1. Introduction

Repetitive construction projects, by definition, require the construction crews to repet the same work across a number of units. As such, the key for efficient scheduling of repetitive projects is allowing the crews to move from one unit to the other with minimal interruptions. This has numerous benefits such as establishing a good work momentum which in turn leads to time and cost savings [1, 2].

To consider the minimal interruptions objective and work continuity constraints, many repetitive scheduling models have been developed since the 1960s. Examples include Vertical Production method [3], Line of Balance, LOB [4], Flowlines [5], Linear Scheduling Model [6], Repetitive Scheduling Method [7], Productivity Scheduling Method [8], Location-Based Management and Scheduling (LBMS) [9], and others [e.g., 10, 11]. Alas, most of the presented scheduling models focus only on the case of identical units. Although there are new methods developed to address the more general case of non-repetitive units [12-14], repetitive scheduling still has challenges that are yet to be addressed and require careful planning to determine the necessary number of crews to achieve the project deadline while maintaining work continuity.

1.1. Repetitive Scheduling as a game of Tetris

Repetitive scheduling can be seen as a visual process where the activities can take on different geometries based on the number of crews used and their arrangement (Figure 1a). It can be seen in Figure 1b-d that having time gaps in the schedule leads to extended project durations. As such, repetitive scheduling can be viewed as a game of Tetris where the successive activities represent the blocks that need to be arranged next to each other with minimal gaps to achieve minimal project duration. This is not an easy endeavour in the case of complex construction projects that involve a large number of tasks.
and a variety of complex constraints. Such complexity often forces project managers to use more costly time-cost trade-off solutions to meet the necessary deadlines. As such, this paper proposes a combination of mathematical computations and scheduling algorithms to systematically produce tightly-packed schedules that meet the project necessary deadlines while maintaining an efficient use of the available crews and project resources.

| Geometry 1 | Geometry 2 | Geometry 3 | Geometry 4 | Geometry 5 |
|------------|------------|------------|------------|------------|
| One or more crews arranged in shifted mode | Shifted crew arrangement, with interruption time | Parallel crew arrangement, with no. of crews < Units | Parallel crew arrangement, with no. of crews = units | Parallel crew arrangement, with interruption time |

---

**Figure 1**: Effect of geometric arrangement on schedule duration

2. **Current CPM/LOB Calculations and Persistent Challenges**

The first step in existing CPM/LOB calculations is use conventional CPM formulae to calculate the time needed to finish one unit \(T_1\). Then, since the all the units \(N\) need to be completed before the deadline \(D_L\), the rate of delivery of the units \(R\) is calculated using Equation 1. Equation 1 incorporates the total float of the activity \(TF\) as an adjustment factor that reduces the delivery rate of non-critical activities and thus requires less crews. Accordingly, the number of required crews is calculated based on required rate \(R\) and the duration of the activity \(D\) using Equation 2, and then the rate of delivery is adjusted based on the new number of crews after rounding (because having fractions of a crew is impractical) through Equation 3. An example can be seen in Figure 2.
Task i desired rate \( (R_i) = \frac{N-1}{(D_L - T_1 + T_F)} \)

Task necessary crews \( (C_i) = \text{Roundup} \left( D_i \times R_i \right) \)

Task actual rate \( (R_i) = C_i / D_i \)

Figure 2: CPM/LOB Analysis of the tasks’ required shifted crews to meet deadline

Repetitive schedules, however, often exhibit deadline violations even when the necessary computations are applied. This is because of the multiple reasons such as rounding of crews, rounding of start times, crew availability limitations, or other reasons that change the geometry of the activity/crew assignment and thus introduces schedule gaps that lead to project duration extensions. As such, this paper presents novel computations and algorithms to produce tightly packed schedules that meet the necessary deadlines. The contributions of this paper are threefold: (1) propose a formulation for designed interruptions that adjust the task geometry and delivery rate to reduce the schedule gaps by synchronizing the delivery rates of all the tasks; (2) introduce a validation loop to avoid deadline violations in relaxed schedules; and (3) propose a novel crew assignment algorithm (First-Come-First-Serve) that reduces the schedule gaps by treating the schedule as a game of Tetris on the activity level.

3. Formulation for Designed Interruption

As seen in Figures 1b and 2a, a schedule that has no time gaps is a schedule whose tasks operate with the same rate of delivery. However, this is not always practical because of multiple reasons such as crew availability constraints and/or rounding of crews (see Equation 2). A more vivid example can be seen in Figure 3a, where activity B has a higher delivery rate than the other activities which creates huge schedule gaps. It can then be seen in Figure 3b that deliberately interrupting activity B to bring its overall rate of delivery closer to its counterparts reduces the schedule gaps which in turn reduces the overall project duration.

Figure 3: Effect of Unsynchronized Delivery Rates
To present a formulation for achieving better synchronization among the activities’ delivery rates through designed interruption, the case in Figure 4 is used, where a faster activity (B) follows a slow one (A). As schematically shown in Figure 4a, the start-time for activity B in case of no interruptions (STO_B) is shown. Assuming one interruption at N/2 is sufficient, the interruption time that allows the bottom half of activity B to start earlier is shown on the figure. As shown, STO_B = Y + Duration of B + Interruption time, therefore:

\[ \text{Interruption Time} = \text{STO of B} - Y \times (\text{STO of B for N/2 units}) - \text{Duration of B} = \]

\[ = (N - 1) \left( \frac{1}{R_A} - \frac{1}{R_B} \right) - ((N - 1)/2) \left( \frac{1}{R_A} - \frac{1}{R_B} \right) - D_B \]

Thus, applying the interruption, the bottom part activity B (with N/2 units) starts with an offset of Y:

\[ \text{Start-Time Offset of B with interruption} = Y = ((N - 1)/2) \left( \frac{1}{R_A} - \frac{1}{R_B} \right) \]

Using this formulation, it can be seen that the lower part of activity B started early, thus, changing the average delivery rate of B (dashed line in Figure 4b) to a lower rate than the original rate of B, and becoming more closer in its rate to its predecessor (A). This average modified rate of B can also be formulated, based on the presentation in Figure 4b, as follows:

\[ \text{Average rate of B with interruption} = \left( \frac{N - 1}{T_A - Y} \right) = \left( \frac{N - 1}{R_A - Y} \right) \]

4. Deadline Checking Loop

Consider the example in Figure 5, with a deadline of 20 days and 5 repetitive units. Figure 5 includes the standard CPM calculations using Equations 1-3 as well as the produced schedule. It can be seen from the schedule at the bottom of Figure 4 that the standardized CPM/LOB formulations produced a schedule that is 24 days long. This is unacceptable as it violates the original 20-day deadline. While introducing interruptions using Equations 4-6 does help reduce the project duration, it does not reduce it enough to meet the deadline (unless an interruption is placed after every unit, which defeats the work-continuity purpose of using repetitive scheduling techniques). It can also be seen that the actual duration of the first unit is 12 days, not 8 as calculated earlier.
Computationally, the calculation and assignment of the different crews in repetitive schedules is controlled by one main parameter which is the rate of delivery (R), which is a function of the deadline and CPM duration as seen in Equation 1. Based on the example shown in Figure 5, the duration of one unit \( T_1 \) was higher than what was originally calculated. Based on these observations, it can be concluded that using a higher value for \( T_1 \) would lead to improved CPM/LOB scheduling procedures and prevent deadline violations. However, having higher delivery rate corresponds to the need for more crews. Hence, there is a need to strike a balance between avoiding deadline violations as well as avoiding the overuse of available resources.

As such, this paper proposes a simple loop that iterates through different values of \( T_1 \) to see if the deadline is met. This loop is introduced in Figure 6, where the difference between the deadline (DL) and \( T_1 \) is divided into five intervals of length G. Hence, if the schedule produced using the original value of \( T_1 \) is violating the deadline, \( T_1 \) is increased by G days and the LOB calculations are reperformed. The loop terminates once a schedule is reached that satisfies the project deadline.

Applying the deadline checking loop on the example introduced in Figure 6, G is calculated to be \((20 - 8)/5 = 2.4\) days. Since the schedule created using the original \( T_1 \) (8 days) did not satisfy the deadline, \( T_1 \) is increased by G days and a new schedule is created using the new \( T_1 \) (\( T_1 = 8 + 2.4 = 10.4 \) days). Based on the new values, a new schedule is created that happened to satisfy the original 20-day deadline as seen in Figure 6. Hence, the loop terminates. Otherwise, the LOB scheduling calculations would have been performed with \( T_1 = 10.4 + 2.4 = 12.8 \) days.

### Table

| Duration | Desired Rate, Eq.1, R | Necessary crews, Eq.2, C = D x R | Actual Rate, R = C / D | Shift 1/R |
|----------|-----------------------|----------------------------------|------------------------|-----------|
| A        | 3                     | 4 \((20-8) = 0.33\)               | Roundup \((3 \times 0.33) = 1\) | \(1/3 = 0.33\) | 3         |
| B        | 2                     | 4 \((20-8) = 0.33\)               | Roundup \((2 \times 0.33) = 1\) | \(1/2 = 0.50\) | 2         |
| C        | 3                     | 4 \((20-8) = 0.33\)               | Roundup \((3 \times 0.33) = 1\) | \(1/3 = 0.33\) | 3         |

**Figure 5:** Original CPM/LOB Calculations Exceed the 20-day Deadline

**Figure 6:** Schedules developed using the proposed computation meet the original deadline

### 5. First-Come-First-Serve (FCFS) Crew Assignment

It can be noticed from the figures and equations proposed earlier is that they are most effective when the repetitive units are identical. As such, non-identical units make the formulations less effective, and
the overall scheduling process more difficult. For example, Figure 7a shows a repetitive task where unit2 has a smaller duration than the typical unit, while unit4 has a larger duration. A scheduling method has been developed to handle such a case [14] where the successor activity is scheduled in isolation, and then “shifted” back to be directly after its predecessor as seen in Figure 7. The shift is calculated based on the smallest difference (delta) between the start of the successor task (in that random point in the future where it was scheduled) and the finish of its predecessor. As such, this approach is commonly referred to as the “Delta-Shift” approach. While this approach offers an effective way to deal with non-identical tasks by treating the schedule as a Tetris game on the activity level, the produced schedule still exhibits significant gaps. As such there is a need for a new method that treats the schedule as a Tetris game on the unit level to reduce the gaps and provide a more efficient schedule.

**Figure 7:** Schedule Gaps Produced by the Delt-Shift Approach for Non-Identical Units

As opposed to the Delta-Shift approach dealing with each task as a large block, the proposed FCFS process deals with each crew at a time, not the task as a whole, as can be seen in Fig. 8 on an example of scheduling a successor activity (B) after a predecessor (shaded task A). The process is as follows:

1. In the first step, all crews are available. Hence, crew 1 is assigned to the first unit and its finish time is updated to reflect the finish time of activity B in the first unit.
2. In step 2, the finish time of the predecessor to activity B in the second unit is greater than the updated finish time of the crew 1. Hence, crew 1 is assigned to the second unit. To maintain work continuity, the start of crew 1 at the first unit is delayed by the difference between the finish time of the second unit and the scheduled finish of the first unit. Finally, the scheduled finish of crew 1 is updated to reflect the finish of activity B in unit 2.
3. In step 3, the finish time of the third unit is smaller than the scheduled finish of crew 1. Crew 2 is assigned to that unit and its finish time is updated accordingly. The predecessor finish time at the fourth unit is less than the finish time of crew 1 but equals to that of the crew 2. Therefore, crew 2 was assigned to unit 4.
4. Similarly, in step 4, crew 1 is assigned to unit 5, the start dates of its previous units (units 1 and 2) are delayed to remove work interruption, and its finish time is updated to reflect the finish of the activity in unit 5.
5. The procedure is continued until all units of activity B have been scheduled, and the process is repeated for other successors activities until all activities have been scheduled.
6. Example Application

To demonstrate the resulting schedule improvements of the proposed computations and algorithms, as well as the Tetris analogy in repetitive scheduling, an example project is considered, as shown in Figure 9. The example consists of four activities with CPM duration being 12 days. In this example, 10 road sections need to be completed within a total project duration of 24 days. All the units have the same duration with the exception of units 5 and 6, whose durations are 0.5 and 1.5 times the typical duration, respectively. Furthermore, Activity A must be performed from both sides of the road. In other words, units 1 and 10 need to start simultaneously, and their crews move from one unit to the next accordingly (e.g., one crew moves through sections 10 to 6 while the other moves through sections 1 to 5).

Figure 8: Steps of the Proposed FCFS Crew Assignment Process

Figure 9: Data and CPM/LOB Calculation Table of the Example

| Duration (D) | Desired Rate (R) | Necessary crews |
|--------------|------------------|-----------------|
| A: 3         | 9/(24-12) = 0.75 | Roundup (3 x 0.75) = 4 |
| B: 2         | 9/(24-12) = 0.75 | Roundup (2 x 0.75) = 2 |
| C: 3         | 9/(24-12) = 0.75 | Roundup (3 x 0.75) = 3 |
| D: 4         | 9/(24-12) = 0.75 | Roundup (4 x 0.75) = 3 |

To save time, Activity A must use 2 crews from each side of the road, thus, two crews go from section 10 to 9 to 8 to 7 to 6, while the other two crews go from 1 to 2 to 3 to 4 to 5.
The table in Figure 9 shows the CPM/LOB calculations of necessary crews, and the steps in producing the schedule, task-by-task, are shown in Figure 10. Initially, the schedule developed based on a $T_1$ value of 12-days (CPM duration), and the schedule (Figure 10) becomes 27 days, violating the deadline.

(a) Step 1 – Pieces of Task A Arriving  
(b) Step 2 – Task A Scheduled  
(c) Step 3 – Task B Arriving  
(d) Step 4 – Task B Scheduled  
(e) Step 5 – Task C Arriving  
(f) Step 6 – Task C Scheduled  
(g) Step 7 – Task D Scheduled

**Figure 10:** Tetris-like Steps of Generating the Schedule

It can be noticed that the schedule in Figure 10 has been created using the Delta-Shift approach. Using the same crew information, a schedule is developed using the proposed FCFS algorithm in Figure 11. The developed schedule has a duration of 26 days, which is smaller than the Delta-shift based
schedule in Figure 10. This proves the efficiency of the proposed approach and its potential time and cost savings due to creating more compact schedules.

![Figure 11: FCFS Schedule using Original Crew Calculations](image)

Since the FCFS schedule alone still did not meet the 24-day deadline, the calculations loop was then introduced and the parameter G is calculated to be \((24 - 12)/5 = 2.4\) days, thus, the new value for \(T_1\) becomes 14.4 days. Accordingly, the revised number of crews were assigned in a FCFS arrangement, and a new schedule was generated in Figure 12, where the crews for activity 4 increased to 4 instead of 3. The schedule satisfies the 24-day deadline constraint. The schedule of Figure 12 thus proves that using the FCFS crew assignment with the validation loop produces schedules with minimal gaps, to meet the project deadline, thus validating the efficiency of the proposed approach.

![Figure 12: Final FCFS Schedule using Validation Loop Calculations](image)

7. Conclusion

Repetitive scheduling is like a Tetris puzzle whose best solution is one that tightly packs all the activities next to each other, thus achieving minimum project time and maximum work continuity. To that end, this paper presented a variety of novel computations and algorithms to produce more efficient schedules by removing the schedule gaps produced by conventional CPM/LOB calculations. First, a formulation to calculate designed interruption was introduced to help reduce project durations by synchronizing the task delivery speeds and adjusting the task geometry. Second, a deadline validation loop was implemented to avoid deadline violations. Third, a novel scheduling algorithm (First-Come-First-Serve) was developed where the schedule is treated as a Tetris puzzle, thus producing minimal schedule gaps. The proposed contributions were implemented on a prototype example to demonstrate their effectiveness and versatility in terms of dealing with non-identical units and other practical project requirements and constraints. The deadline checking loop allows for adjusting the project schedule by manipulating one parameter only (\(T_1\)) as opposed to tweaking every activity manually. The implementation shows that is possible to produce repetitive schedules with shorter durations by modifying the task geometry and arrangement without the need for costly crashing alternatives.

Overall, repetitive scheduling is a key project management skill that is suggested to be an integral part of project management education so that these powerful techniques can become mainstream in
managing large-scale repetitive projects, which are the majority of infrastructure projects. To support this educational objective, interested readers can download useful education tools and a spreadsheet program to implement repetitive scheduling from the first author’s dropbox at: https://www.dropbox.com/s/dxtzo9bижнёго/Repetitive%20Scheduling.zip?dl=0. Many areas of research could also improve existing techniques and provide better decision support for these projects. These including integration with BIM models, simulation studies, large-scale optimization, and better visualization of the resulting schedules.

References
[1] Altuwaim, A., and El-Rayes, T. (2018). “Optimizing the Scheduling of Repetitive Construction to Minimize Interruption Cost.” Journal of Construction Engineering and Management, 144(7): 04018051.
[2] Bakry, I., O. Moselhi, and T. Zayed. (2014). Optimized Acceleration of Repetitive Construction Projects. Automation in Construction. 39: 145–151.
[3] O’Brien, J. J. (1975). “VPM scheduling for high-rise buildings.” Journal of the Construction Division, ASCE, 101(4), 895–905.
[4] Arditi, D., and Albulak, M. Z. (1986). Line-of-balance Scheduling in Pavement Construction. Journal of Construction Engineering and Management, 112(3), 411–424.
[5] Russell, A. D., and Wong, W. C. M. (1993). “New generation of planning structures.” Journal of Construction Engineering and Management, ASCE, 119(2), 196–214.
[6] Harmelink, D. J., and Rowings, J. E. (1998). Linear Scheduling Model: Development of Controlling Activity Path. Journal of Construction Engineering and Management, 124(4), 263–268.
[7] Harris, R. B., and Ioannou, P. G. (1998). Scheduling Projects with Repeating Activities. Journal of Construction Engineering and Management, 124(4), 269–278.
[8] Lucko, G. (2008) Productivity scheduling method compared to linear and repetitive project scheduling methods, Journal of Construction Engineering and Management, 134(9) 711–720.
[9] Kenley, R., and Seppänen, O. (2010). Location-based Management System for Construction: Planning, Scheduling and Control. Spon Press, London and New York.
[10] Su, Y. & Lucko, G. (2016). Linear Scheduling with Multiple Crews Based on Line-of-Balance and Productivity Scheduling Method with Singularity Functions. Automation in Construction, 70: 38-50.
[11] Shah, R. K. (2014). A New Approach for Automation of Location-Based Earthwork Scheduling in Road Construction Projects. Automation in Construction, 43: 156-169.
[12] Elbeltaei, E., and Dawood, M. (2011), “Integrated visualized time control system for repetitive construction projects”. Automation in Construction (20), pp. 940–953.
[13] Hegazy, T., and E. Kamarah. (2008). “Efficient Repetitive Scheduling for High-Rise Construction.” Journal of Construction Engineering and Management, ASCE, 134(4), 253–264.
[14] Long, L. and Ohsato, A. (2009). “A genetic algorithm-based method for scheduling repetitive construction projects” Automation in Construction, (18), pp. 499–511.