A matheuristic for the Roadef 2020 Challenge Final version

F. Parreño, R. Alvarez-Valdes, C. Parreño-Torres

1 Integer formulations

We have developed several integer linear formulations with different objective functions that approximate the objective of the original problem.

Initially, we define the variables:

$$x_{it} = \begin{cases} 1, & \text{if intervention } i \text{ starts in period } t,\ i \in I,\ t \in H \\ 0, & \text{otherwise} \end{cases}$$

r_{it} is the average risk of starting intervention i at time t and q_{it} is the sum of the quantiles, calculated independently for each intervention in all scenarios in each period in which it would be in process. The first two formulations only differ in the objective function and share constraints (4)-(6). OF_1 only considers the risk and not the excess. OF_2 approximates the excess by using individual interventions quantiles. The third formulation attempts to minimize the sum of the maximum risks in each period and requires new variables, M_t, for the maximum risk in period t, and constraints (7) linking them to the x_{it} variables.

$$(OF_1) \qquad Minimize \sum_{i \in I} \sum_{t \in H} r_{it}\, x_{it} \tag{1}$$

$$(OF_2) \qquad Minimize \sum_{i \in I} \sum_{t \in H} (\alpha q_{it} + (1-2\alpha) r_{itj})\, x_{it} \tag{2}$$

$$(OF_3) \qquad Minimize \sum_{t \in H} (\alpha\, M_t + (1-2\alpha) \sum_{i \in I} r_{ij}\, x_{it}) \tag{3}$$

$$\sum_{t \in H} x_{it} = 1 \qquad \forall i \in I \tag{4}$$

$$l_t^c \leq \sum_i \sum_{t' | t' \leq t \leq t' + \Delta_{it'} - 1} r_{it'}^{ct} x_{it'} \leq u_t^c \qquad \forall c \in C, \forall t \in H \tag{5}$$

$$x_{it_1} + x_{jt_2} \leq 1 \qquad \forall (i,j,t) \in Excl \tag{6}$$

$$\forall t_1 | t_1 \leq t \leq t_1 + \Delta_{it_1} - 1$$

$$\forall t_2 | t_2 \leq t \leq t_2 + \Delta_{jt_2} - 1$$

$$\sum_i \sum_{t' | t' \leq t \leq t' + \Delta_{it'} - 1} r_{it'} x_{it'} \leq M_t \qquad \forall t \in H \tag{7}$$

2 Matheuristic

The procedure consists basically of two phases. First, the integer models are solved, producing a set of solutions that are optimal for the previously designed models, but not for the original problem being solved. Then, the second phase improves the solutions obtained according to the actual objective function using a VND algorithm.

2.1 Solving the integer model

A preliminary computational study showed that the third model took too long on some instances and that the second model, including information about the quantiles as an approximation to the true objective function, produced better results than the first model. Therefore, in the first phase of the procedure, we solve the second model, but instead of solving it once for the given value α, we run it several times, depending on the available running time and the difficulty of the instance, varying α, so as to obtain a pool of optimal solutions with some degree of diversity among them. The solutions in the pool are distributed in two lists that are processed in parallel and each solution in these lists is improved by a Variable Neighbourhood Descent (VND) procedure.

2.2 VND

The following improvement movements are applied in order and when an improvement is found the procedure returns to the first movement. The pseudocode is shown in Algorithm 1.

1. BestT: Determine the best time for an intervention. This is done in two ways. First, by looking at the time with the minimum risk. Second, by looking for a time in which if the intervention is carried out, the excess is reduced.

2. Exchange: Swap two interventions, exchanging their starting times.

3. Ejection: Determine the best times for a pair of interventions. Two interventions are removed from the solution. The first of them is placed in the starting time of the other, and the second looks for the best time to be placed.

4. Ruin and build: Part of the solution is removed and rebuilt using a randomized constructive algorithm.

Algorithm 1 VND Algorithm

```
function VND                                              ▷
    s ← Initial Solution                                  ▷ Best solution found so far
    listBT, listEx, listEjec ← ∅                          ▷ Changes made
    Ejec, Ex ← true
    Step 1: Best T
    if listBT = ∅ then
        BestT(s)
        if BestT(s) then Update listBT, listEx, listEjec
    else
        do BestT(s) only for changes in listBT
        listBT ← ∅
        if BestT(s) then Update listBT, listEx, listEjec
    Step 2: Exchange two interventions
    if Ex then
        Exchange(s)
        Ex ← false
        listEx ← ∅
        if Exchange(s) then
            Update listBT, listEx, listEjec
            Go to Step 1
    else if listEx <> ∅ then
        do Exchange(s) only for changes in listEx
        listEx ← ∅
        if Exchange(s) then
            Update listBT, listEx, listEjec
            Go to Step 1
    Step 3: Ejection chain
    if Ejec then
        Ejection(s)
        Ejec ← false
        listEjec ← ∅
        if Ejection(s) then
            Update listBT, listEx, listEjec
            Go to Step 1
    else if listEjec <> ∅ then
        do Ejection(s) only for changes in listEjec
        listEjec ← ∅
        if Ejection(s) then
            Update listBT, listEx, listEjec
            Go to Step 1
    Step 4: Ruin and Build
    RuinnBuild(s)
    if RuinnBuild(s) then
        Update listBT, listEx, listInt
        Go to Step 1
    return s
```

2.3 Reactive VND

The first move is always applied, but the other moves are applied according to certain probabilities. Initially, for a given number of iterations, all the moves are applied, but a record is kept of which moves are finding improvements and then their probability is updated, giving more probability to those moves that obtain more improvements.

2.4 Intensification

The best solutions obtained throughout the VND procedure are kept in an elite set. When the VND ends, it is applied again but only to these elite solutions.

3 Results on set C

The integer linear models have been solved by CPLEX 20.1 and the algorithms run on 2 CPUs on an i7-9700 @3GHZ with 16GB of memory. Table 1 shows the results obtained for the C set instances with time limits of 900 and 5400 seconds.

Table 1: Results of the matheuristic algorithm instances C

Instance	Short	Long
C_01	8515.92736	8515.90377
C_02	3541.78774	3542.88774
C_03	33520.21415	33515.56415
C_04	37591.97358	37591.66321
C_05	3166.30094	3166.18962
C_06	8444.60849	8401.52453
C_07	6084.76905	6083.85952
C_08	11182.35800	11171.53800
C_09	5603.09811	5600.74717
C_10	43352.46520	43351.41863
C_11	5750.62059	5749.95735
C_12	12743.34424	12731.52173
C_13	42488.49500	42488.03152
C_14	26489.57773	26495.85386
C_15	39761.59833	39759.59100