Optimization of the Cost of Repetitive Construction Projects Using Computerized Iteration Method

Aqeel abdulhassan*, *Ruqaya A*. Muter, *Ali* Majdi, *Sabah Mohammed* Abd Mosehab, *Fatin Hashim* Kareem and *Israa Mohsin Kadhim* Al-Janabi

Al-Mustaqbal University, 51001 Hillah, Babil, Iraq

Abstract. This study introduces an iterative scheduling method that combines two approaches for managing repetitive construction projects: the Critical Path Method (CPM) and the Repetitive Scheduling Method (RSM). The primary objective of this study is to demonstrate how optimization techniques can be applied to minimize the cost of construction projects within a defined range, spanning from the shortest to the longest possible project durations. In the shortest project duration (as determined by CPM), all activities are allocated idle times based on precedence constraints, while in the longest duration (as determined by RSM), there is no idle time allocated. To calculate the optimal schedule, a computerized iterative method specifically designed for this purpose considers all possible combinations of activities with and without idle time. The optimum schedule is the one that minimizes the total project cost. The study reveals that by using an Excel spreadsheet, it is feasible to deterministically optimize the cost of repetitive construction projects, achieving the minimum cost. This minimization process can also be implemented as a Python application. Notably, this proposed system provides multiple optimal solutions, enabling managers to select the most suitable one. This advantage distin-guishes it from conventional methods, such as genetic algorithms and other optimization techniques. However, there are some limitations when applying this application, one of which is the maximum capacity available to run the application.

1 Introduction

This study deals with the optimization of the cost of repetitive construction projects. Many methods have been used to optimize the cost in such projects using a similar approach which is the balancing of cost of idle times with the overhead costs of the projects. The technique of the balancing is the main subject that differs in each method. Some methods are built on a statistical basis and others are graphical. This study introduce a method that can solve the problem of balancing with simple algebraic steps. These steps represents a mid-way between two major methods in scheduling: Optimization; Critical Path Method (CPM) and Repetitive Construction Method (RSM). The algorithm gained by the method is utilized in constructing an application by using Python.

2 Literature Reviewing

The methods of optimization of repetitive construction projects can be reviewed briefly as following : (This is based on the work of Mostafa Sakr et al 2021) [1]

2.1 Mathematical Methods

Mathematical methods require explicit formulation Which is hard and time-consuming. They are: (1) linear programming, (2) non-linear programming, (3) dynamic programming, and (4) constraint programming.

2.2 Heuristic methods

Heuristic methods depend on putting rules that can be based on the experience of problem-solving. As an example is utilizing different combinations of the decision variables for the activities and the combination that will lead to minimum duration and cost will be selected.

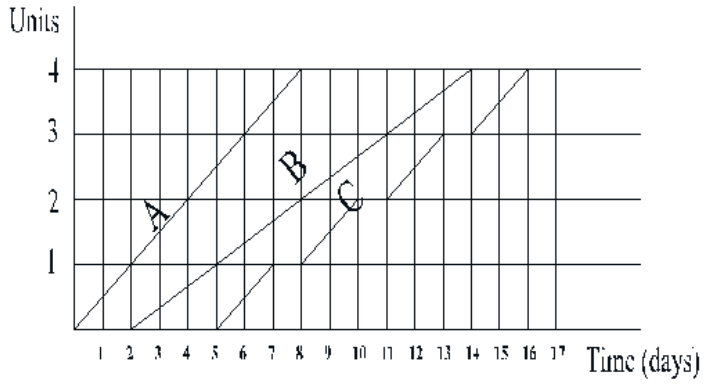
* Corresponding author: aqeelabdulhassan@uomus.edu.iq

2.3 Meta-heuristic methods

These are used to solve large scale problems. These methods perform iterative computation against a criteria to find the optimal or near-optimal solutions. Examples are: (1) genetic algorithm (GA), (2) particle swarm optimization (PSO), and (3) ant colony algorithm (ACA).

3 Idle Times in Repetitive Construction Projects

Repetitive construction projects differ from discrete pro-jects in that they encompass the constraint of continuity in addition to the precedence constraint. Due to the na-ture of repetitive projects, certain activities may have idle times for resources, during which resources must wait until the next activity becomes available or ready to begin [2][3]. For example, as depicted in Figure (1), workers involved in activity C must wait after completing activity B, illustrating this situation.



3.1 Example

A repetitive project consists of six activities is shown in table (1).

Table 1 Six activities construction repetitive project.

Activities	Duration(days)	Idle time cost$
A	2	3
B	1	5
C	3	2
D	1	8
E	4	4
F	3	7

Number of units (n) is 4 and the indirect cost (IC) is ($8) per day.
Figure (2) shows the project when it is scheduled with CPM with idle times taking place. Figure (3) shows the same project when it is scheduled with RSM and no idle time appears.

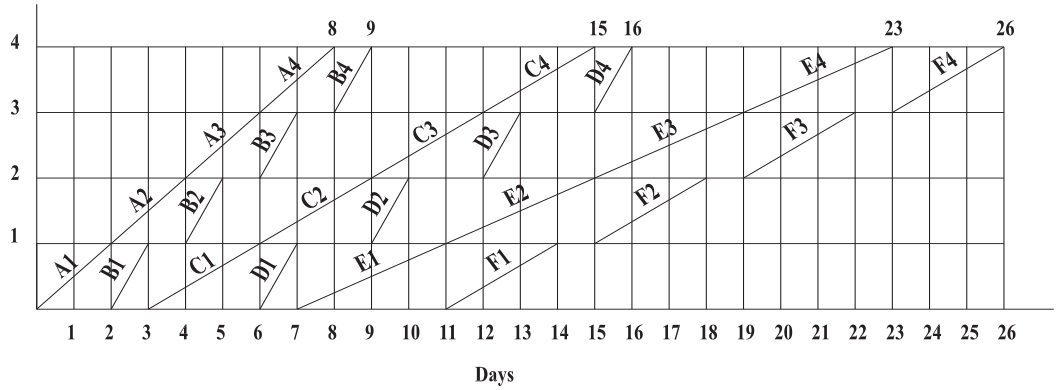


Fig .2 The project scheduled with CPM

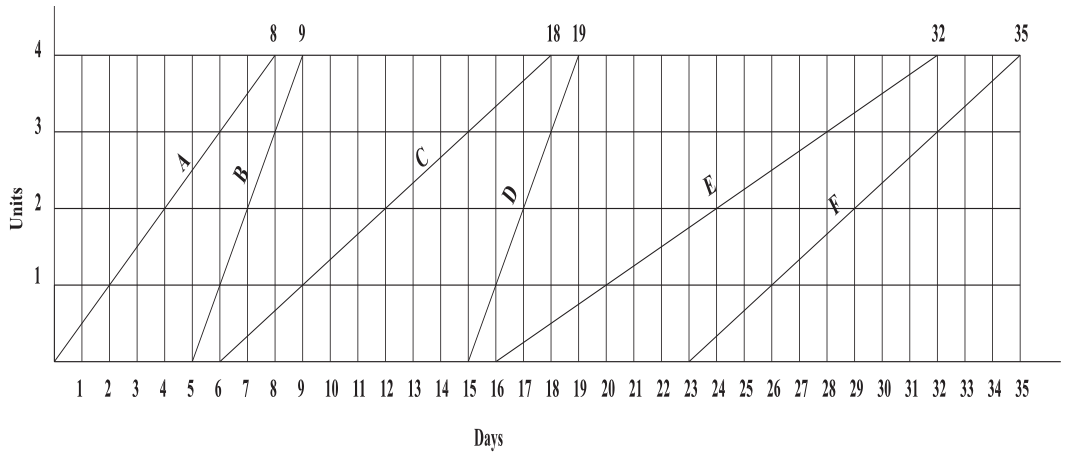


Fig. 3 The project scheduled with RSM

Idle time for any activity can be estimated (if it exists) from the following formula:

$$I(i) = d(i-1) - d(i) + I(i-1) \text{ , if } I(i) < 0 \text{ take } I(i) = 0 \qquad (1)$$

Where:

$d(i)$ = duration of current activity
$d(i-1)$ = duration of previous activity
$I(i-1)$ = idle time of previous activity
$I(i)$ = idle time of current activity

It is obvious that the first activity in any project has no idle time. The rest idle times can be calculated one by one. Applying the equation on the given example the result will be as following:

$I(A) = 0$
$I(B) = d(A) - d(B) + I(A) = 2 - 1 + 0 = 1$
$I(C) = d(B) - d(C) + I(B) = 1 - 3 + 1 = -1$, take $I(C) = 0$
$I(D) = d(C) - d(D) + I(C) = 3 - 1 + 0 = 2$
$I(E) = d(D) - d(E) + I(D) = 1 - 4 + 2 = -1$, take $I(E) = 0$
$I(F) = d(E) - d(F) + I(E) = 4 - 3 + 0 = 1$

Total idle time costs (IC) for four units will be equals to (n-1) times total cost for one unit:
$3\times[\ 0\times3 + 1\times5 + 0\times2 + 2\times8 + 0\times4 + 1\times7] = \$84$
Cost resulting from increasing the total duration of the project from 26 days to 35 days is : $9\times8 = \$72$.
So, it is preferable here to eliminate all idle times and adopt RSM scheduling. It becomes evident that a general approach is required to achieve the optimal scenario. A range of alternative solutions may also offer answers to this problem. Therefore, it is essential to explore all possibilities. Initially, the commencement and conclusion of each activity should be represented using a set of equations [4],[5].

3.2 CPM Mode

Start of each activity:

$S(i) = S(i-1) + d(i-1)$
Thus for previous situation :
$S(A) = 0$ *first activity*
$S(B) = S(A) + d(A) = 0 + 2 = 2$
$S(C) = S(B) + d(B) = 2 + 1 = 3$
$S(D) = S(C) + d(C) = 3 + 3 = 6$
$S(E) = S(D) + d(D) = 6 + 1 = 7$
$S(F) = S(E) + d(E) = 7 + 4 = 11$

These are shown in figure (2)

Finish of each activity:

$F(i) = S(i) + (n \times d(i)) + ((n-1) \times I(i))$
$F(A) = 0 + 4 \times 2 + 3 \times 0 = 8$
$F(B) = 2 + 4 \times 1 + 3 \times 1 = 9$
$F(C) = 3 + 4 \times 3 + 3 \times 0 = 15$
$F(D) = 6 + 4 \times 1 + 3 \times 2 = 16$
$F(E) = 7 + 4 \times 4 + 3 \times 0 = 23$
$F(F) = 11 + 4 \times 3 + 3 \times 1 = 26$
These are shown also in figure (2)

3.3 RSM Mode

Start of each activity :
$S(i) = S(i-1) + d(i-1) + (n-1) \times I(i)$
$S(A) = 0$ *first activity*
$S(B) = 0 + 2 + 3 \times 1 = 5$
$S(C) = 5 + 1 + 3 \times 0 = 6$
$S(D) = 6 + 3 + 3 \times 2 = 15$
$S(E) = 15 + 1 + 3 \times 0 = 16$
$S(F) = 16 + 4 + 3 \times 1 = 23$
These are shown in figure (3)
Finish of each activity :
$F(i) = S(i-1) + d(i-1) + n \times d(i) + (n-1) \times I(i)$
$F(A) = 0 + 0 + 4 \times 2 + 3 \times 0 = 8$
$F(B) = 0 + 2 + 4 \times 1 + 3 \times 1 = 9$
$F(C) = 5 + 1 + 4 \times 3 + 3 \times 0 = 18$
$F(D) = 6 + 3 + 4 \times 1 + 3 \times 2 = 19$
$F(E) = 15 + 1 + 4 \times 4 + 3 \times 0 = 32$
$F(F) = 16 + 4 + 4 \times 3 + 3 \times 1 = 35$
These are shown also in figure (3)

Now, if idle time of activity B is eliminated, the new schedule should be as in figure (4).

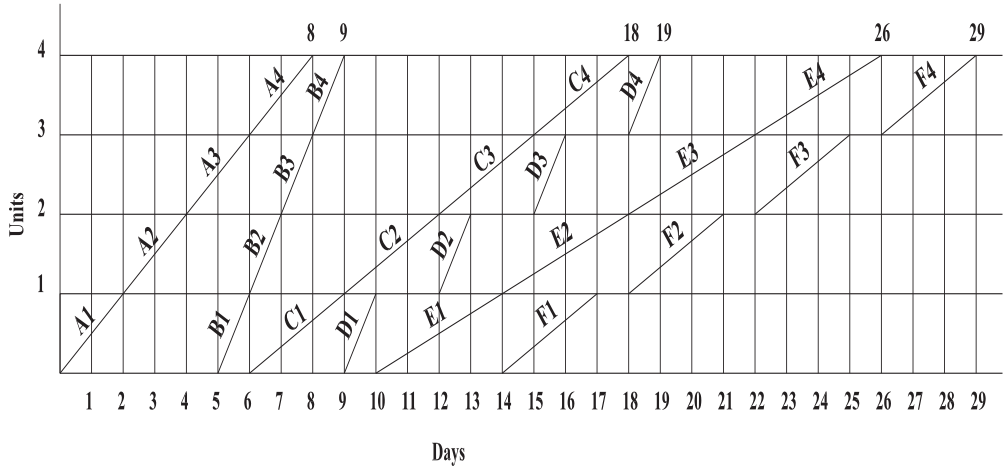


Fig. 4 The project schedule if idle times of activity B are eliminated

If idle time of activity D is eliminated, the resulting schedule should be as in figure (5) . The similar case for activity F is shown in figure (6). Figure (7) shows the case of combination B & D , figure (8) shows B & F while figure (9) shows case of eliminating idle times of D & F . Table (2) gives summary for all possible combinations in cases resulting from eliminating of idle times.

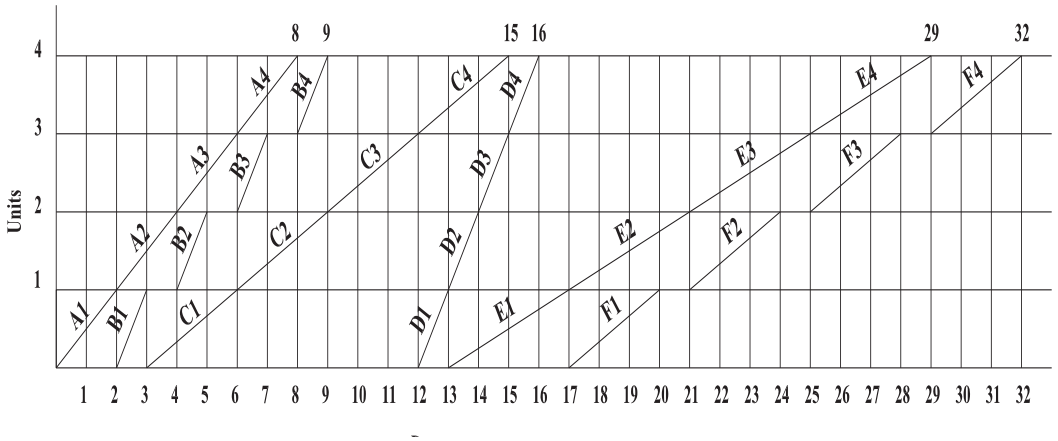


Fig.5 The project schedule if idle times of activity D are eliminated

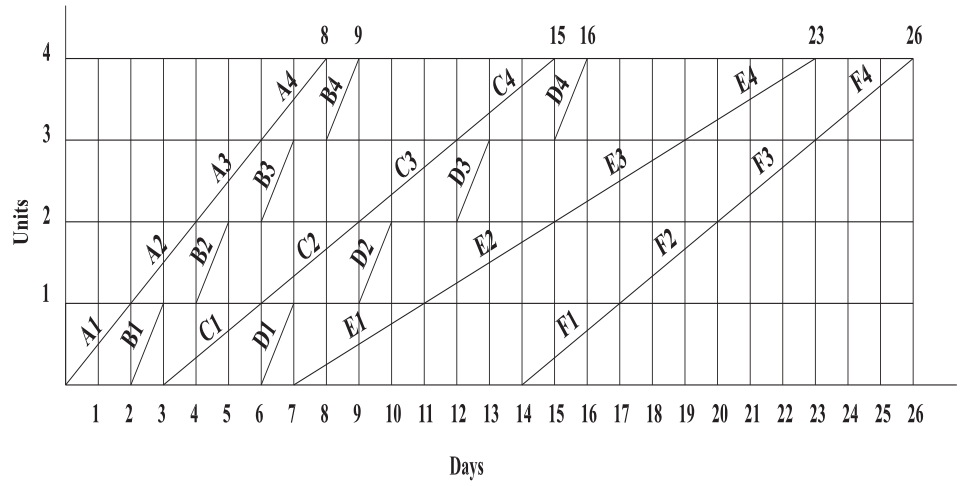


Fig.6 The project schedule if idle times of activity F are eliminated

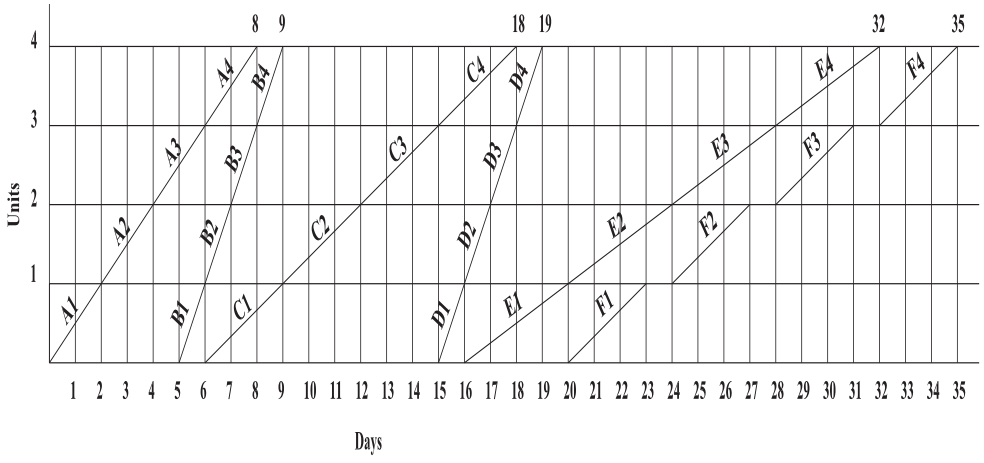


Fig.7 The project schedule if idle times of activities B and D are eliminated

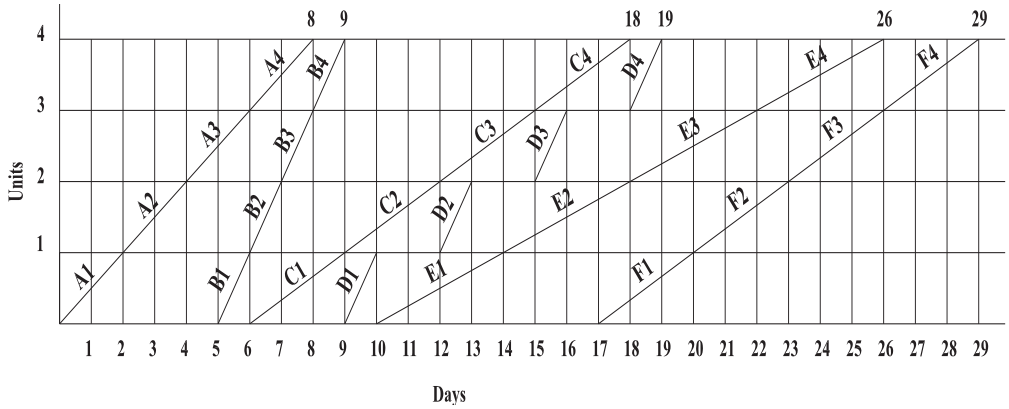


Fig.8 The project schedule if idle times of activities B and F are eliminated

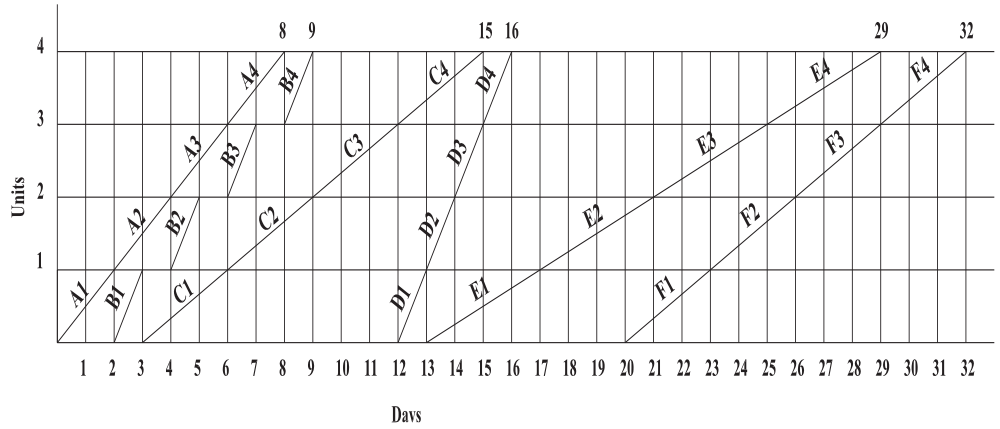


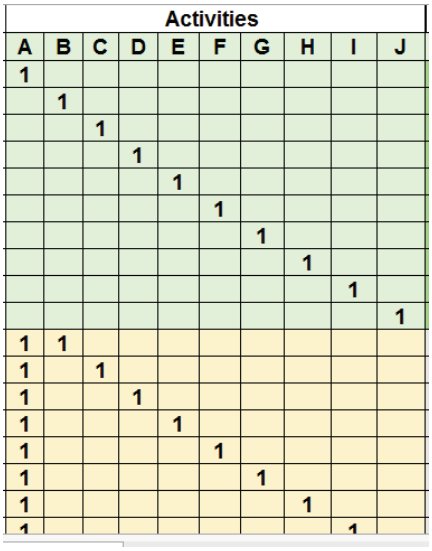
Fig.9 The project schedule if idle times of activities D and F are eliminated

Table 2. Summary of all possible combinations

Case	Duration (days)	Idle time cost$	Overhead cost$	Total cost$
CPM Mode	26	84	208	292
RSM Mode	35	0	280	280
I.t of (B) eliminated	29	69	232	301
I.t of (D) eliminated	32	36	256	292
I.t of (F) eliminated	26	63	208	271
I.t of (B & D) eliminated	35	21	280	301
I.t of (B & F) eliminated	29	48	232	280

4 The structure of algorithm

Excel program is used to make the process of optimization more easier . The basis of the algorithm is to build a matrix from which the selection of eliminated idle times can be made . This matrix is then the ones' matrix . It looks like that in figure (10) .[6].



Activities									
A	B	C	D	E	F	G	H	I	J
1									
	1								
		1							
			1						
				1					
					1				
						1			
							1		
								1	
									1
1	1								
1		1							
1			1						
1				1					
1					1				
1						1			
1							1		
1								1	

Fig.10 The ones Matrix

In each raw of such matrix a new combination of activities regarding their idle times is introduced . The idle time of each activity may vary according not only to equation (1) but also there is another possibility that must be taken. The condition function of an activity to find its idle time is (using Excel terms) :

If there is number 1 in the corresponding cell of ones' matrix to the activity then:
First term : I'
I' = 0 for the first activity. For other activities
If D(i-1)-D(i)+I'(i-1)<0 then I'(i) =0
Otherwise : I'(i) =D(i-1)-D(i)+I'(i-1)
Second term : I"
I' = 0 for the first activity. For other activities:
If there is number 1 in the corresponding cell of ones' matrix to the activity then: I"(i) =0
Otherwise :
If D(i-1)-D(i)+I"(i-1)<0 then I"(i) =0
Otherwise : I'(i) =D(i-1)-D(i)+I'(i-1)
D is the activity duration .

After getting the real idle time for each activity (I") , the rest calculation is made in direct manner : find start and finish , find idle time cost , summing the idle time costs and find the total indirect cost and lastly the total cost [7],[8].

5 Translating Algorithm into an Application

The process of optimization can be translated to be an application such that one in figure (11) which is made using Python language . However there is a limitation regarding the number of activities that can be treated and hence in the capacity of the application . The large number of activities leads to very large calculating process that need large capacity computers . Following the main code that was used in python, The proposed system was designed by using Python3.10, in a computer with processor intel®core™ 2.30 GHZ, RAM 4.00 GB.

```
class Cell(QWidget):
    def init_from_item(self, row, column, is_min):
        self.setProperty('dataRow', row)
        self.setProperty('dataColumn', column)
        self.setProperty('isMin', str(False))
class NTDelegate(QStyledItemDelegate):
    def __init__(self, *a, **kwargs):
        super(NTDelegate, self).__init__(*a)
        self.min_max = kwargs["min_max"]
        self.cell = Cell(self.parent())
```

```
        def paint(self, painter, option, index):
            min_total_costs = self.min_max["min_total_costs"]
            max_total_costs = self.min_max["max_total_costs"]
            row = index.row()
            column = index.column()
            is_min = False
            for key in min_total_costs:
                if row == key:
                    is_min = True
            self.cell.init_from_item(row, column, is_min)
            self.initStyleOption(option, index)
            style = option.widget.style() if option.widget else
    QApplication.style()
            style.polish(self.cell)
    style.drawControl(QStyle.CE_ItemViewItem, option, painter, self.cell)
    class TableView(QTableWidget):
        def __init__(self, table_data, min_max):
            self.min_total_costs = min_max["min_total_costs"]
            self.min_cost_value = min_max["min_cost_value"]
            columns = len(table_data)
            rows = len(self.min_total_costs)
            QTableWidget.__init__(self, rows, columns)
     self.setItemDelegate(NTDelegate(self, min_max=min_max))
            self.data = table_data
            self.set_data()
            for i in range(0, int(columns)):
                if i >= int(columns) - 3:
                    self.setColumnWidth(i, 120)
                else:
                    self.setColumnWidth(i, 40)
        def custom_set_item(self, row, column, value):
            new_item = QTableWidgetItem(value)
            self.setItem(row, column, new_item)
        def get_item(self, row, column):
            return self.item(row, column).text()
        def set_data(self):
            hor_headers = []
            for n, key in enumerate(self.data.keys()):
                hor_headers.append(key)
                cost_key = "Total Cost"
                row = 0
                for m, item in enumerate(self.data[key]):
                    if int(self.data[cost_key][m]) == self.min_cost_value:
```

```
                    new_item = QTableWidgetItem(item)
                    self.setItem(row, n, new_item)
                    row += 1 self.setHorizontalHeaderLabels(hor_headers)
class MainForm(QMainWindow):
    def __init__(self, data, units, activities,  **kwargs):
        super().__init__()   self.setAttribute(Qt.WA_DeleteOnClose)
      self.setFocusPolicy(Qt.TabFocus)
        scroll = QScrollArea(self)
       scroll.setWidgetResizable(True)
        self.central_widget = QStackedWidget()
        units_label = RegularLabel('Units : ' + units)
        activities_label = RegularLabel('Activities : ' + activities)
        min_max = data["min_max"]
        table_data = data["table_data"]
        table = TableView(table_data, min_max)
        table.setObjectName("main-table")

        body = QWidget()
        layout = QVBoxLayout()
        first_line = QHBoxLayout()
        first_line_widget = QWidget()
      first_line.addWidget(units_label)
        first_line.addWidget(activities_label)
        first_line_widget.setLayout(first_line)
        layout.addWidget(first_line_widget)
        layout.addWidget(table)
        body.setLayout(layout)
        self.central_widget.addWidget(body)
scroll.setWidget(self.central_widget)
        self.setCentralWidget(scroll)
        self.toolbar = self.addToolBar('&Main')
self.toolbar.setLayoutDirection(Qt.RightToLeft)
self.toolbar.toggleViewAction().setEnabled(False)
self.setContextMenuPolicy(Qt.PreventContextMenu)
        self.toolbar.addActions(btn for btn in ToolBar(self).Buttons)
        self.statusBar().showMessage("Concurrent Activities for
construction projects")    self.toolbar.installEventFilter(self)
        self.installEventFilter(self)
      self.setObjectName("main_window")
```

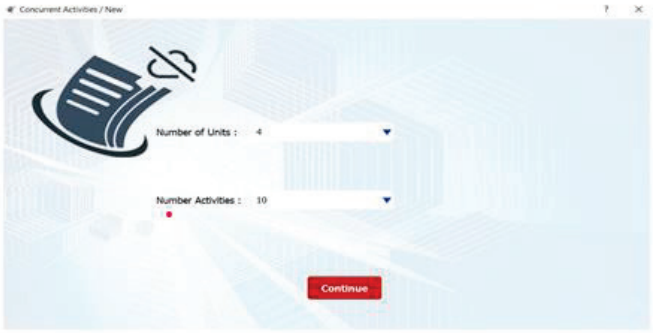


Fig.11a Application: Entry of units and activities

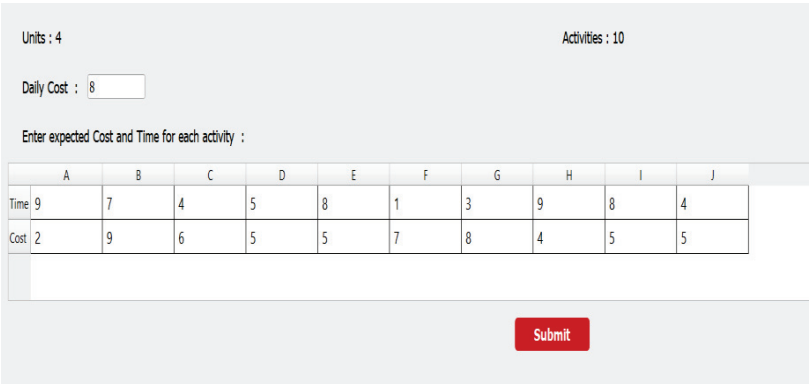
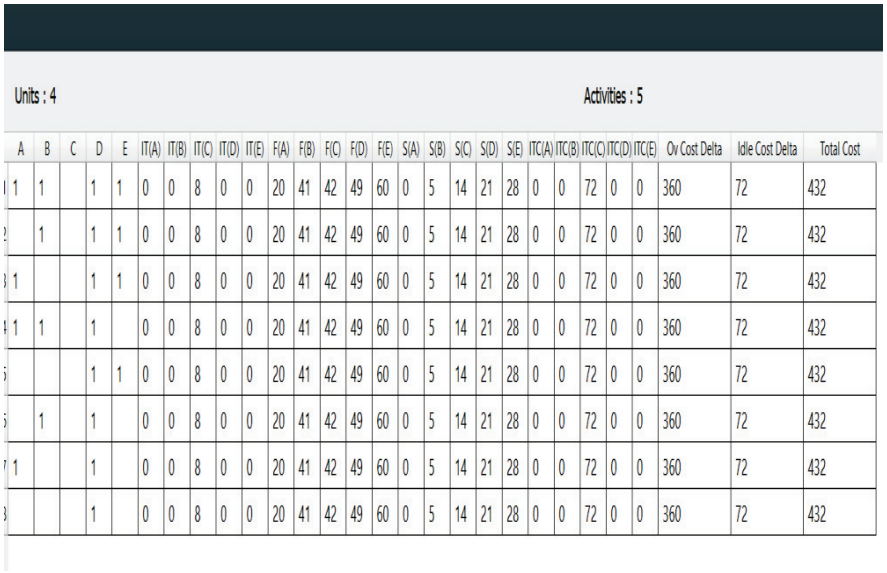


Fig.11b Application: Entry of indirect cost and idle time costs



A	B	C	D	E	IT(A)	IT(B)	IT(C)	IT(D)	IT(E)	F(A)	F(B)	F(C)	F(D)	F(E)	S(A)	S(B)	S(C)	S(D)	S(E)	ITC(A)	ITC(B)	ITC(C)	ITC(D)	ITC(E)	Ov Cost Delta	Idle Cost Delta	Total Cost
1	1		1	1	0	0	8	0	0	20	41	42	49	60	0	5	14	21	28	0	0	72	0	0	360	72	432
	1		1	1	0	0	8	0	0	20	41	42	49	60	0	5	14	21	28	0	0	72	0	0	360	72	432
1			1	1	0	0	8	0	0	20	41	42	49	60	0	5	14	21	28	0	0	72	0	0	360	72	432
1	1		1		0	0	8	0	0	20	41	42	49	60	0	5	14	21	28	0	0	72	0	0	360	72	432
			1	1	0	0	8	0	0	20	41	42	49	60	0	5	14	21	28	0	0	72	0	0	360	72	432
	1		1		0	0	8	0	0	20	41	42	49	60	0	5	14	21	28	0	0	72	0	0	360	72	432
1			1		0	0	8	0	0	20	41	42	49	60	0	5	14	21	28	0	0	72	0	0	360	72	432
			1		0	0	8	0	0	20	41	42	49	60	0	5	14	21	28	0	0	72	0	0	360	72	432

Fig.12 Application: Outputs (combinations with minimum total costs)

6 Conclusions

1. Using a spread sheet in Excel, it is possible to optimize the repetitive construction project cost in deterministic manner and the minimum cost can be obtained.
2. The process of minimization can be translated to be an application such that one in figure (10).

3. This proposed system provides all possible optimal results, allowing managers the opportunity to select the most suitable solutions. This advantage sets it apart from conventional methods, such as genetic algorithms and other optimization algorithms.
4. However there are some limitations in applying this application from them the maximum capacity that may be available to run the application.

References

1. Mostafa Sakr et al , Optimization of Repetitive Projects Scheduling in Construction Analysis for the State – of – Art Methods , Journal Of Al-Azhar University Engineering Sector , 2021
2. Harris RB, Ioannou PG. Repetitive scheduling method. Center For Construction Engineering And Management, Michigan. 1998 Nov.
3. Vanhoucke M. Work continuity constraints in project scheduling. Journal of Construction Engineering and Management. 2006 Jan;132(1):14-25.
4. Bennett FL. The management of construction: A pro-ject lifecycle approach. Routledge; 2007 Jun 1.
5. Uher TE. Programming and Scheduling Techniques (Construction Management). University of New South Wales Press (UNSW); 2003.
6. Agrama FA. Linear projects scheduling using spread-sheets features. Alexandria Engineering Journal. 2011 Jun 1;50(2):179-85.
7. Mubarak SA. Construction project scheduling and control. John Wiley & Sons; 2015 Mar 23.
8. Sullivan A, Harris FC. Delays on large construction projects. International journal of operations & production management. 1986 Jan 1;6(1):25-33.