the two subroutines.

GALO.exe stores all data that is entered by the user, sent out to the subroutines, and returned by the subroutines.

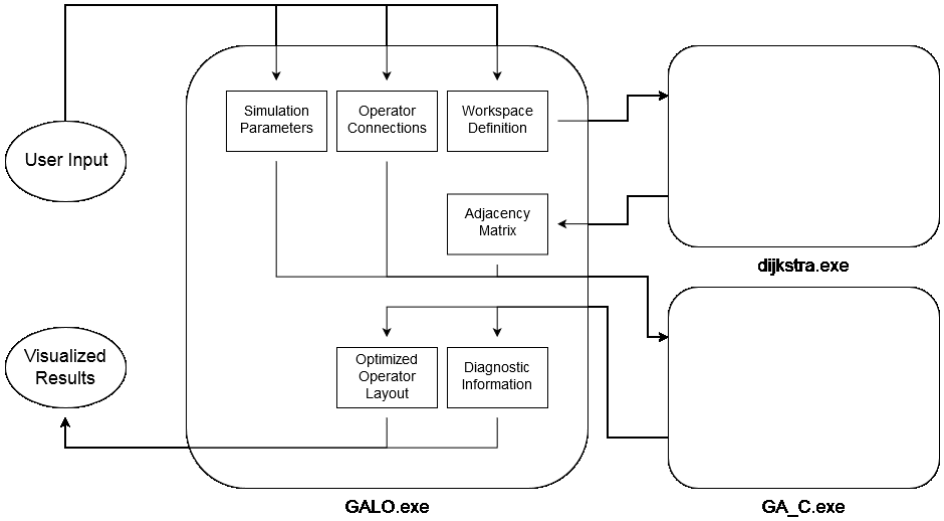

Figure 26: *GALO.exe database overview.*

Figure 26 provides an overview of the program's data structure. As the user enters their workplace definition, operator connections, and simulation parameters through the GUI or from a previously saved model, these three sets of information are updated in GALO.exe. When the simulation runs, the adjacency matrix information is updated in GALO.exe from the results of dijkstra.exe. GA_C.exe then takes the adjacency matrix, operator connections, and simulation parameters and creates a set of optimized operator layouts and diagnostic information, which is read and updated by GALO.exe. This information is visualized to the user.

3.2.2 Dijkstra subroutine (dijkstra.exe)

As shown in Figure 27, the subroutine dijkstra.exe takes a GALO model workplace definition and outputs an adjacency matrix. As its name suggests, dijkstra.exe uses the Dijkstra algorithm to determine the shortest path distance between every pair of open cells.

Figure 27: *Input and output of Dijkstra subroutine.*

Dijkstra's algorithm finds the shortest path from a starting node on a graph to every other node, given the weight of each connection. Workspaces in the GALO model are represented as an $m \times n$ grid with a list of barrier cells and unassignable cells. Thus, dijkstra.exe first converts the GALO model parameters in dijkstra_map.txt to a graph. Graphs in dijkstra.exe are also stored as adjacency matrices, referred to as map adjacency matrices.

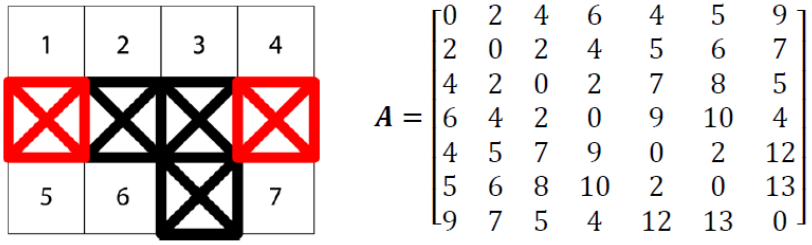

Figure 28: *Example of workspace and map adjacency matrix.*

Once a map adjacency matrix is generated, the adjacency matrix can be created using Dijkstra's algorithm. The adjacency matrix is an $n \times n$ square matrix, where n is the number of open cells.

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots \\ a_{21} & a_{22} & a_{23} & \cdots \\ a_{31} & a_{32} & a_{33} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{5}$$

As Dijkstra's algorithm finds the shortest path from a starting node to every other node, running Dijkstra's algorithm on an open cell i fills out row i in the adjacency matrix. The entire adjacency matrix is filled by running Dijkstra's algorithm on each open cell and filling in the appropriate row in the adjacency matrix. An example of workplace definition and its corresponding adjacency matrix is shown in Figure 28.

Once the adjacency matrix is complete, it is written to the file operatorWeights.txt as input data for GA_C.exe.

3.2.3 Genetic algorithm subroutine (GA_C.exe)

GA_C.exe takes a workplace definition, adjacency matrix, and set of simulation parameters and output a set of winner solutions along with their fitness scores. The number of winner solutions generated is specified by the number of independent simulation runs in the set of simulation parameters. All parameters and results are stored in text files.

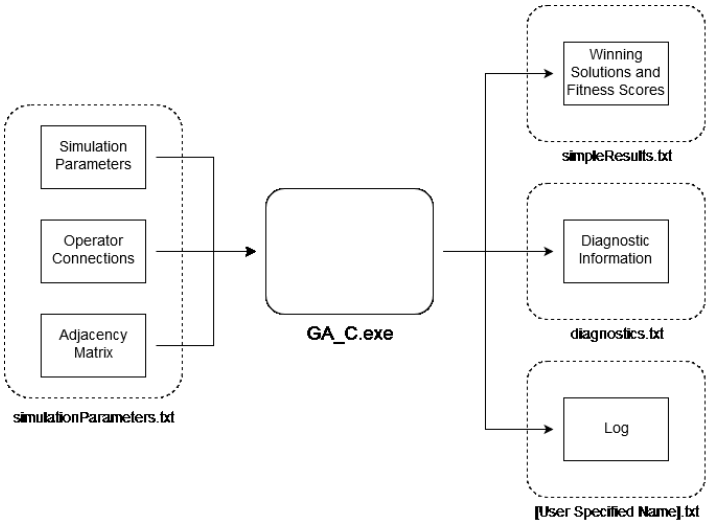

Figure 29: *Input and output of GA_C.exe subroutine.*

Outputs from the algorithm are stored in simpleResults.txt and diagnostics.txt. They are read by GALO.exe and graphically presented to the user through the main program GUI. The structure of GA_C.exe is shown in Figure 29.

4 Summary

GALO is a desktop software tool for workplace layout optimisation. It was developed based a prototype program originally written in C. GALO expands this prototype by enhancing its functionalities in four main areas: provision of a user-friendly GUI for data entry as well as result visualization, introduction of the concept of barrier and unassignable cells in workspace abstraction, prioritization of operator communication links, and the inclusion of an algorithmic solution to calculate the shortest path between two cells. With these features, GALO can be used to model a wide variety of work environments and address the allocation of operators (and their workstations) into optional workspaces.

From the perspective of software architecture, GALO is comprised of three components: the main program for rendering user interfaces (i.e., the GUI), a subroutine for pathfinding, and a subroutine that runs the genetic optimization algorithm. The GUI (encapsulated in GALO.exe) is a Windows Form App using Microsoft .NET Framework 4.5, and the two subroutines are executable files compiled from source code written in C. These three components communicate with each other (for example to sync up on execution) and store output data using a number of text files.

Barrier cells, unassignable cells, and weighted operator connections are new features that were introduced into GALO to improve the software's flexibility for modelling workspaces. By representing a workplace as a two dimensional grid, each cell within this grid can be defined as either a barrier cell (e.g., to represent walls or pillars), an unassignable cell (e.g., to represent pathways), or an open cell where operators can be allocated. Weighted operator connections allow a modeller to prioritize inter-operator communication requirements. With the introduction of weighted operator connections, GALO can also be used to address layout problems where each operator's workspace requirements are non-equal. More specifically, the concept of a super-operator is adopted in this work to model the type of operators whose workspace requirement is larger (i.e., encompassing more than one grid space). Such a super-operator can be represented in GALO as an aggregate of multiple fictitious regular operators that each occupies a single grid space. Bi-directional communication links with a weight of 9 are defined between each pair of these fictitious operators to ensure the genetic algorithm will likely collocate these fictitious operators in close proximity, so that the aggregated space by these fictitious operators can be used to accommodate the large space requirement of the super-operator.

Another significant improvement of GALO is the adoption of a pathfinding algorithm for computing the transit cost between two grid cells. This is achieved by using the Dijkstra's algorithm. The output of this algorithm is an adjacency matrix that contains the shortest path between every pair of open cells in the workspace. Compared to the original prototype genetic algorithm program (which uses the Euclidian distance between the centres of two open cells as a measure), GALO's assessment of physical separation among open cells is more precise and better represents the transit effort required to move from one location to another in a workplace. The introduction of this algorithm also improved the runtime efficiency of the overall program.

From an analyst's point of view, GALO follows a 5-step process in workspace layout modelling. In each step, the analyst has the ability to define a portion of the input data required for the genetic algorithm, including workspace information, operator and communication requirements, and algorithm parameters, before starting the simulation. User defined models can be saved as text files and later re-loaded into

GALO for further analysis. Once a simulation is complete, the optimal solutions are presented to the analyst in a graphical format, together with meta simulation data regarding the algorithm's efficiency.

To sum up, complex work environments such a command centre are critical facilities of a military operation. These are collaborative workplaces where a team of operators work interdependently to support a common mission. The layout of such a workplace is an important design consideration since the position of operators (and their workstation) affects communication effectiveness between collaborators and the overall efficiency of information flow in the workplace. The GALO program that was described in this document provides a software tool to examine optimal layout arrangements using a genetic algorithm.

References

[1] Wang, W. (2017). An algorithmic solution to improve command centre layout, In *Proceedings of the Human Factors and Ergonomics Society – 61st Annual Meeting,* Santa Monica, CA: Human Factors and Ergonomics Society, Defence Research and Development Canada, External Literature, DRDC-RDDC-2017-P112.

[2] Wang, W. (2018). Optimal configuration of a genetic algorithm for command space layout analysis, In *Proceedings of the Human Factors and Ergonomics Society – 62nd Annual Meeting,* Santa Monica, CA: Human Factors and Ergonomics Society, Defence Research and Development Canada, External Literature, DRDC-RDDC-2018-P156.

List of symbols/abbreviations/acronyms/initialisms

DRDC	Defence Research and Development Canada
GALO	Genetic Algorithm for Layout Optimization
GUI	Graphical User Interface
RCN	Royal Canadian Navy

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 tasking agency, is entered in Section 8.)

DRDC – Toronto Research Centre
Defence Research and Development Canada
1133 Sheppard Avenue West
Toronto, Ontario M3K 2C9
Canada

2a. SECURITY MARKING (Overall security marking of the document including special 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.)

GALO—Genetic Algorithm for Layout Optimization: A technical documentation for modellers and developers

4. AUTHORS (Last name, followed by initials – ranks, titles, etc., not to be used)

Lin, J.; Wang, W.

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 references cited.)
January 2019	28	2

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

Reference Document

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

DRDC – Toronto Research Centre
Defence Research and Development Canada
1133 Sheppard Avenue West
Toronto, Ontario M3K 2C9
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 number under which the document was written.)
01ab	

10a. DRDC PUBLICATION NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)	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-D139	

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.)

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

Workplace layout; Genetic Algorithm; modelling and simulation

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

GALO (Genetic Algorithm for Layout Optimization) is a desktop application tool that uses a genetic algorithm to determine the optimal layout of a group of operators in a workplace. In GALO, a workplace is modelled as a rectangular grid that is comprised of many cells. Each cell is defined as an open workspace that can be assigned to an operator, a barrier that functions as a spatial obstruction, or an unassignable space that could not be designated as a workspace. An optimal layout is defined as the one that best supports inter-operator communication (or interaction). Communication requirements are modelled as weighted directional connections. For each workplace model, after inter-operator communication requirements are specified, GALO has the ability to generate the optimal layout configuration based on a genetic algorithm. This Reference Document provides an instruction guide for future users of the tool, and a technical documentation of the program for future software developers.

GALO (*Genetic Algorithm for Layout Optimization*) est une application bureautique qui utilise un algorithme génétique pour déterminer l'aménagement optimal du lieu de travail d'un groupe d'opérateurs. GALO permet de modéliser un lieu de travail sous la forme d'une grille rectangulaire constituée de nombreuses cellules. Chaque cellule représente un espace de travail ouvert pouvant être attribué à un opérateur, une barrière faisant office d'obstacle spatial ou bien un espace ne pouvant pas être attribué ni désigné comme zone de travail. On définit l'aménagement optimal comme étant celui qui favorise le plus la communication (ou les interactions) entre les opérateurs. Les exigences de communication sont modélisées sous la forme de liaisons directionnelles pondérées. Une fois qu'on a précisé les exigences de communication entre les opérateurs pour chaque modèle de lieu de travail, GALO a la capacité de déterminer l'aménagement optimal en fonction d'un algorithme génétique. Le présent document de référence renferme un guide d'instructions pour les futurs utilisateurs de l'outil, ainsi que des renseignements techniques sur le programme à l'intention des futurs développeurs de logiciels.