# Optimization of reinforced concrete structure scheduling using a genetic algorithm

Arthur Calvi[1], Xavier Jourdain[1], Farid Benboudjema[1]

[1] *Université Paris-Saclay, ENS Paris-Saclay, CNRS LMT-Laboratoire de Mécanique et Technologie, 4 avenue des sciences 91190 Gif-sur-Yvette*

**Abstract :** In construction, the scheduling will determine the profitability of a site. In this article, a proof of concept is presented to optimize the time, the cost and the quality of construction scheduling using a genetic algorithm. To maintain the constructibility of a schedule during the run of the genetic algorithm a matrix of constraints is used. Data needed by the algorithm are collected from a BIM digital model. Results obtained on simple building plans are encouraging.

**Keywords :** optimization, construction scheduling, reinforced concrete building, BIM, genetic algorithm

## 1 Introduction

Even thought several studies have shown the possibility to optimize construction scheduling with various algorithms, there are little studies about reinforced concrete structures. Indeed, before the rise of Building Information Modelling (BIM) the collection of all data needed about the reinforced concrete structure was a long and arduous process. Today, the BIM has permitted a fuller approach of construction scheduling [1]. Thus, optimizing reinforced concrete structure scheduling with algorithms became feasible once again.

## 2 State of the art

Researchers have approached solving scheduling problems with different tools and techniques. Beliveau.Y and Mrad.A [2] propose a case-based reasoning algorithm (KNOW PLAN) to generate and simulate construction schedules, O'Brien & al [3] create a model-based program (MOCA) with the use of a critical path to perform the scheduling of a reinforced concrete structure. Levitt.R and Kartam.N [4] propose an expert system called SIPE which use artificial intelligence to generate correct activity networks for multi-story office building projects, Karim.A and Adeli.H [5] use a neural network approach to optimize construction cost and ultimately cost-duration tradeoff by varying the project duration, Faghihi & al [6] use a genetic algorithm to optimize time, cost, and job-site movement of the crew of steel constructions. Among all these methods, genetic algorithm seems to prevail. It is the most used method in construction scheduling since it is capable to solve multi-objective problem [7]. For all algorithms the main problem was to collect the data, namely the dimensions, the materials used and other parameters. However, thanks to Building Information Modelling (BIM) which is thriving in the world of construction, it is today easier and faster to collect data with digital models.

## 3 Case

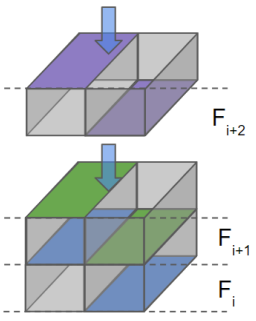


**Figure 1** – Nesting floors

The study is limited to repetitive buildings in the aim of considering floors like blocks which will be nested *(cf fig1)*. The main advantages of this approach are :

- To simplify the study with just one floor.
- To determine only one construction schedule for one floor and to reuse it for all other floors of the building.
- To save time of calculus.

The considered floor contains inferior slabs as well as superior slabs in order to determine the most effective schedule. The study is also limited to building composed only with slabs and walls, which is the most common type of building in France. Therefore, the algorithm is limited to a specific range of buildings and could be enhanced afterward to consider other structures (Reinforced concrete buildings with columns, steel frame, etc.).

# 4 Approach

The aim is to realize the construction scheduling with a genetic algorithm. At the beginning, the program will randomly choose parts of walls and slabs per phase to determine schedules. Then along the generations, the program will modify these schedules with operations such as : mutation and crossover. At the end of a generation, a selection of the best individuals is performed according a certain fitness function. Thus, the program will progressively generate better schedules.

The first step is to process the data for the genetic algorithm. The program receives the floor plan thanks to the IFC file of the building. According to the size of the smallest mould available, walls of the floor plan are cut into smaller elements. The method is to realize the scheduling and thereupon to group this smaller elements to form wall parts that match with the different mould's sizes available. Regarding slabs, the program realize a simpler cut : the aim is to ease the work of the construction crew by limiting the number of pouring area per phase. Thus, the slabs are cut into elements that have a surface equal to the half of the moulded surface capacity per phase. Thereafter, all the cut elements will be designated by the primary elements. The program can take different entries to run : *1)* the wall and slab capacity of moulding of the construction crew or *2)* the number of phase of the scheduling. In both cases the other parameter is calculated by the program and is considered like an aim.

Once the data are processed, the genetic algorithm can be considered. The first step is to think about the search space and the criteria. The search space is wide since it contains the numbers of primary elements and those could be arranged at any phase. Thus, the space search contains $N^p$ possibilities. $N$ is the number of primary elements of a floor plan and $p$ is the number of phases. Yet the algorithm should provide a scheduling which meet several criteria :

- Constructibility : to verify that the scheduling is physically feasible. For instance to verify that an inferior slab is moulded before all the walls above it.
- Economic : For instance to use the same number of moulds per phase to buy the smallest number of moulds.
- Time : to respect the maximal number of phases.
- Convenience : to ensure the possibility for the crew to reach the work zones at any time of the schedule and to minimize their moving.

In this case, it is effortless to explore all the search space. In addition, the different criteria are not consistent and could lead to an ineffective research. To avoid this issue, all the operators which perform the initialization, the mutation and the crossover have been coded in the aim to force a schema which should satisfy the constructibility criteria. The operators have been biased, which is a common method in genetic algorithm [8]. Indeed, the initialization create a population of individuals which have a constructibility superior to 0.75. It means that in average the individuals are 75% buildable. The notion of constructibility is explained in the part *5 : Genetic Algorithm*. All the other operators : mainly mutation and crossover ensure to at least keep the constructibility and sometimes to enhance it. To force a schema into the algorithm, each individual is associated with a list which contains the moulding phase of an element and a matrix of constraints inspired by the work of Faghihi and al [6] about metallic construction. This two objects are closely connected and defined by :

**Matrix of constraints.** It contains the following information : the object side by side and the primary elements which should be moulded at different phases : primary element 1 before 2 or the reverse. This matrix is initialized with *2* and *1* just by reading the plan composed by the primary elements. Then the matrix is modified with *3* and *-3* according to each individual.

**List of phases.** It contains the following information : the moulding phase of the primary elements and the bracket of phases where the primary element can be moulded without losing the constructibility. During the initialization of an individual and with the help of the MoC the brackets are calculated for each primary element. The brackets are updated during the crossover.

$$\text{primary element} = pe \qquad \begin{matrix} pe_1 & pe_2 & \cdots & pe_n \end{matrix}$$

$$MoC = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix} \begin{matrix} pe_1 \\ pe_2 \\ \vdots \\ pe_n \end{matrix}$$

$$a(i,j) = \begin{cases} \text{None if } i = j \\ \text{1 if } pe_i \text{ and } pe_j \text{ are side by side} \\ \text{2 if } pe_i \text{ and } pe_j \text{ should be moulded at different phases} \\ \text{3 if } pe_i \text{ is an inferior slab in relation to } pe_j \\ \text{-3 if } pe_i \text{ is a superior slab in relation to } pe_j \end{cases}$$

**Figure 2** – Matrix of constraints (MoC)

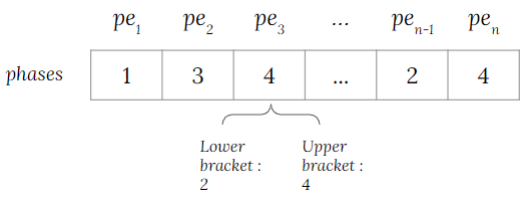


**Figure 3** – List of phases

Thanks to this two objects, it is possible to monitor the constructibility of an individual. The use of this two objects force a schema into the algorithm and reduce the search space.

# 5 Genetic Algorithm

In this section, the genetic algorithm presented on *figure 4* is explained. A genetic algorithm simulate the evolution of a population and imitate the natural selection [9]. In our case, the population contains individuals that are possible schedules. Indeed, an individual contains the moulding phase of each primary elements. The initial population is created, then to form the next generation the population will be affected by different operators :

**Selection :** The selection used is the probabilistic selection introduced by Holland in 1975 [10].

**Crossover :** Because the individual have to respect the constructibility, this operator firstly find potential individuals (parent2) to cross with the one selected (parent1). Then to form a child, the parent 1 will receive the moulding phase of the parent 2 with a probability of 0.75 for each primary elements of walls that can be exchanged. The operator can not exchange the moulding phase of primary elements of slabs yet.

**Mutation :** The operator change the moulding phase of primary elements with the probability of 5%.

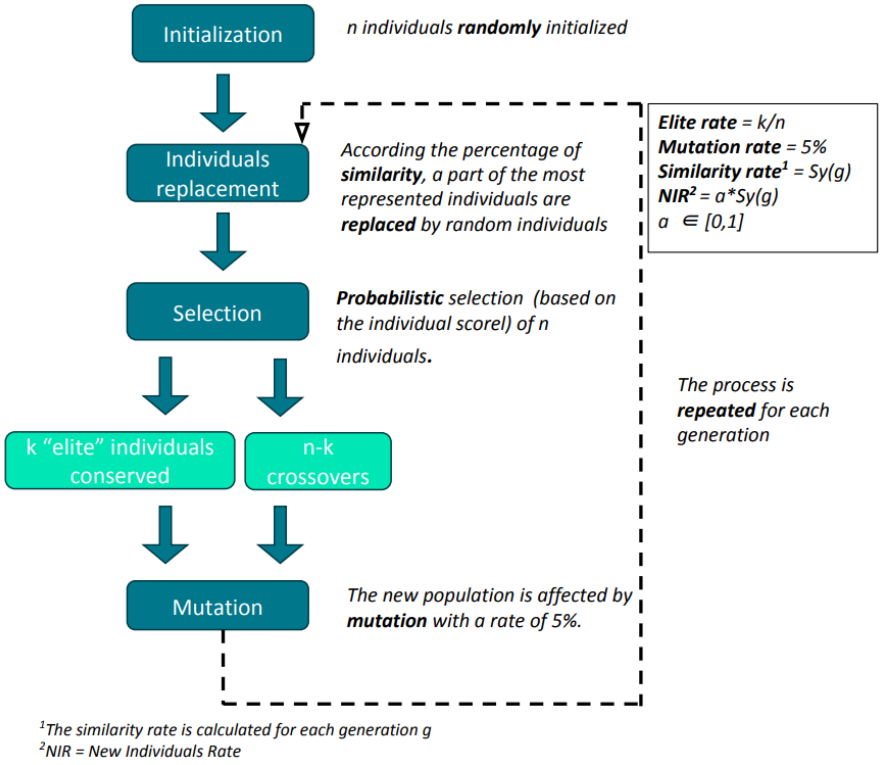


**Figure 4** – Genetic algorithm schema

The schedule should respect multiple objectives, therefore the fitness function used to select individual is a weighted sum of all the criteria that the individual should meet. The fitness function proposed is :

$$F = \frac{\gamma_1 \times C_{walls} + \gamma_2 \times C_{slabs} + \gamma_3 \times C_{moulds} + \gamma_4 \times C_{constructibility} + \gamma_5 \times C_{accessibility}}{\sum_i^5 \gamma_i} \tag{1}$$

The coefficient $\gamma_i$ have been fixed according to the importance of each criterion and with the help of several simulations. The different criteria are : ($\sigma$ is the standard deviation, $\mu$ is the average over the phases).

- The time of work to mould walls should be smoothed : $C_{walls} = \left\langle 1 - \frac{\sigma_{time}}{\mu_{time}} \right\rangle_+$ and $\gamma_1 = 5$.
- The time of work to mould slabs should be smoothed : $C_{slabs} = \left\langle 1 - \frac{\sigma_{time}}{\mu_{time}} \right\rangle_+$ and $\gamma_2 = 5$.
- The number of moulds should be smoothed : $C_{moulds} = \left\langle 1 - \frac{\sigma_{time}}{\mu_{time}} \right\rangle_+$ and $\gamma_3 = 1$.
- The constructibility criteria counts how many primary elements it is possible to build : $C_{constructibility} = \frac{\sum_i^N pe_{buildable}}{\sum_i^N pe}$ and $\gamma_4 = 10$.
- The distance that workers should walk to reach the both faces of a wall during its moulding phase, this criterion is calculated with the help of a Fast-Marching algorithm [11]. A brief explanation is available in Appendix A. $C_{accessibility} = \left\langle 1 - \frac{\sigma_{dist}}{\mu_{dist}} \right\rangle_+$ and $\gamma_5 = 3$.

**Parameters.** The main parameters of a genetic algorithm are the size of a population, the number of generations, the mutation rate and the elite rate. The mutation rate and the elite rate are both fixed to 5% according the literature [12]. Concerning the number and the size of a generation it depends on the floor plan complexity. Generally, the population is composed by 250-500 individuals and there are 20-40 generations. The last parameter to consider is the New Individuals

Rate, the value is fixed at 10% of the similarity of a population. It permits introducing new individuals when a population presents a high redundancy of individuals.

**Monitoring the population.** To understand how the genetic algorithm impact the population, constructibility, similarity, average score, best score and the standard deviation of scores are recorded for each generation. The similarity counts the number of individuals that have not twins in the population, it permits controlling if operators are too biased and force a schedule in particular.

# 6 Results

In the first place, the algorithm was developed with a simple example of floor plan. The results are shown in the appendix B. Then the algorithm was tested on a more complex building. The results are presented in the next paragraph and illustrated by *figures 5 and 6*. Parameters : *Generations : 30*, *Population size* = 500 *Individuals*.

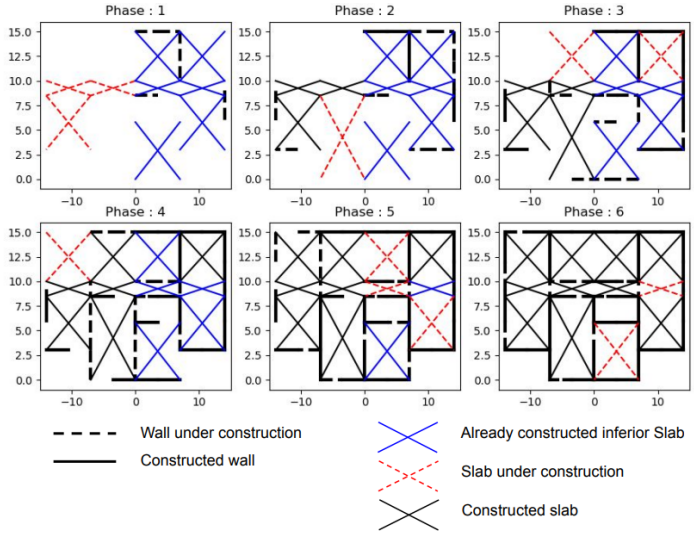


**Figure 5** – Solution found by the algorithm

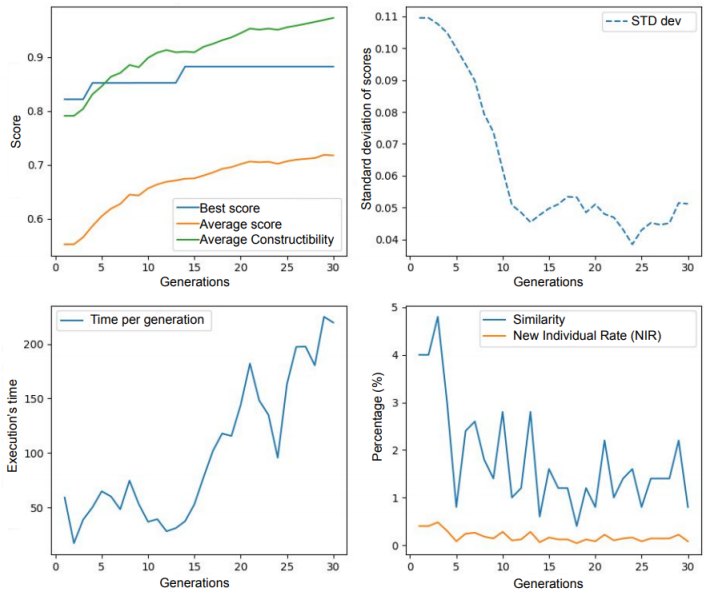


**Figure 6** – Monitoring

The blue crosses represent the slabs already moulded during the construction of the lower floor. The algorithm is respecting the concept of nesting blocks since it is a repetitive building. The entry for the crew is considered between the points (-7,0) and (7,0). The given solution satisfy the accessibility of the crew for any phase, the constructibility is to 100% and the time of work for walls, slabs and moulds are smoothed. Concerning the genetic algorithm, the similarity stay below 5%, the average score is increasing along the generations and the standard deviation of scores is decreasing. Those trends prove that the algorithm leads the population over the generations to a higher score target and concentrates the population to a smaller interval. The execution time is about 45min for this floor plan, running on a quad core processor with a clock frequency of 2.8Ghz and 16Gb of RAM.

# 7 Conclusion

To conclude, even thought the program is not complete and need modifications or improvements such as : a better fitness function using a Pareto front [13] [14], to complete the crossover operation to take into account slabs exchanges, and to expand the program to columns and beams, the results demonstrate that using a genetic algorithm to optimize the cost, the time and the quality of a schedule is possible. With the BIM, almost every details of a construction plan could be collected. Thus, future algorithms should be able to perform accurate scheduling for reinforced concrete structures using IFC files. The algorithm could be extended to other construction methods. Moreover,it could be easily modified to prefabricated buildings. Indeed, a modification of the accessibility criterion and the cut parameter for walls and slabs should be sufficient to run the scheduling of prefabricated buildings.

# Références


[1] Aouad.G Murat Tanyer A. Moving beyond the fourth dimension with an ifc-based single project database. *Automation in Construction*, 14(1) :15–32, 2005.

[2] Beliveau Y Morad A. Knowledge-based planning system. *J Constr Eng Manag*, 117(1) :1–12, 1990.

[3] O'Brien Evans M Fischer M, Aalami F. Model-based constructibility analysis : the moca system. *CIB W78 workshop on computer integrated construction. CIB, Helsinki*, 1994.

[4] Levitt R Kartam N. Intelligent planning of construction projects. *J comput Civ Eng*, 4(2) :155–176, 1990.

[5] Karim A Adeli H. Scheduling/cost optimization and neural dynamics model for construction. *J Constr Eng Manag*, 125(5) :368–376, 1997.

[6] Julian H. Kang Vahid Faghihi, Kenneth F. Reinschmidt. Construction scheduling using genetic algorithm based on building information model. *Expert Systems with Application*, 41 :7565–7578, 2014.

[7] Kenneth F. Reinschmidt Julian H. Kang Vahid Faghihi, Ali Nejat. Automation in construction scheduling : a review of the literature. *Int J Adv Manuf Technol*, 81 :1845–1856, 2015.

[8] Nicholas F. McPhee Riccardo Poli, William B. Langdon. *A Field Guide to Genetic Programming*, chapter 9.4 - Operator Bias. 2008.

[9] Darrell Whitley. A genetic algorithm tutorial. *Statistics and Computing*, 4 :65–85, 1994.

[10] M. Lozano J.L. Verdegay F. Herrera. Tackling real-coded genetic algorithms : Operators and tools for behavioural analysis. *Artificial Intelligence Review*, 12 :265–319, 1998.

[11] Sethian J.A. A fast marching level set method for monotonically advancing front. *Proc.Natl. Acad. Sci.*, 93 :1591–1595, 1996.

[12] Andre D. Keane M.A Koza J. R., Bennett III F. H. Four problems for which a computer program evolved by genetic programming is competitive with human performanc. *Artificial Intelligence Review*, 12 :265–319, 1996.

[13] A. Osyczka and S. Kundu. A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. *Structural Optimization*, 10 :94–99, 1995.

[14] Nicholas Nafpliotis Jeffrey Horn and David E. Goldberg. A niched pareto genetic algorithm for multiobjective optimization. *IEEE World Congress on Computational Intelligence*, 1 :82–87, 1994.

# A Annexe - Fast Marching

The following figures present how the accessibility criterion is calculated with the help of the Fast Marching algorithm. For each phase, the walk distance from the access zone to reach the both sides of the wall which will be moulded during this phase is calculated. The distances are represented by grey levels.

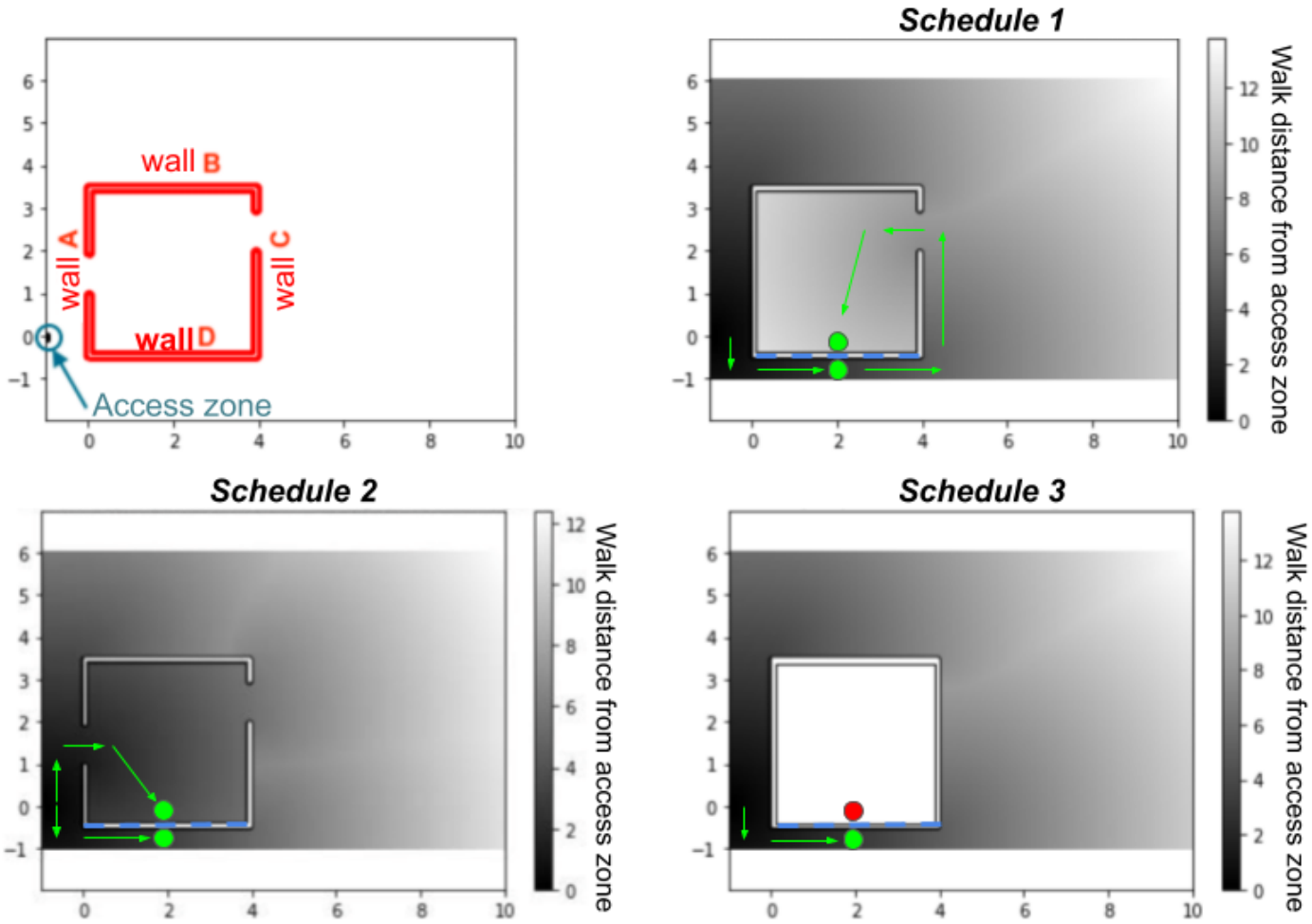


**Figure A.1** – Fast Marching

In this example : the wall D should be moulded during the phase. The accessibility of the wall D is calculated for 3 different schedules. For the schedule 1 and 2 the both sides of the wall D can be reached, but not for the schedule 3. The schedule 3 is therefore excluded. Then, the schedule with the shortest distance to reach the both side of the wall D (Schedule 2) receive a better accessibility score for this phase.

# B Annexe - Results for an other building

These are the results of the building used to develop the algorithm. The parameters of the algorithm are :

- Population size = 250 individuals
- Number of generations = 30
- Other parameters are fixed according the section *Genetic algorithm - Parameters*

The program cut walls into primary elements of 5m :

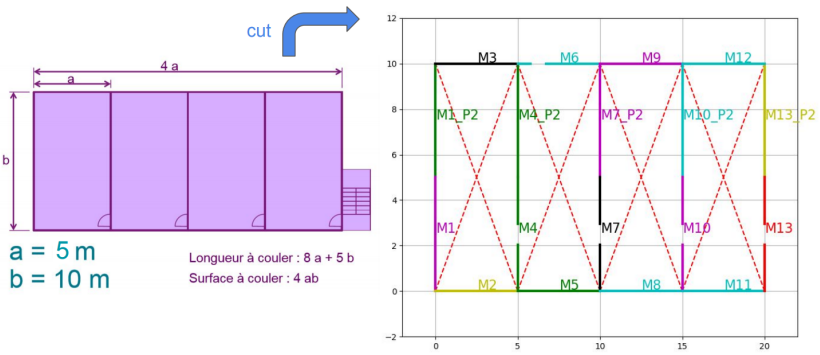


**Figure B.1** – Cut operation

The results (30 generations over 23min) :

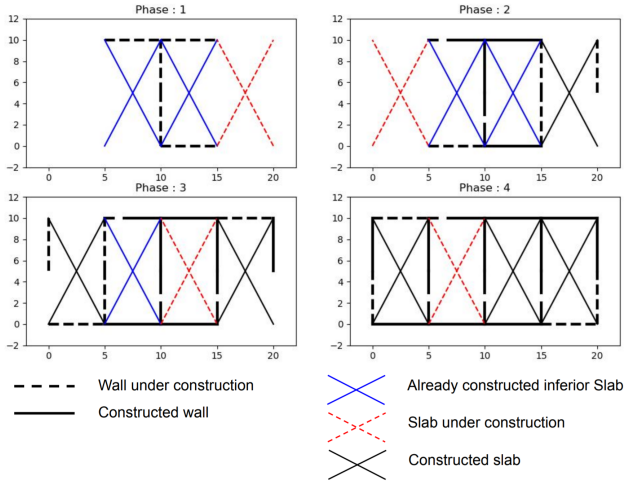


**Figure B.2** – solution

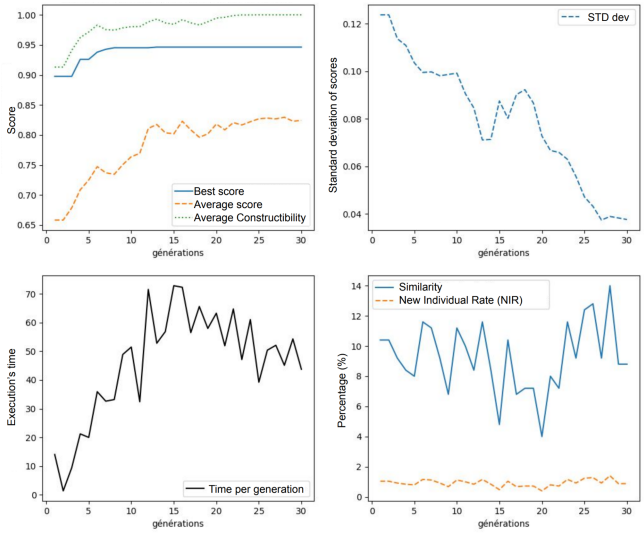


**Figure B.3** – monitoring

| | phase 1 | phase 2 | phase 3 | phase 4 |
|---|---|---|---|---|
| Moulds (door) | 2 | 1 | 1 | 1 |
| Wall length $(m)$ | 25 | 20 | 25 | 20 |
| Slab surface $(m^2)$ | 50 | 50 | 50 | 50 |
| $\sum accessibility_{pe,i}$ $(m)$ | 131 | 101 | 116 | 143 |

| Criterion | $C_{walls}$ | $C_{slabs}$ | $C_{moulds}$ | $C_{constructibility}$ | $C_{accessibility}$ |
|---|---|---|---|---|---|
| score | 0.89 | 1.0 | 0.65 | 1.0 | 0.87 |