Construction scheduling using Genetic Algorithm based on Building Information Model

Vahid Faghihi a,⁎, Kenneth F. Reinschmidt a, Julian H. Kang b

a Zachry Dept. of Civil Engineering, Texas A&M Univ., College Station, TX 77843-3136, United States
b Dept. of Construction Science, Texas A&M Univ., 3137 TAMU, College Station, TX, United States

ARTICLE INFO
Keywords:
Construction project schedule
Building Information Model
Genetic Algorithm
Stability relations

ABSTRACT
The construction project schedule is one of the most important tools for project managers in the Architecture, Engineering, and Construction (AEC) industry that makes them able to track and manage the time, cost, and quality (a.k.a. Project Management Triangle) of projects. Developing project schedules is almost always troublesome, since it is heavily dependent on project planners' knowledge of work packages, on-the-job-experience, planning capability and oversight. Having a thorough understanding of the project geometries and their internal interacting stability relations plays a significant role in generating practical construction sequencing. On the other hand, the new concept of embedding all the project information into a 3-dimensional representation of a project (a.k.a. Building Information Model or BIM) has recently drawn attention to the construction industry.

In this paper, the authors demonstrate a novel approach of retrieving enough information from the BIM of a project and then develop construction sequencing for the installation of the project elements. For this reason a computer application is developed that can automatically derive a structurally (statically) stable construction sequence, using the concept of the Genetic Algorithm (GA). The term “structurally stable sequencing” in this article refers to the sequencing order of erection in which the structure remains statically stable locally and globally during the entire installation process. To validate the proposed methodology, the authors designed 21 different experiments and used the proposed method for generating stable construction schedules, which all were successfully accomplished. Therefore, this methodology proposes a novel approach of construction project application of the GA, as an Expert System tool.

1. Introduction

The development of project schedules is a critical part of all types of projects including engineering, manufacturing, construction, and others. However, engineering education, whether at the graduate or undergraduate level, typically provides little instruction on how to develop good construction or fabrication schedules. Construction engineers and managers on projects learn on the job how to visualize the sequence of activities that will lead to good, feasible schedules, without formal training. By integrating project scheduling with virtual three-dimensional geometric modeling, students could learn through hands-on interaction with the system how to generate more effective project networks and schedules. The literature review performed by the authors (Faghihi, Nejat, Reinschmidt, & Kang, 2014) showed that other researchers were mainly focusing on scheduling with respect to resource handling and usage. This research intends to show how the geometric information of a project can be used directly to generate construction sequence. This direct use of the geometric information and the list of all the components from the BIM of the project can eliminate the potential occurrence of errors in transcription of the 3D data.

The main purpose of this research is to create an environment for construction planners to have a visually interactive communication between the planning process and 3D models of the project at each increment of time. This environment uses an algorithm that simulates the natural evolutionary process in a rule-based approach to reach a feasible project schedule. The natural evolutionary process in this research considers the relationships and dependencies of the project elements from the Matrix of Constructability Constraints (MoCC), presented by the authors (Faghihi, Reinschmidt, & Kang, 2014a) and shown in Eq.(1), and uses previous knowledge gained through experience from similar works. The determination and calculation of the relationships and dependencies of the project elements are handled through a
well-developed mathematical algorithm reading all the geometric information on the project elements from the 3D project file. The necessary common knowledge and previous experience would help the initial phase of developing the algorithm by defining sets of rules to express element dependences. The geometry reading algorithm and common knowledge used in this paper is described briefly in the “Reading the Geometry” section and detailed description is provided in another paper by authors (Faghihi et al., 2014a). Using this algorithm, a project planner can change the work strategy using predefined parameters and the 3D model, as a visual representation of the entire project, to see the effects of different strategies on the schedule. These work strategies can be, but not limited to, determining crane locations, material unloading spots, installation direction priorities, and finding workaround solutions.

The proposed algorithm can potentially extend to have a two-way interactive environment between project geometric model (BIM) and its schedule. This environment can bring new dimensions to the management team of the project in which changing the design of the 3D model directly and immediately results in a new and updated project schedule. Also, when the project schedule is manipulated, the extended algorithm can detect which parts or elements of the project may not be constructible regarding the updates in the project schedule by highlighting them. The extended version of this algorithm (Faghihi, Reinschmidt, & Kang, 2014b) and the managerial usage of the entire proposed algorithm (Faghihi, Reinschmidt, & Kang, 2014c) is described in details in other research papers.

\[ \text{Eq. (1). Matrix of Constructability Constraints} \]

\[
\begin{bmatrix}
A_1 & A_2 & \ldots & A_n \\
S_{1,1} & S_{1,2} & \ldots & S_{1,n} \\
S_{2,1} & S_{2,2} & \ldots & S_{2,n} \\
\vdots & \vdots & \ddots & \vdots \\
S_{n,1} & S_{n,2} & \ldots & S_{n,n}
\end{bmatrix}
\]

where:
- \( A_i \): is the project tasks (geometric elements in the 3D model or the activities to be scheduled),
- \( S_{ij} \): is dependencies between elements (that could be either 0 or 1 showing not dependent or dependent respectively).

Considering Graph Theory and definitions of the matrices, the MoCC is a Directed Adjacency Matrix or a Directed Design Structure Matrix (Directed DSM). The Directed DSM is a matrix representing the project network (Kanda, 2011) and has been in use since the 1960s (Steward, 1962). The spatial translation of 3D BIM to the MoCC and then to the project network is shown in Fig. 1. The Genetic Algorithm will later use this project network (or MoCC) to generate construction schedule for the given 3D model. The novel approach of decoding the 3D model stability rules into a Directed DSM is briefly mentioned in this article and mainly described in another article written by the authors (Faghihi et al., 2014a). In the current article, a new approach of generating and developing construction schedules using the Genetic Algorithm and the MoCC is described, tested, and verified. In the mentioned figure, the matrix in the middle, MoCC, is generated based on the spatial relations of the geometry elements in the 3D model considering the previously mentioned common knowledge of stability. By having this matrix, as a numerical representation of the stability relations in a BIM, the next step will be generating a constructable and stable project schedule for that specific 3D model. To reach a set of fully stable project schedules, the stability score can be assigned to each possible solution. This stability score will be the percentage of the project elements that are scheduled for installation in a stable order (i.e. obeying the constructability constrains calculated in MoCC). Then, the target score will be 100% of the stability that should be reached. This approach brings the environment of optimization methods where the stability score should be maximized. To accomplish that outcome, one of the best methods is the Genetic Algorithm (GA), which has already been proven to be useful in project management and in Expert Systems as an optimization tool. In addition, since the proposed matrix consists of zeros and ones, representing stability relations, it is very well suited for the GA fitness function. The entire methodology process is summarized in the following schematic view (Fig. 2). More about the process will be described later in this paper.

2. Reading the geometry

In this paper, the 3D model input to the algorithm is in a standard text-based format, called Industry Foundation Classes (IFC).
IFC, as an open and neutral specification, is an object-based file format. The data model of this neutral file format is developed by buildingSMART with the main goal of facilitating interoperability between the AEC companies, as a commonly used file format for BIM (buildingSMART, 2013). The IFC model specification is listed as an official International Standard ISO 16739:2013 (International Organization for Standardization, 2013). More definitions on how the reading and extraction algorithm is working will be described as follows.

The current developed algorithm is supporting only the structural elements, columns (IfcColumn) and beams (IfcBeam), from the IFC file format of the BIM of a project. For the simplification of the calculations, these two element types (beam and column) were assumed as lines with a boundary box around them. An example of how the boundary box of beams or columns are assumed is shown in Fig. 3. By going through the IFC standard (buildingSMART, 2011), defining the dimensions of all of these elements can be calculated and simplified to just start point and end point of a line.

In this research, whenever two elements are intersecting within their boundary box regions, they are assumed as physically connected. Locating these connections by performing calculation on the data retrieved from IFC file and applying stability rules as shown in Table 1, the MoCC can be generated (Faghihi et al., 2014a). More description of the table can be found in the mentioned article.

### Table 1

<table>
<thead>
<tr>
<th>Column</th>
<th>Same level</th>
<th>Upper level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beam</td>
<td>Supporting columns or beams</td>
<td>-</td>
</tr>
<tr>
<td>Wall</td>
<td>Adjacent columns and beams</td>
<td>-</td>
</tr>
<tr>
<td>Slab</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Roof</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Door</td>
<td>Container wall</td>
<td>-</td>
</tr>
<tr>
<td>Window</td>
<td>Container wall</td>
<td>-</td>
</tr>
</tbody>
</table>

### 3. Genetic Algorithm

The Genetic Algorithm is an optimization tool that uses a heuristic search that mimics the natural evolutionary process (Mitchell, 1996). Using a well-defined fitness function, as the objective function or core metric, randomly generated initial genomes can evolve into optimized solution(s) for a given problem, considering objectives that are mathematically defined by the fitness function. The GA has become a practical optimization tool in construction-related fields of research due to the inherent features and characteristics. These characteristics can be summarized in having ability to perform multi-objective optimization, avoiding exhaustive search, and ability to find (at least) near optimum solutions (Ward Systems Group, Inc., 2013), which all are fitted with the problems in construction field.
3.1. Literature review

Davis introduced the use of the Genetic Algorithm for optimizing job shop scheduling in the 1980’s (Davis, 1985). A few years later, Wall used this algorithm for resource constrained scheduling as his dissertation topic (Wall, 1996). He optimized the sequencing of job shops by feeding the GA with more than 1000 different types of scheduling problems ranging from small job shops to project scheduling (10–300 activities, 3–10 resource types). Chan et al. presented their work as scheduling of resource-constrained construction projects using GA (Chan, Chua, & Kannan, 1996). In their paper, they showed how their proposed GA-Scheduler can optimize the resource usage and do the resource leveling to come up with better project schedules compared to the heuristic methods in regard to resource allocation. Gonçalves et al. continued this work later by tackling resource-constrained multi-project scheduling (Gonçalves, Mendesb, & Resendec, 2008). Murata, Ishibuchi, and Tanaka (1996) introduced their multi-objective GA to reach Pareto fronts of flowshop scheduling and described how their GA was developed.

Using the Genetic Algorithm, as an Expert System tool, for solving and applying in optimizing schedules has been a focusing point for researchers. Li, Gao, and Shao (2012) presented their “active learning genetic algorithm based” (ALGA) model that simplified the optimization and integration of scheduling and process planning. Chen, Wu, Chen and Chen (2012) developed their own GA-based scheduling algorithm to do job shop scheduling with reentrant process and parallel machines characteristics. In 2012, other researchers proposed their intelligent GA for optimizing scheduling problem of truck distributions for different centers (Lee, Kim, & Jou, 2012). Later in the same year, Kavesghar, Huynh, and Rahimian (2012) improved the efficiency of the Genetic Algorithm use for solving the Quay Crane Scheduling Problems (QCSP). Joo and Kim (2013) used their developed GA to solve single-machine scheduling problem with multiple rate-modifying activities and time-dependent deterioration. Chang, Wu, Lee and Shen (2014) used emergency and immediate logistics scheduling problems to prove the usefulness of their developed Greedy-Search-based Multi-Objective Genetic Algorithm (GSMOGA) in such cases.

In solving construction project scheduling problem, Toklu (2002) used Genetic Algorithm for both having and not having resource constraints. He used a model for defining the relationships between the network activities (Start-to-Start or SS, Start-to-Finish or SF, Finish-to-Start or FS, and Finish-to-Finish or FF). Toklu simplified the relationships by defining basic mathematical equations; for instance he defined the Start-to-Start relation between task i and task j as \( T_i + L_{ij} \leq T_j \), where \( L_{ij} \) is the start-to-start time lag between task i and task j. As seen in the above-defined mathematical relation, there can be different relationship types between task i and task j, but the defined mathematical relation still remains valid. Jaskowski and Sobotka (2006) described their Evolutionary Algorithm as a system having the inputs such as relationship structure, available resources and resource requirements of each project task. With these inputs, their system was able to find the shortest duration of performing the project (Jaskowski & Sobotka, 2006).

The researchers mentioned above have demonstrated the usefulness of the GA for construction schedule optimization. Although they were mainly focusing on resource leveling and assignment, none of them generated construction schedules using the GA. The main goal of all the mentioned researchers was to find better optimization approaches for available project schedules and not developing one. To develop a logical construction schedule, the stability rules of the structure should be known. Having these rules stored to a well-known matrix (Directed DSM), the GA can start generating the desired construction schedules using the rules in the matrix.

The proposed algorithm in this research can be used to produce the construction schedule from scratch and for this reason a fitness function is required. To fulfill this requirement, using the derived stability matrix from the BIM of the project will help. The defined objective in this GA optimization is maximizing the constructability of the project, which means that the GA would try to find constructible project schedules for the given project 3D model. The constructability of a project schedule herein is defined as having all the elements statically stable during and after the installation process. For example, the installation of a beam is considered constructable and stable only if the two end structural supports (columns or beams) have been installed earlier.

Below are the general descriptions of core Genetic Algorithm functions and their definitions in this research.

3.2. Main GA functions

The following sections describe how the important functions of the proposed Genetic Algorithm are defined and working. There are less important functions running in the process to facilitate the calculations and transferring of genomes form one population to another; however, they are not mentioned in here.

3.3. Genome creation

In this approach the genomes consist of lists of elements to be installed in each time-unit (e.g. day, week, or month) throughout the total project duration. By this definition, a genome can be shown in either of the following two ways. The Matrix of Genome (MoG), as shown in Eq. (2), consists of \( n \) rows, each of which represents a single element from the 3D model, and \( k \) columns, indicating total installation duration. The non-zero value of \( g_{ij} \) shows the installation time \( j \) for the element, \( i \). For example, if \( g_{5,3} = 1 \), it means that element number 5 is scheduled to be installed in the time 3-unit (which could be either hour, day, or week based on user definition). If all the rows of this matrix are put in a single row in a way that the first column of a row gets placed after the last column of the previous row, then a single string of matrix values is generated (as shown in Eq. (3)) and is ready to be used in a GA population.

\[
\text{Eq. (2). Matrix of Genome (MoG)}
\]

\[
\begin{pmatrix}
D_1 & D_2 & \ldots & D_n \\
A_1 & A_2 & \ldots & A_n \\
\end{pmatrix}
\]

\[
\text{Matrix of Genome} \ (\text{MoG}) = A_1 [g_{1,1}, g_{1,2}, \ldots, g_{1,k}] \\
\vdots \\
A_n [g_{n,1}, g_{n,2}, \ldots, g_{n,k}] \\
\]

OR

\[
\text{Eq. (3). Genome}
\]

\[
genome = \{g_{1,1}, \ldots, g_{1,k}, g_{2,1}, \ldots, g_{2,k}, \ldots, g_{n,1}, \ldots, g_{n,k}\}
\]

where:

- \( n \): is the number of project tasks (geometric elements in the 3D model or the number of activities to be scheduled) and
- \( k \): total project time-unit (e.g. days, weeks, or months).

The length of the genome or the number of genes, \( g_{ij} \) in the genome is calculated as follows:

\[
\text{Eq. (4). Length of the genome}
\]

\[
\text{Length of the genome} = \text{number of genes} = n \times k
\]
In a random genome generation, the total project duration would be chosen based on initial data from the user. Then, a string of zeros and ones is generated, with the length calculated by multiplying the total project duration and the number of elements (tasks) retrieved from the 3D BIM file. Therefore the only limitation for this random genome would be as shown in Eq. (5):

\[ g_{ij} = 1 \ & \ g_{ij+1} = 0 \ - \ \forall m > j + 1 : g_{im} = 0 \quad (5) \]

The above condition simply means that if an element has been installed before, it cannot be reinstalled. The easiest way to create this genome is to spread out \( n \) number of ones in string of \( n \times k \) zeros, where \( n \) is the number of elements and \( k \) is the total number of time-units. With this condition, an element could be installed in one or more time-units if and only if all the time-units are sequential. To simplify the genome more, each element would be installed in only one time-unit and no more. Through this simplification, in each row of the genome matrix there would only be a single 1 and all other values would be 0. This simplification means that each element can be installed in one or less than one time-unit and the installation of an element cannot be extended to more than one time-unit. The time-unit can be assumed as hour, day, or week based on user preferences. As a realistic simplification, this assumption is mimicking real practices in different assembly processes specifically in construction projects. The logic that lies behind this simplification is the installation process for prefabricated elements. Prefabricated elements in construction projects can be listed as precast concrete panels, beams, and columns, steel structure elements, doors and windows, HVAC ducts and devices, pipelines, etc. These type of elements are typically fabricated outside the project jobsite (e.g. factory) and the installation process is performed by simply placing them in their specific locations, realistically assumed as instant installation. The same simplification is also valid in other manufacturing processes such as soldering electronic components on the circuit board and assembling body parts of a car or even an aircraft.

The algorithm is programmed in a way that it schedules the model elements to be installed using cumulative normal distribution, simulating the S-curve work load in real project completion phase.

3.4. Elite members

In the GA, it is desirable that the fitness function score does not decrease from one population to the next when maximizing the objective. Thus, when generating a new population, some of the better genomes are allowed to move from the current generation to the next generation, unchanged. This method is known as elitist selection and those selected genomes are called elite members of the old population.

As an example, looking at Tables 6 and 7 from Appendix A, it is noticeable that the first genome, which has the highest score, is moved to the next generation intact. In that example, the elite rate is set to 20%, one genome out of entire 5 member population.

3.5. Fitness function

The fitness function for the GA could have multiple variables to measure and in this case it is considered as a multi-objective GA. However, in this step of the research only one objective is defined: constructability of the project sequences.

The constructability objective is the most important objective for the construction sequences and should be closely tracked and measured. Above all, a project schedule should be completely constructible covering all the project components scheduled to be installed. The constructability score, in the form of a percentage, is calculated based on the number of the elements that are obeying the constraints defined in MoCC, divided by total number of the elements, as described below.

To determine the constructability score, the Matrix of Constructability Constraints (MoCC) is developed as the key factor, which has been shown in Eq. (1). In this matrix, all the rules and constraints related to the specific 3D BIM are defined element-by-element using the rationales behind the geometry and their dependencies to each other. Having the MoC, mentioned in Eq. (2), a function can easily be defined to read the sequence of elements installed from the genome and determines the elements that are scheduled to be installed in each time-unit. Then, for each of those elements, all the constructability constraints (prerequisite elements) would be retrieved from MoCC. Then again, each element from this list of constraints would be checked against the MoG to see if it has been scheduled for installation before or not. In case all the prerequisite elements (constructability constraints) are satisfied, the element would be considered as constructible. By dividing the total number of constructible elements by the total number of elements and multiplying by 100, the constructability percentage of the genome is calculated.

All the descriptions for the fitness function of this paper is summarized in the Fig. 4. The final calculated score is the basis for the genome (schedule) selection described in the next section.

3.6. Selection method

Selection functions that are traditionally used in GA are categorized in the following three groups: (Sivaraj & Ravichandran, 2011)

- **Proportionate selection**: in this method (better described as Roulette Wheel selection), each genome will get a score assigned, \( f_i \), using the fitness function. Then, the cumulative fitness of the entire population, \( Pf \), will be calculated. After that, the probability of selection for each genome is calculated as \( psel_i = \frac{f_i}{Pf} \). This fitness score is then used to assign the probability of being selected to each individual genome.

- **Ranking selection**: In linear ranking selection (Baker, 1987), first the individual genomes are ranked based on their fitness values. Those genomes that have higher fitness values will be ranked higher and those with lower fitness values will have lower ranks. Then, the genomes are selected based on a probability that is linearly relative to the rank of the genomes in the population.

- **Tournament selection**: it consists of running several “tournaments” between a few individual genomes selected randomly from the population. The one with the best fitness (the winner of each tournament) is then chosen for crossover.

For this research, the Fitness Proportionate Selection (FPS, a.k.a. roulette wheel method) introduced by Holland (1992) is chosen to be the selection function to pick parent genomes for crossover function.

The selection function uses the calculated constructability scores from the fitness function for the FPS method. The value of genome score divided by total sum of all scores of a given population is equal to the selection chance of that genome for the crossover function. The higher the score of a genome is, the more chance is has to be selected as a parent in crossover function.

3.7. Crossover

In the GA, two genomes are selected as parents from the previous generation, which has just been created, and paired for breeding two new genomes as their children for populating the new
As described before, the parent selection would be handled using the specified Selection Function. In this section, the process of how to breed two new child genomes from the parent genomes is described. The parent selecting and breeding process will continue until the desired number of genomes in the new population (defined by the user and equal to the size of previous population) is reached.

To do the crossover for this GA, after selection of the two parents, a random duration will be selected from one of the parents and both parents would be split from that random point in time, in equal proportion. Mathematically the cutting point for both parents would be calculated as follows:

\[
\text{Cutting point of parent } 1 = \text{random}(1, \text{duration of parent } 1) - 1 \\
\text{Cutting point of parent } 2 = \text{integer} \left( \frac{\text{Cutting point of parent } 1}{\text{duration of parent } 1} \times \text{duration of parent } 2 \right) 
\]

(6)

The reason for this proportional cutting point is that, since both parents are schedule representations of a single project, they would naturally have a similar pattern. This way of cutting creates better children because each part of the two parents may carry similar construction schedule information and pairing them up in this way results in more constructible children. The efficiency of this approach has been tested and verified during this research, as the growth in constructability score is sped up by doing the crossover in the mentioned way.

The concern here is that all the elements are supposed to be scheduled for installation in one and only one time-unit. This assumption simply means that there should not be an element such that all the values in its row in the MoG are zero and no element that has multiple values of 1. In other words, there should be no row whose sum is not equal to 1, meaning that every element is installed at one time. The Validation Function performs this validating process to ensure all the genomes are satisfying this condition. More description of the Validation Function can be found in “Genome Validation” section.

The following steps show how this crossover function is performed. Here two different genomes have been defined. These two genomes represent two different installation sequences (in this case) for a three-element structure with durations equal to 4 and 3 time-unit respectively for genome 1 and genome 2 (see Fig. 5).

\[
\text{Genome 1} = [0 \ 0 \ 1 \ : \ 0 \ 1 \ 0 \ : \ 0 \ 1 \ 0 \ 0] \\
\text{Genome 2} = [0 \ 0 \ 1 \ : \ 0 \ 1 \ 0 \ : \ 1 \ 0 \ 0]
\]

Fig. 5. Sample genomes.

As mentioned earlier, these genomes can be shown as the following two matrices shown in Fig. 6. The first genome (matrix) schedules no installation on the first time-unit. The first element is sequenced for the third time-unit and the second element for the fourth time-unit and the last element for the second time-unit. In the second genome (or matrix), the third element is scheduled for the first time-unit and the second element for the second time-unit and the first element for the last time-unit.

To perform the crossover function on these two genomes (matrices), as described above, a random time-unit will be selected from the range defined in Eq. (6). The duration range from the MoG 1 for random number pick up is 1 to 3 time-unit. Assumption in this example is that the randomly picked number is 3. Therefore the MoG 1 will have a cutting point between its third and fourth time-unit. Using the second part of the same mentioned equation, the cutting point of the MoG 2 will be cutting between the second and the third time-unit. These cutting points are illustrated as red dotted lines in Fig. 6.

\[
\text{MoG 1} = \begin{bmatrix}
0 & 0 & 1 & : & 0 \\
0 & 0 & 0 & : & 1 \\
0 & 1 & 0 & : & 0
\end{bmatrix} \\
\text{MoG 2} = \begin{bmatrix}
0 & 0 & : & 1 \\
0 & 1 & : & 0 \\
1 & 0 & : & 0
\end{bmatrix}
\]

Fig. 6. Sample MoGs, showing cutting points as red dotted lines. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)
are installed. Based on the MoG shown in Fig. 8, the beam is scheduled to be installed after all the structural supports are installed. The early matrices are called parent matrices and the later ones are the child matrices. The result of the crossover operation on these two MoGs considering the calculated cutting points will be as the following.

As seen in Fig. 7, both genomes are invalid regarding the placement schedule of the elements. This invalidity is due to both not scheduling an element for installation (element number 2 in new MoG 1 and element number 1 in new MoG 2) and double installation time for another element (element number 1 in new MoG 1 and element number 2 in new MoG 2). There is another function named “Validation Function” that is responsible to make necessary changes in genomes to make them valid. This function is described further in this paper.

3.8. Mutation

As a part of the GA, there is a mutation function changing the genomes in some point randomly. The random mutation helps to optimize to avoid being trapped in local minima. To do this mutation, a random gene needs to be picked and inverted its value to mimic the mutation. After doing this mutation, the genome needs to be validated also. The mentioned random mutation could be mathematically shown as below:

\[
\begin{align*}
\text{randE} &= \text{random}([1, \text{number of elements}]) \\
\text{randT} &= \text{random}([1, \text{number of time-units}]) \\
\text{Select Gene} &= g_{\text{randE.randT}} = 1 \\
\text{if} g_{\text{randE.randT}} = 0 \rightarrow g_{\text{randE.randT}} = 1 \\
\text{else} g_{\text{randE.randT}} = 0
\end{align*}
\]

However, a better mutation function could be described in another way. Each schedule genome has multiple project elements that are not scheduled for installation obeying the MoCC, if and only if the constructability score is less than 100%. To define a better mutation function reaching for the 100% score faster one way is to find out those MoCC violating elements and randomly mutate their installation time to somewhere later than the current time. Since changing the installation time of a violating element to a later time is not violating any predefined rules, Genome Validation is not required anymore and the calculation speeds up.

A simple example of how the mutation function effects the genome is shown below. For this purpose, imagine that the matrix shown in Fig. 8 for a very simple structure consisting of two columns supporting a beam. The columns are considered as the first and the second elements defined in the following matrix and the beam is the third one.

As shown in the Fig. 8, one of the columns is scheduled to be installed in the first time-unit and the other in the third. Since the beam is supported by the two columns, as assumed in this example, it should be installed after all the structural supports are installed. Based on the MoG shown in Fig. 8, the beam is scheduled to be installed in the second time-unit, before installation of the second column. Therefore, the constructability score of this genome is 66% (two elements out of three are scheduled correctly regarding the structural stability of the model).

\[
\begin{align*}
\text{New MoG 1} &=
\begin{bmatrix}
0 & 1 & 1 \\
0 & 0 & 0 \\
0 & 1 & 0
\end{bmatrix} \\
\text{New MoG 2} &=
\begin{bmatrix}
0 & 0 & 0 \\
1 & 1 & 1 \\
1 & 0 & 0
\end{bmatrix}
\end{align*}
\]

In case this genome is selected to be mutated in the GA process, the third element (the beam) will be chosen for mutation. In the first step of the mutation process, the initial installation time value will be set to zero. Then, the function will find the latest installation time of the structurally supporting element, which in this example is the latest time for two columns, calculated as the third time-unit. Therefore, the mutation process determines the range from the next time-unit, 4, and the total duration of the schedule, which is again 4, and will pick a random number from that range. In this simple example the random time-unit selected from the range will be 4. Thus, the gene that should be mutated is the 4th time-unit of the 3rd element. The mutated MoG is shown in Fig. 9.

Since the mutation function is not violating the structural stability rules, the validation of the genome is not needed.

3.9. Genome Validation

As mentioned earlier in “Crossover” section of this paper, it is very likely that the newly generated genomes after the crossover function are not valid project schedules. This invalidity can be due to not scheduling at least one element for installation at all or schedule it more than one time to be installed. Considering the Fig. 7, both child MoGs are invalid, considering the two above mentioned criteria. To resolve this problem, the “Validation Function” is added to the methodology. The logical formulation of this function is shown in Eq. (8).

\[
\begin{align*}
\forall i \in \{1, 2, \ldots, n\} \exists j \in \{1, 2, \ldots, k\}: g_{ij} &= 1 \\
\text{AND} \forall i \in \{1, 2, \ldots, n\} \& j \in \{1, 2, \ldots, k\}: c < k &\rightarrow g_{ij} = 1 \& g_{ij+c} = g_{ij-c} = 0 \\
\text{OR} \forall i \in \{1, 2, \ldots, n\}: \sum_{j=1}^{k} g_{ij} = 1
\end{align*}
\]

where:

- \(g_{ij}: \text{ the value of element } i \text{ in time-unit } j \text{ in the genome} \)
- \(i: \text{ the element number} \)
- \(j: \text{ time-unit step number} \)
- \(n: \text{ total number of elements} \)
- \(k: \text{ total number of time-unit steps} \)
- \(c: \text{ any random number between zero and } k \)

The Validation Function takes a look at the MoG and finds out which elements are not defined to be installed or they have multiple installation times. For the first case it randomly schedules the elements within the project duration and for the later it maintains the first installment and removes the rest. Taking the new MoG 1 of Fig. 7 as an example, the validated version of it will look like Fig. 10. In this example, the second installation for the first element is removed and randomly the second element is scheduled to be installed in the third time-unit.

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]
3.10. Calculation termination

Similar to all the Genetic Algorithm applications, this proposed method will repeat all the functions and steps mentioned above until all the population genomes (construction schedules) reach the full score (100%) for their constructability objective. At that point, the calculation will be terminated and the last population will be considered as the solutions to the scheduling problem. A simple example of how this methodology can generate project schedules, using all of the above mentioned functions, is available in “Appendix A” section. The example is the project scheduling development for the model shown in Fig. 1 using the proposed methodology of the paper.

4. Research validation

There are several different parameters and factors effecting the process of the proposed algorithm in this research. To validate that this proposed algorithm is working in any cases and it is a valid and usable methodology, different combinations of the values for the effecting parameters should be investigated to check the soundness of the results. Hence, for validating this proposed model, the “Experimental Validation and Design” is chosen and it fits well for this research, considering different types of validations (Landry, Malouin, & Oral, 1983). The parameters and their input ranges for doing this validation process is gathered through related literatures and are described in next section.

4.1. Experimental design

In an experiment, one or more process variables (or factors) should be changed intentionally so that the effect of the changes on one or more response variables could be monitored. The design of experiments (DOE) is an effective way in order to analyze the obtained data and produce valid and objective conclusions (NIST ITL, 2012) with a minimum number of experiments.

The variables selected to be changed in this research are as follows:

- Changing the Complexity of the input 3D BIM:
  - Number of elements
  - Connection types
- Changing Genetic Algorithm parameters:
  - Elite member percentage
  - Mutation rate
  - Number of genomes per population (population size)
  - Construction duration range

For the 3D model inputs, three different models have been created to represent simple, moderate, and complex BIMs. In this research, the authors are only focusing on structural models and architectural elements, as well as piping, equipment, and HVAC, are not included in these models. The level of complexity of the models is detected based on number of the structural elements, size of the model, and connection types between structural elements. The screen shots from three different BIM inputs to the method are shown in Fig. 11.

The model (a) in Fig. 11 is a simple structural model with 42 elements, 18 columns and 24 beams. The second model, (b), is a more complex model with 42 columns and 58 beams, summing up to 100 elements. The last model is a generic turbine building structural model with 274 elements that consists of 102 columns and 172 beams (146 girders and 26 joists). The last model is extracted from the models of typical turbine buildings used in the power-plant industry and is considered as one of the complex steel structures in construction industry. As the detailed descriptions of each 3D model input stated, these three 3D model input cover the complexity range for 3D steel structure from a simple one (Fig. 11(a)) to a complex one (Fig. 11(c)).

Previous researches have shown optimum ranges for the GA population size, Mutation rates, and Crossover rate (or Elite member rate). The researchers have shown different sizes for the GA population as the optimum value of the parameter. The proposed population sized ranges from 16 (Haupt, 2000) to 20–30 (Grefenstette, 1986; Schaffer, Caruana, Eshelman, & Das, 1989) and sometime up to 50–100 genome (Obitko, 1998). Similar sensitivity analyses for mutation rate has shown the range from 5% to 30% found the best optimum answers (Haupt, 2000). The crossover rate is stated to be set as high as 85% to 90% (Obitko, 1998), which by the definition of Elite member rate, the value for Elite member rate would be calculated as 10% to 15%.

Considering these values from other researchers, the authors in this paper selected a range from 20 to 100 genomes per population to cover almost all the mentioned population sizes. Similarly, the range from 5% to 20% for the Mutation rate and 5% to 30% for the Elite member rate are selected as the parameters of the experimental design. The GA parameters changes are shown in Table 2.

To design the experiment to prove the benefits of the proposed methodology for developing project schedules, the authors...
executed 21 different combinations of parameter changes. These 21 different cases used in this experiment consist of 3 model complexity changes as defined in Fig. 11 for each 7 GA parameter change sets shown in Table 2. All of the above changes in parameters (both GA parameters and model changes) would result in 21 different runs of the entire algorithm to validate its usefulness in almost any case.

The expected outcome for all of these different runs is to achieve multiple (the same number as the population size) complete construction sequences that satisfy the constructability and stability constraints of the model. These constructability constraints are calculated before in MoCC, as mentioned earlier. If it can be shown that all the different designed experiments are satisfying the objective of this research, it can be justified that the proposed algorithm is applicable for automatic development of stable construction project scheduling.

5. Results

As defined in the last section, different sets of inputs have been created and imported to the proposed algorithm to see how they could generate completely constructible project schedules. As described earlier, the constructability objective is defined as, having the elements of the project scheduled for the construction in a way that the local and global stability of the project (model) is preserved. This stability of the elements and the model is controlled by the MoCC that is calculated earlier. More details on how these different runs reach the objective of this research is shown in Table 3 and Fig. 12. The execution of this methodology was performed on regular personal computer (CPU: Intel® Core™ 2 Duo @ 316 GHz, RAM: 8 GB, OS: Windows® 7 Enterprise 64-bit).

The first label for the “Run Name” field in the Table 3 is as, Simple, Moderate, or Complex that are defined earlier as input model

<table>
<thead>
<tr>
<th>GA Parameter Sets:</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
<th>5th</th>
<th>6th</th>
<th>7th</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size</td>
<td>30</td>
<td>20</td>
<td>50</td>
<td>30</td>
<td>20</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>Elite number</td>
<td>20%</td>
<td>10%</td>
<td>30%</td>
<td>20%</td>
<td>10%</td>
<td>30%</td>
<td>5%</td>
</tr>
<tr>
<td>Mutation rate</td>
<td>10%</td>
<td>5%</td>
<td>15%</td>
<td>5%</td>
<td>15%</td>
<td>10%</td>
<td>20%</td>
</tr>
<tr>
<td>Duration range</td>
<td>15 ± 20%</td>
<td>10 ± 20%</td>
<td>20 ± 20%</td>
<td>10 ± 10%</td>
<td>20 ± 10%</td>
<td>15 ± 10%</td>
<td>25 ± 10%</td>
</tr>
</tbody>
</table>

Table 3
Completion results for different runs for the experimental design.

<table>
<thead>
<tr>
<th>Run name</th>
<th>1st%100 score occurred @</th>
<th>Ended @</th>
<th>Calculation duration per generation (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple, 30G, 15T</td>
<td>82</td>
<td>100</td>
<td>0.05</td>
</tr>
<tr>
<td>Simple, 20G, 10T</td>
<td>535</td>
<td>559</td>
<td>0.04</td>
</tr>
<tr>
<td>Simple, 50G, 20T</td>
<td>75</td>
<td>99</td>
<td>0.10</td>
</tr>
<tr>
<td>Simple, 30G, 10T</td>
<td>98</td>
<td>121</td>
<td>0.04</td>
</tr>
<tr>
<td>Simple, 20G, 20T</td>
<td>276</td>
<td>306</td>
<td>0.04</td>
</tr>
<tr>
<td>Simple, 50G, 15T</td>
<td>67</td>
<td>97</td>
<td>0.08</td>
</tr>
<tr>
<td>Simple, 100G, 25T</td>
<td>36</td>
<td>64</td>
<td>0.26</td>
</tr>
<tr>
<td>Moderate, 30G, 15T</td>
<td>7785</td>
<td>7817</td>
<td>0.13</td>
</tr>
<tr>
<td>Moderate, 20G, 10T</td>
<td>49,932</td>
<td>50,050</td>
<td>0.07</td>
</tr>
<tr>
<td>Moderate, 50G, 20T</td>
<td>4916</td>
<td>4954</td>
<td>0.24</td>
</tr>
<tr>
<td>Moderate, 30G, 10T</td>
<td>8217</td>
<td>8289</td>
<td>0.10</td>
</tr>
<tr>
<td>Moderate, 20G, 20T</td>
<td>4325</td>
<td>4364</td>
<td>0.11</td>
</tr>
<tr>
<td>Moderate, 50G, 15T</td>
<td>4888</td>
<td>4915</td>
<td>0.23</td>
</tr>
<tr>
<td>Moderate, 100G, 25T</td>
<td>3979</td>
<td>4013</td>
<td>0.66</td>
</tr>
<tr>
<td>Complex, 30G, 15T</td>
<td>81,183</td>
<td>81,210</td>
<td>0.68</td>
</tr>
<tr>
<td>Complex, 20G, 10T</td>
<td>244,200</td>
<td>244,709</td>
<td>0.42</td>
</tr>
<tr>
<td>Complex, 50G, 20T</td>
<td>4,146</td>
<td>4,204</td>
<td>1.26</td>
</tr>
<tr>
<td>Complex, 30G, 10T</td>
<td>52,078</td>
<td>52,165</td>
<td>0.57</td>
</tr>
<tr>
<td>Complex, 20G, 20T</td>
<td>105,523</td>
<td>105,657</td>
<td>0.54</td>
</tr>
<tr>
<td>Complex, 50G, 15T</td>
<td>17,185</td>
<td>17,215</td>
<td>1.05</td>
</tr>
<tr>
<td>Complex, 100G, 25T</td>
<td>13,806</td>
<td>14,512</td>
<td>3.43</td>
</tr>
</tbody>
</table>
complexity shown in Fig. 11 representing models (a)–(c), respectively. In that field, the label G is showing the number of genomes in each population for that specific run and the label T represents the mean of the initial duration range as defined in Table 2.

Fig. 12 shows the trend in which each of the designed experiments followed for maximizing the objective. The goal of this research, as mentioned earlier, is to have stable and constructible project schedules for any given 3D model. By maximizing the defined objective, constructability score, the GA tends to incline to the highest score, 100%, in each population generation step. As seen in the Fig. 12, some of the designed experiments reach the complete score much faster than the others. This difference in

<table>
<thead>
<tr>
<th>Calculation rounds</th>
<th>Calculation duration per generation</th>
<th>Total calculation duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoG</td>
<td>-0.05</td>
<td>0.98</td>
</tr>
<tr>
<td>Number of elements</td>
<td>0.58</td>
<td>0.62</td>
</tr>
<tr>
<td>Population size</td>
<td>-0.32</td>
<td>0.55</td>
</tr>
<tr>
<td>Mean of the initial duration range</td>
<td>-0.31</td>
<td>0.43</td>
</tr>
</tbody>
</table>

Fig. 13. Screen-shot of a generated construction sequence (Moderate, 50G, 15T).
the pace of completing the calculations is due to several parameters. The two most important effecting parameters are: input 3D model complexity and number of genomes in each population. The more complex the input 3D model is, the harder for the algorithm to schedule the entire 3D model elements. Since the infinite number of model elements is not possible, the calculation will always merge to the defined objective. However, the calculation time increases by having more elements in the model. Also by increasing the number of genomes in each generation, there will be better chances to have better crossovers and mutations to reach the goal of the experiment. To show the trade-off between the different input variables and the calculation duration of the proposed algorithm, a new metric is defined in this research. This new metric is named NoG (the Number of Genes in each population) and is calculated as shown in Eq. (9).

\[ \text{NoG} = \text{Average genomelength} \times \text{Population size} \]  

where based on Eq. (4):

\[ \text{Average genomelength} = \frac{\text{Number of 3D elements}}{\text{Mean of initial duration range}} \]

The correlation coefficient between the NoG of the 21 runs and their calculation duration per generation, as shown in Table 3 is equal to 0.9, showing a high positive correlation between these two variables. This correlation score clearly shows that by increasing the size of NoG, either by increasing the 3D model element number, initial duration range, or population size, the completion of calculation will take a longer time for each population. More correlation coefficient calculations are shown in Table 4.

Table 4 shows how the generation parameters are interacting with the calculation process. As seen in that table, NoG has an extreme (>90%) positive impact on the generation calculation time and minor (<40%) impact on the entire calculation of the runs. On the other hand, the number of elements of the input 3D model has moderate (>40% & <90%) positive impact on the calculation rounds and durations. Population size and average initial duration of the runs have very similar impact on the calculation process. These two both have minor negative impact on calculation rounds as well as total calculation time, beside slightly higher but positive impact on the each generation calculation duration.

Furthermore, the calculation duration for each generation is increased almost on the same scale of increase in NoG. The calculation duration is also increased around half the scale of increase in either number of 3D model elements, population size, or average duration of the initial population (as the user inputs). On the other hand, the number of total generation rounds to complete the schedule generation process increased in half the scale of increase in number of 3D model elements. Also it is decreased in the third scale of increase in either population size or average initial duration. As can be seen in Table 4, the increase or decrease in NoG would not affect the total number of calculation rounds, but its increase would increase the total calculation duration by the scale of 30%. The total project duration is increased by the factor of 76% of the scale of increase in number of 3D model elements. In the meantime, it decreased by 10% of the increase scale of the population size and average initial duration.

As a summary for Table 4, the calculated correlations show how the GA parameters can impact the needed calculations for the desired outcome. These correlation values can be used for finding the optimized values for the GA parameters in further researches. Fig. 12 shows the outcome plot of all the 21 different experiments. As can be seen in both regular and zoomed views, the experiments merged to the score of 100% as the objective of this research. Some of these runs reach the final score much faster than the others, as described earlier in this paper (see Table 4 and its descriptions). Generally, the similar pattern in all the runs shows initial sharper increase in the constructability score that become gentler as the elements are getting scheduled.

The developed tool for this methodology not only generates stable construction schedules for the project, it also shows the 4-dimensional representation of the construction sequence, illustrating the completion of the 3D model in the time spans. This type of outcome, besides showing how the project is supposed to be built, can be used to evaluate the project schedule for stability and constructability too. Fig. 13 shows eight screen-shots of the generated 4D construction sequence animation for one of the designed experiments (Moderate, 50G, 1ST).

6. Conclusion and future work

The authors showed that the previous researches were focused on enhancing and optimizing the project schedules that were already generated. In this research, the authors proposed a new usage for the Genetic Algorithm for developing and generating construction schedules using the geometric information of a project from its BIM. As shown in the methodology, the proposed algorithm uses the geometrical information in a 3D model to understand stability rules and store the information in a Directed DSM. Then these information is used as the basis for the GA Fitness function to maximize the constructability of construction schedules (each defined as a separate genome). Iterations in the GA calculation shows that even in different sets of input parameters the results always merge to the maximum constructability score (100%) from randomly generated schedules, proving that the methodology is working as expected.

As it has been shown in the result section, all the different inputs to the proposed algorithm merged to reach completely constructible project schedules. Although the runs needed different calculation times based on their inputs, the performed experimental design proved that this method can produce 100% constructible schedules. As mentioned earlier in this paper, the main objective was to retrieve construction project schedules from inherent geometry information embedded in BIM of the project. This objective has been proved to be achievable through this methodology.

Therefore, the main contribution of this research is to introduce and define all the needed functions for the Genetic Algorithm so that it can develop and generate construction schedules for a given BIM of a project. This paper proved the concept of using embedded geometrical information from structural BIM to develop constructible sequence of element installations. Later development of this research methodology can be useful to do the following list of tasks for enhancing construction related processes (Faghihi et al., 2014b):

- Scheduling Learning Tool (e.g. students can learn the impacts of element placement on the schedule and vice versa)
- Customized Schedule Template (i.e. planning expert can start improving the output of this methodology as a basis for developing project schedule)
- 3D Pareto Front (i.e. using the fitness function scores from different objectives)
- Workaround Solutions (e.g. revise the schedule in case of lacking material)
- Material Ordering and Dumping (i.e. optimum time and unloading place for required materials)
- Built Sequence Policies (i.e. defining prioritized direction and ordering of element installations)
- Installation Starting Point (e.g. installation based on crane location or site entrance)
- and more.
To make this methodology more useful for future projects and industrial and educational use, different element types (such as walls, doors, windows, HVAC, pipeline, pumps, etc.) should be added to the MoCC creation approach. Adding these new types of elements to the MoCC creating algorithm will enable the application to handle oil and gas projects, which have inherent difficulties to develop project network and schedule. In addition to extending the detection of more element types, different objectives should be added to the constructability objective mentioned in this paper. Other objectives could lead the outcome of the methodology (project schedules) to be more optimized in cost, time, and even workability in the real job-site. Adding all of these improvements to the proposed algorithm, enables it to produce semi-perfect project schedules in terms of reducing construction duration, labor cost, and site mobilization and workability of the construction processes.

After future extensions of the current work, the outcome can be tested to see the benefits of this methodology of automatic project scheduling from the project BIM in an educational environment. To do this evaluation, the previously proven tool of predicting future project schedules, but also develop them from the scratch. Having this proof of concept, other researches can extend the usability of the projects in educational environment, called Project Management Prediction Market (Damnjanovic, Faghihi, Scott, McTigue, & Reinschmidt, 2013), can be used alongside the extended version of this proposed methodology.

7.1. Future work

This paper showed how GA can be used to not only optimize project schedules, but also develop them from the scratch. Having this proof of concept, other researches can extend the usability of this method in divers and different aspects. As direction for future research studies the following list of works can be listed.

- Modifying the 3D model reader module to detect other 3D elements and projects (not only complete construction models, but other manufacturing 3D models such as cars, circuit boards, etc.) to calculate the MoCC and then generate the assembly sequence for any given 3D model.
- The correlation coefficient values calculated in this article, while combining with other results from more examples, can help designers of other Expert Systems to enhance the structure of their model (for instance, parameters in the Artificial Neural Networks, such as number of layers, number of neurons in each layer, the initial weights on each neuron, the learning algorithm, etc.).
- Performing several complex and extended examples to formulate effects of parameters changes on the schedule structure. (e.g. impacts of 3D model complexity on the resulting schedules and finding optimum number of genes, population size, and initial duration)
- Extending the use of multi objective capability of the GA in a way that the resulting schedules are not only constructible, but also optimized in assembly time, resource needed and cost of assembly.
- Developing an algorithm that can generate the common knowledge rules from the BIM of the model rather than being hard coded in the algorithm.
- Further development of MoCC in a way that it can contain a range from −1 to +1, showing correlations of the dependency between elements.

Appendix A

A.1. Simple step-by-step example

Considering the example in Fig. 1, with five genomes per population, 20% of population as elite members, 10% chance of mutation, and the tentative duration of 5 ± 20% time-unit (i.e. either 4, 5, or 6 time-units), the first population generated by the described GA is as shown in Table 5. This table consists of two main columns: the left column containing the genome number, constructability score, and schedule duration, and the right column containing the genomes. Since there are eight elements in the model shown in Fig. 1, the constructability scores (in form of percentage of the number of elements that have been scheduled based on MoCC) are either 0%, 12.5%, 25%, 37.5%, 50%, 62.5%, 75%, 87.5%, or 100%.

The genomes in the right column of Table 5 are basically the Genome Matrix or MoG (defined in Eq. (2)) that are shown in the form of Eq. (3), but the rows of the matrix are put consecutively one after the other. The dotted lines are the dividers between the rows of the MoG. Because there are eight elements in the 3D model, all of the genomes have eight sections that are divided from each other by the dotted line, as seen in Table 5. Then, each of these eight sections in the genomes contains the same number of zeros and ones as the duration of the genome. For example, the first four digits of the genome number 0 in the first population, as seen in Table 5 (i.e. 0100) is the installation schedule for the first element of the 3D model, column 1:#143, as shown in Fig. 1. These four digits define the installation of the element number 1:#143 to the second time-unit as its second digit is 1 and the rest are 0s.

As mentioned before, the genomes in this paper can be represented as the MoG too. The MoG for the first genome of the first population is shown in Fig. 14.

Given the genome in the form of the matrix, it can be easily interpreted and the scheduling of the installations are much more understandable. As illustrated by Fig. 14, elements number 1, 2 and 3 are scheduled to be installed in the second time-unit while the elements number 4 and 5 are planned to be done in the fourth time-unit. Similarly, the element number 6 is scheduled for the third time-unit, but the remaining two elements, 7 and 8, are not planned to be installed in this sequencing order defined by this genome.

Clearly, the schedule in Fig. 14 has four time-units as represented by four columns in its matrix. The calculated constructability score for this genome (construction schedule of the 3D model) as shown in Table 5 is 62.5%. This score means that 62.5% (5 elements out of 8) are scheduled for installation correctly based on MoCC constraints detected from the model as MoCC (see Fig. 1). These five elements are columns 1:#143, 2:#209, 3:#239, and 4:#26 and the beam 5:#375. The first four columns do not have any installation constraints detected in the MoCC...
and the only beam has columns 1 and 2 as its supporting constraints being scheduled for installation as of fourth time-unit (columns are scheduled in the second time-unit and the beam is scheduled in the fourth time-unit).

All five genomes in the first population were generated based on the random genome generation function described in section “Genome Creation” Performing all the inherent Genetic Algorithm functions (i.e. Elite selection, Crossover function, Mutation function, Selection function) described in “Genetic Algorithm” section of this paper, the next populations will be generated as shown in Tables 6–10, showing from second population to the sixth respectively.

As seen in Table 10, the first genome with the complete score of 100% for its constructability is generated. After this achievement, the proposed algorithm will continue until all the genomes in the population reach the complete 100% score for their constructability. The final population can be seen in Table 11, as the 20th population in this simple run that all the genomes have reach the 100% score. The algorithm has now ended.

References


Table 6
Population number 2.

| 0.875%-5 | 01000100000000000000000001 | 1000001000000000000000000001 |
| 1.75%-4  | 001000000000000000000000001 |
| 2.75%-5  | 000000000000000000000000001 |
| 3.625%-4 | 000000000000000000000000001 |
| 4.50%-5  | 000000000000000000000000001 |

Table 7
Population number 3.

| 0.875%-5 | 01000010000000000000000001 |
| 1.75%-4  | 010000000000000000000000001 |
| 2.75%-5  | 010000000000000000000000001 |
| 3.625%-4 | 010000000000000000000000001 |
| 4.50%-5  | 010000000000000000000000001 |

Table 8
Population number 4.

| 0.875%-5 | 01000000000000000000000001 |
| 1.75%-5  | 010000000000000000000000001 |
| 2.75%-5  | 010000000000000000000000001 |
| 3.75%-5  | 010000000000000000000000001 |
| 4.50%-5  | 010000000000000000000000001 |

Table 9
Population number 5.

| 0.875%-5 | 01000000000000000000000001 |
| 1.875%-5 | 010000000000000000000000001 |
| 2.50%-4  | 010000000000000000000000001 |
| 3.50%-4  | 010000000000000000000000001 |
| 4.50%-5  | 010000000000000000000000001 |

Table 10
Population number 6.

| 0.100%-5 | 01000000000000000000000001 |
| 1.00%-6  | 010000000000000000000000001 |
| 2.00%-6  | 010000000000000000000000001 |
| 3.00%-5  | 010000000000000000000000001 |
| 4.00%-6  | 010000000000000000000000001 |

Table 11
Population number 20.

| 0.100%-6 | 01000000000000000000000001 |
| 1.00%-6  | 010000000000000000000000001 |
| 2.00%-6  | 010000000000000000000000001 |
| 3.00%-5  | 010000000000000000000000001 |
| 4.00%-6  | 010000000000000000000000001 |