This document provides the instructions for using software for solving the models and algorithms that allow the joint designation of an efficient and effective system of compact, protected areas with their connecting corridors as described in the manuscript 'An Integrated Approach to the Design of Compact and Connected Reserve Systems'.

1 OPL Projects

Description for solving optimization models P_1 to P_4: The 0-1 quadratic optimization model P_1, the linear mixed-integer model P_2, the 0-1 quadratic optimization model P_3, and the linear mixed-integer model P_4 are modeled using CPLEX Optimization Studio. For these models, the input parameters are the size of the grid system (m, n), the minimum coverage requirements for each species s (n_s), and the matrices that describe the availability of each species in the $m \times n$ grid system.

We provide four different OPL projects to solve these models, and the OPL projects are named as below.

1. EMSModelP_1
2. EMSModelP_2
3. EMSModelP_3
4. EMSModelP_4

To solve the optimization models P_1 and P_2, the user needs to select the value of the parameter lambda following the steps described in Algorithm 1 in the manuscript.

Sample input file for these OPL projects is given below.

```
/*********************************************
 * OPL 12.10.0.0 Data
 * Author: Lakmali Weerasena
 * Creation Date: Apr 3, 2021 at 7:18:36 PM
 *********************************************/

// The following text file provides the input data for the test problem described
in Weerasena et al. (2014).
 n=14; // the number of rows in the grid system (input parameter)
 m=14; // the number of columns in the grid system (input parameter)
 s=61; // total budget available for the reserve system (input parameter)
 s1=49; // the minimum coverage requirement for species Type 1 (input parameter)
 s2=41; // the minimum coverage requirement for species Type 2 (input parameter)
 s3=44; // the minimum coverage requirement for species Type 3 (input parameter)
 lambda=1.5; // the initial input parameter for Algorithm 1



// species-availability matrices for three type of species are given below. The
data matrices are named as areaofs3, areaofs2, areaofs3 (input matrices)

areaofs1
 =[[1   0   1   1   1   1   0   1   0   1   0   0   0   0]
 [1     1   0   1   1   0   0   0   1   0   1   1   0   0]
 [1     0   1   1   0   0   0   1   1   1   1   0   1   1]
 [1     0   0   1   1   1   0   0   0   0   0   1   0   0]
 [1     0   0   0   0   0   0   0   1   0   0   1   1   0]
 [0     1   1   0   0   0   1   1   0   0   1   1   0   0]
 [0     1   1   0   0   0   0   0   0   0   1   1   1   0]
 [1     1   1   0   0   1   0   0   0   0   0   0   0   0]
 [0     0   1   1   0   0   0   0   1   0   1   1   0   1]
 [0     0   0   0   0   1   1   0   1   0   1   1   0   0]
 [0     0   1   1   0   1   1   1   1   1   0   0   1   1]
 [0     0   1   1   0   1   1   1   0   1   1   1   0   0]
 [0     0   0   1   1   1   1   1   1   1   1   0   0   0]
 [1     0   1   0   0   0   0   1   1   1   0   0   0   1]
 ];
```

```
areaofs2
 =[[0   1   1   0   0   1   0   1   0   0   0   0   0   1]
 [0     1   1   1   0   1   1   0   0   0   0   0   1   1]
 [1     1   1   1   0   1   0   1   1   1   1   1   0   0]
 [1     0   0   0   1   0   0   1   1   0   0   0   0   0]
 [1     1   0   0   0   1   0   0   1   1   1   1   0   0]
 [0     1   1   1   0   0   0   0   1   1   1   1   1   1]
 [0     0   0   0   0   0   1   0   0   0   0   1   0   1]
 [1     1   0   0   1   0   0   0   0   0   1   1   0   0]
 [1     1   0   0   1   0   1   0   0   1   0   0   0   0]
 [0     0   0   0   1   1   0   1   1   1   0   0   0   0]
 [1     0   0   0   1   1   1   0   1   1   1   0   0   0]
 [0     0   0   0   1   0   1   1   1   0   1   0   0   1]
 [1     0   0   0   0   0   1   1   1   0   1   0   0   1]
 [0     0   1   0   1   0   0   0   1   0   1   1   0   0]
 ];

areaofs3
 =[[0   1   1   0   1   1   1   0   0   1   0   1   0   0]
 [0     0   1   1   1   1   1   1   0   0   1   0   0   1]
 [1     1   1   1   1   1   1   0   0   1   0   1   0   1]
 [0     0   1   0   1   0   0   0   0   0   1   1   0   1]
 [1     1   1   0   0   0   0   0   0   1   0   1   0   0]
 [0     1   0   0   1   1   0   0   0   0   1   0   0   1]
 [0     1   1   0   1   0   0   0   0   0   0   1   0   0]
 [1     1   0   1   0   0   0   1   0   1   0   0   1   0]
 [1     0   0   1   0   0   1   0   0   1   0   1   1   0]
 [1     1   0   0   1   0   0   0   1   0   1   0   0   0]
 [0     0   0   1   0   1   1   1   0   1   0   1   0   0]
 [0     0   1   1   0   1   0   1   1   1   1   0   1   1]
 [0     1   0   1   1   0   1   1   1   0   1   0   0   0]
 [0     0   0   1   0   0   0   0   0   0   0   1   0   0]
 ];
```

MATLAB programs

Description: *ClusterConnect* is an algorithm written in MATLAB to obtain corridor sites to connect the original clusters and to determine a set of boundary sites whose removal maintains feasibility of the system with the added corridors and removed boundary sites. This procedure is described in the manuscript 'An Integrated Approach to the Design of Compact and Connected Reserve Systems'.

We provide three different MATLAB script files that are needed to obtain the outcome described in this manuscript. All other required functions are included in the *ClusterConnect* folder.

1. 'RunFileClusterConnect.m' script file
2. 'RunFileModelFive.m' script file
3. 'RunFileModelSix.m' script file

'RunFileClusterConnect.m' script file for Algorithm 2

The output of Algorithm 2 is a set of corridor sites that cover the largest number of additional species including the number of extra species from each type already appearing in the clusters. These corridor sites can be obtained by running the 'RunFileClusterConnect.m' script file. This script file requires two input files and three input parameters.

Required input data

1. Upload the text file ('ClusterData.txt') which describes the disjoint clusters of core areas. The data is input as an $m \times n$ matrix. Sites in each cluster are denoted by successive integers 2, 3 ,4,….
   Sample cluster input data is given below. It shows a 5×6 reserve system with four input clusters.

2	2	0	0	0	0
2	0	0	0	4	0
0	0	3	3	0	0
0	0	3	3	0	5
0	0	0	0	0	5

2. Upload the species-availability data text file ('SpeciesData.txt'). This is an $Sm \times n$ binary matrix where S is the number of species in the reserve system.
   Sample input species-availability data is given below. It shows the availability of two types of species in the reserve system. The first block (rows 1-5) corresponds to the availability of species Type 1 in a 5×6 reserve system while the second block (rows 6-10) corresponds to the availability of species Type 2 in a 5×6 reserve system.

1	1	0	1	1	0
1	0	0	0	1	1
1	0	0	1	0	0
0	0	0	1	0	1
0	0	0	0	0	0
1	1	1	0	0	0
1	0	0	0	1	1
0	0	1	1	0	0
0	0	1	1	0	1
0	1	0	0	0	1

3. Input the vector *MinimumCoverage*, where *MinimumCoverage* specifies the requirements for the minimum number of sites *n_s* to protect each type of species *s*. This is a 1×S row matrix where *S* is the number of species.
4. Input the value *NumSpe*, where *NumSpe* is the number of species in the reserve system.
5. Input the value *NumClu*, where *NumClu* is the number of original clusters in the reserve system clusters.

Output of the 'RunClusterConnect.m' script file

The algorithm returns two output files

1. A data file which contains the original clusters and the sites selected for corridors (ClustermultiData.txt).
2. The number of extra species ('rhs.txt') of each type already appearing in the clusters.

‘RunFileModelFive.m’ script file to solve Model P_5 (Uniform Costs)

The output of Model P_5 is a set of boundary sites that can be removed from the system to restore feasibility when the site costs are uniform. That is, a solution that satisfies constraints (3)-(4) in model (P1) of the manuscript. These boundary sites can be obtained by running the ‘RunFileModelFive.m’ script file. The implementations utilize the MATLAB integer-programming solver (intprog) to solve the 0-1 linear Model P_5. This script file requires three input files and two input parameters.

Required input data

1. Upload the text file ('ClustermultiData.txt’) which describes the selected sites for the corridors and original clusters (one of the output files of the ‘RunFileClusterConnect.m’ script file).
   Sample input data is given below. It shows a 5×6 reserve system with four clusters and five corridor sites. Sites in each cluster are denoted by successive integers 2, 3 ,4, 5 and the corridor sites are denoted by 1.

2	2	0	0	0	0
2	1	0	1	4	1
0	1	3	3	0	1
0	0	3	3	0	5
0	0	0	0	0	5

2. Upload the text file (‘rhs.txt’) which describes the number of extra species from each type already appearing in the clusters (one of the output files of the ‘RunFileClusterConnect.m’ script file). This is a 1×*S* column matrix, where *S* is the number of species in the reserve system.

3. Upload the species-availability data text file (‘SpeciesData.txt’). This is an *Sm*×*n* binary matrix, where *S* is the number of species in the reserve system.

4. Input the value *M*, where *M* is the number of corridor sites added to the connect the original clusters.

5. Input the value *NumSpe*, where *NumSpe* is the number of species in the reserve system.

Output of the ‘RunFileModelFive.m’ script file

1. A data file (‘ClustermultiData.txt’) which contains a set of boundary sites that can be removed from the system and the coverage of each removed site.

‘RunFileModelSix.m’ script file to solve Model P_6 (General Cost)

The output of Model P_6 is a set of boundary sites that can be removed from the system to restore feasibility when the site costs are general. That is, a solution that satisfies constraints (3)-(4) in model (P1) of the manuscript. These boundary sites can be obtained by running the ‘RunFileModelSix.m’ script file. The implementations utilize the MATLAB integer-programming solver (intprog) to solve the 0-1 linear Model P_6. This file requires four input files and two input parameters.

Required input data

1. Upload the text file ('ClustermultiData.txt’) which describes the selected sites for the corridors and clusters (one of the output files of the ‘RunClusterConnect.m’ script file). Sample input data is given below. It shows a 5×6 reserve system with four clusters and five corridor sites. Sites in each cluster are denoted by successive integers 2, 3 ,4, 5 and the corridor sites are denoted by 1.

2	2	0	0	0	0
2	1	0	1	4	1
0	1	3	3	0	1
0	0	3	3	0	5
0	0	0	0	0	5

2. Upload the text file (‘rhs.txt’) which contains the number of extra species from each type already appearing in the clusters (one of the output files of the ‘RunClusterConnect.m’ script file).
3. Upload the species-availability data text file (‘SpeciesData.txt’). This is an *sm*×*n* binary matrix, where *s* is the number of species in the reserve system.
4. Upload the cost data text file (‘cost.txt’) which describes the budgetary cost of purchasing, conserving or maintaining each site (*i*, *j*). This is an *m*×*n* matrix.
   Sample cost input data is given below. It shows a cost data for a 5×6 reserve.

0.13	0.64	0.07	0.06	0.52	0.18
0.65	0.71	0.04	0.45	0.20	0.59
0.02	0.08	0.02	0.31	0.95	0.42
0.90	0.65	0.15	0.04	0.78	0.25
0.66	0.49	0.31	0.22	0.75	0.33

5. Input the value *ConnectCost*, where *ConnectCost* is the additional cost (in excess of total budget available for the reserve system) incurred when *M* corridor sites are added to connect the original clusters.
6. Input the value *NumSpe*, where *NumSpe* is the number of species in the reserve system.

Output of the 'RunFileModelSix.m' script file

1. A data file ('CheckSol.txt') which contains a set of boundary sites that can be removed from the system, the coverage of each removed site and the corresponding cost.