PRODUCTIVITY SCHEDULING METHOD WITH MAXIMUM CONSTRAINTS

ABSTRACT
This research investigates maximum constraints for linear scheduling via a mathematical model. Maximum constraints are improperly ignored by most current scheduling techniques, especially for linear schedules. However, they can have substantial practical relevance, e.g. to express safety constraints. Therefore the goal is to renew awareness of this type and enable its application to linear schedules with activities of constant productivity. The methodology includes a literature review, followed by an algorithm that relies upon singularity functions. Its first two steps refer to the existing Productivity Scheduling Method, which is extended with mathematical steps to handle maximum constraints correctly. For validation the new algorithm is applied to a schedule example with different scenarios and its peculiarities are discussed. Being able to incorporate maximum constraints will provide project managers with more realistic schedules.

KEYWORDS
Maximum Constraint; Linear Scheduling Method; Productivity Scheduling Method; Singularity Functions; Algorithm.

1. INTRODUCTION
Development and application of construction scheduling techniques like the Critical Path Method (CPM) or the Linear Scheduling Method (LSM) has been permeated with an implicit assumption, that one must determine how soon after one activity finishes the following one could start. Such minimum constraint or buffer between activities has many different origins, foremost is likely the desire of owners to finish projects in as short as possible a duration. But contractual, technical, or safety considerations in practice may also imply that after a predecessor finishes, its successor must start within a certain period or space, which constitutes a maximum constraints. Their treatment has been neglected in construction scheduling techniques. In particular, LSM, which was often considered to be only a graphical representation, does not consider maximum constraints. Previous research (Lucko 2008b) created a new mathematical approach for linear scheduling, but failed to include maximum constraints, which hampers its practical applicability. Therefore a mathematical model must be created and an algorithm must be derived that can properly handle this fundamental element to finally gain a comprehensive quantitative approach.

2. CRITICAL PATH METHOD
The origins of CPM lead back to the 1950s, when projects started to use network diagrams to explicitly portray relations between activities (Krishnamoorthy 1968). Kelley and Walker (1959) conceived of the method, which they claimed to be a “real, honest-to-goodness, outgrowth of a progressive management’s active search for better way to do things” (Kelley and Walker 1989, p. 7). Until then, Gantt Bar Charts dominated, which had been used since the early 1900s. Their main feature was the simplicity and efficiency to display start, progress, and finish of activities in a time-scaled graphical representation (Melin and Whiteaker 1981). The basic concept of CPM is to view projects as assemblies of discrete activities with start and finish events. They consume resources and interact with each other, forming a continuous path from first to last activity. CPM sequentially calculates all earliest and latest possible starts and finishes, and compares them to
identify a critical path and which activities may be delayed without harm. Two graphic versions exist, (1) representing activities as nodes that are connected with arrows (activity-on-node) or (2) activity-on-arrow. The former one is more adept at representing relations between activities.

2.1. Relationships

A vital input of CPM are relations between starts and finishes of adjacent activities within a schedule network, whose calculated results are important information for project managers (Fondahl 1962). Four common relations have been defined for activity-on-node, namely finish-to-start, start-to-start, finish-to-finish, and start-to-finish from predecessor to successor. But they make several limiting assumptions. First, only starts and finishes are linked, i.e. only point-to-point relations exist. Second, continuous relations or constraints do not exist in CPM, but exist in form of areas to denote required (minimum) time and work buffers between activities in linear schedules. Lucko (2008a) has called them range relations as compared to point-to-point relations. Figure 1a represents a point-to-point relation with an arrow; its diverging activities form a start-to-start link. Conversely, converging activities form a finish-to-finish link, which is omitted for brevity. Figure 1b generalizes that ‘at least [i.e. a minimum constraint] one time unit must always remain from predecessor to successor’ by applying it along the entire predecessor, as represented with a gray shaded area. Further generalization as time or work constraints is discussed below.

2.2. Forward and Backward Pass

In planning and scheduling with CPM, once all durations and relations are defined, two calculation steps are performed, the forward and backward pass through the network. They yield, respectively, earliest and latest dates for all activities. In the forward pass, it is explicitly assumed that all activities start as soon as their starting conditions are fulfilled – a minimum constraint. In the backward pass, their starts are assumed as late as possible while not harming successors in a worst case scenario to determine criticality and float, but again based on minimum constraints. Ultimately, the earliest starts and finishes are adopted as proactive targets, even though these may not be the most efficient or the safest by which to perform the project, as will be discussed. In some cases, maximum constraints may even be more relevant for decision-makers. Reasons can include safety, physical or non-physical requirements, or the need to meet certain milestones.

3. LINEAR SCHEDULING METHOD

Scheduling that relies upon networks has been deeply studied in the literature despite various conceptual limitations: They focus on start and finish of an activity, but fail to show anything in between, which precludes the scheduler from foreseeing delays. Moreover, the actual metrics of production, e.g. work quantity that has been completed or a location that has been reached, are by definition excluded. Furthermore, if a project involved repetitive work, its network becomes inflated to very cumbersome size. Finally, maximum constraints are still essentially disregarded,
even though they can be crucial in practice for safe, contractual and technical execution, and in
theory to gain a complete method. LSM or the similar Repetitive Scheduling Method are more
effective at portraying such activities, where repetitiveness is incorporated into a single larger
activity, which can also be variable (Harris and Ioannou 1998). Examples of projects are e.g.
highways, high-rise buildings, and any others that incur repetitive work, which is very common
in construction. Figure 2 shows a small LSM chart for three activities, where activity A precedes
B, which in turn precedes C.

Important observations include that the planned productivity of each activity must be known
as work that a crew can perform per period. In Figure 2, it is the inverse of the slope, because the
y-axis represents time and the x-axis is work quantity or location. In other words, A and C have
higher productivity and lower slope in this example. Note how they overlap in time. This hints at
the existence of – often not visualized – minimum constraints. They are preventive, as “activities
cannot both take place in the same location at the same time” (Lucko 2008a, p. 3). A “minimum
time constraint indicates that two activities cannot approach each other more than a specified
amount of time, at any time during the project duration” (Kallantzis and Lambropoulos 2004, p.
212). This implies that, different from networks, linear schedules employ continuous constraints.

Besides time constraints, building components and basic relations between activities also
infer space constraints (Echeverry Ibbs Kim 1991). The sequencing and scope of activities are
also bound to restrict the successor to start after a certain amount of work has passed, and that is
known as space buffer. Space and time buffers may be converted into one another by using the
common productivity rate of their activity. The circled minor difference is resolved per Figure 3.

For example, after placing concrete into slab formwork, several days must pass before it has
hardened sufficiently to support itself and other work on top of it. Often the planned productivity
is assumed as constant; and minimum constraints, or buffers, are also applied as constant over
the entire range of their host activity. Studies of maximum constraints are scarce. Only Kallantzis
and Lambropoulos (2004) applied maximum constraints graphically in LSM, but their research
lacked any underlying model or algorithm and appears to not have resonated in the literature.

3.1. Convergence and Divergence
In LSM, the relative productivities of adjacent activities matter greatly. If a preceding
activity has a lower productivity than the succeeding one, they converge. On the other hand, if
the predecessor has a higher productivity than its successor, they diverge. If both have the same
rate, they are balanced and performed in parallel. This ideal case gives the smoothest production
flow, but may not always be achieved in practice. In Figure 2, \{A, B\} diverge; \{B, C\} converge.

3.2. Float in Linear Schedules
Among the most commonly known and used float types in CPM are free float and total float.
The former asks how much an activity can be delayed before any successor is impacted, and the
latter asks how much delay would impact the project finish itself. Total float is less relevant in
the following discussion. Various other float types were defined in the literature, e.g. interfering
float as the different of total and free float, conditional float in the undesirable scenario of a late predecessor (Hajdu 1997), independent float in the worst-case scenario of a late predecessor and early successor for the given activity, and others that appear to be either unused or unknown in professional practice and are left to be studied and potentially revived under future research.

Extending the definitions of float types leads to the insight that other types can exist in LSM:

1. Early float occurs if activities converge to form a finish-to-finish relation by virtue of their relative productivities. The successor could start earlier than planned, but not earlier than its predecessor. (2) Late float occurs for divergence that generates a start-to-start link and the predecessor can finish later than planned, but not later than its successor (Lucko and Peña Orozco 2009). Still, none of the previous derivations addressed maximum constraints and their schedules may be impractical.

4. SINGULARITY FUNCTIONS

Lucko (2007) first studied how to equip LSM with analysis capabilities of singularity functions. Their mathematics was introduced independently by Macaulay (1919) in Great Britain and by Föppl (1927) in Germany. The definition of a term is a case distinction (Lucko 2008a):

\[
s \cdot (x-a)^n = \begin{cases} 
0 & \text{for } x < a \\
 s \cdot (x-a)^n & \text{for } x \geq a 
\end{cases}
\]

Eq. 1

It determines that the function is equal to zero at any argument \( x \) less than \( a \), and \( s \cdot (x-a)^n \) at any \( x \) greater than or equal to \( a \). This definition finally turned graphical LSM into a numerical model, which can be handled in algorithms and also implemented in computer programs. Each activity within the project is modeled with exactly one function. The Productivity Scheduling Method (PSM, Lucko 2008b) is explained in a following section on how to optimize a schedule.

Before explaining it, converting time and work buffers must be resolved, which the literature failed to address. The algorithm that this research will develop can handle either type in dealing with maximum constraints, while focusing on time optimization. In singularity function for work (or space) buffers, the term \([-1/P(P) \cdot (x - (\text{max} - w_{buffer}))^1]\) is added to a time buffer equation to convert it into a work (space) buffer; where \( P(P) \) is the predecessor productivity; \( x_{\text{max}} \) is the total work quantity, and \( w_{\text{buffer}} \) is the work buffer. This term removes the small triangle by which time and work buffer differ. Conversely, the term is simply subtracted. In the following algorithm, asterisks after the buffer equation mean that this term must be added if it is a work buffer. With this provision for automatic conversion, the algorithm can handle either type of constraints. One module of the new algorithm necessarily applies the previously developed PSM (Lucko 2008b) for modeling linear schedules by using singularity functions, which is explained in detail in a following section; while the remaining majority of the algorithm constitutes new original work.

5. MAXIMUM CONSTRAINTS

Maximum constraints in terms of time can be necessary for various technical, operational, as well as contractual reasons. Safety is an overarching concern and a primary consideration. For example, projects that involve trenches or tunneling will require substantial excavation activities. Subsurface work in all types of soil except for solid rock must be supported to prevent cave-ins or collapse. Time and distance between excavation, drilling, boring, or blasting and their proper installation of bracing, sheeting and shoring or lining are subject to vital maximum constraints, because they cannot remain unsupported for long. Technical considerations apply in practice, e.g. for installing building enclosures, especially if existing buildings are renovated and remain
occupied during the work. Not having façade elements, insulation, and windows make the interior vulnerable to weather and therefore must be completed as soon as possible. Besides, the contract may allow only a limited duration for phased renovation. Failure to complete the work on time may incur liquidated damages, because the owner lacks revenue-generating use of their building. Moreover, subcontractors typically are only involved with a project for a limited time and may be unavailable after a certain date, e.g. when starting work on another project. They would thus be scheduled to perform before a deadline. Furthermore, maximum constraints may also be employed to help achieve schedule milestones at a general level, or as a detailed means to guarantee that dependent activities remain in close proximity to one another, e.g. asphalt paving and compacting by roller, which must occur while the asphalt mixture is still hot and workable.

Handling maximum constraints dates to Roy (1959), who developed the Method of Potentials (also called Roy Method or Metra Potential Method, MPM) at the Université Paris-Dauphine. This method depicts activities as nodes; precedence relations by default are start-to-start and shown as arrows. Figure 4 shows an example of a to g between start D (début) and finish F (fin). Numbers on the arrows are equivalent to start-to-start with lags in CPM. For example, 8 hours must pass after the start of c before f can start as well. The noteworthy idea of maximum constraints in this method can be understood as “backward-acting restraints” (Battersby 1970, p. 92). Here the arrows of relations direct to an activity which would happen before according to the logic of precedence in the schedule. This is a negative lag (or lead). In the example of Figure 5, it means p must start at least -12 hours from start of r. Yet its negative value makes the statement sound unreasonable. A more intuitive formulation would be ‘r must start within the first 12 hours after start of p’.

6. JUSTIFICATION AND NEED
As mentioned, the feasibility and safety of schedules is hindered by omission of maximum constraints in the existing scheduling techniques. Several previous key studies have described maximum constraints. Roy (1959) provided an early discussion of maximum constraints, but did so in the abstract context of graph theory and did not create an algorithm that would easily lend itself to integration into project scheduling. Hajdu (1997) aptly discussed maximum links for different combinations of predecessor and successor starts and finishes as an advanced topic in network scheduling. Notably, four types were illustrated as small network and linear schedules in pairs. But the latter used a generic 100% axis designation and was intended only as an alternative representation of the networks. They were solved iteratively by modifying starts and finishes until their minimum and maximum constraints were fulfilled. The approach was not developed into a linear scheduling model. Kallantzis and Lambropoulos (2004) handled examples of linear schedules with minimum and maximum time and distance (or work) constraints and allowed interruptability of activities to fulfill both types of constraints. But they did not develop any calculation model, nor did they create an algorithm to generate a feasible schedule. Lucko (2008b) finally created a mathematical model, which removed the previous limiting notion that linear schedules were essentially a graphical method. But he failed to consider maximum links. Therefore, despite the significant conceptual promise of linear schedules, which represent two dimensions of information – time and work, as opposed to only time in bar charts and networks, researchers have not yet realized the full theoretical potential. There remains an important unfilled need to equip linear scheduling with maximum constraints as part of a mathematical approach. This research will therefore examine possible cases where maximum constraints may be violated and derive an algorithm that will automatically generate feasible schedules, regardless of the type of initial infringement. In other words, the algorithm must avoid or identify and adjust any infeasible constellation. Singularity functions, having proven their capability to model complex schedules accurately and flexibly, will be employed to formulate this extension.

7. RESEARCH OBJECTIVES

The goal of this research is to facilitate the application of maximum constraints within linear schedules without a need for graphical representation. It will be addressed by several objectives:

- First, the overall feasibility of any schedule must be determined for a given set of constraints;
- Second, the possible ways in which these constraints can relate with adjacent activities must be examined and singularity functions must be employed to represent the maximum buffers;
- Third, a structured algorithm must be derived to efficiently handle minimum and maximum constraints in the newly expanded approach, which will be more comprehensive than PSM.

A limitation will be that only activities with constant productivity will be analyzed due to space limitations; further generalization to variable productivity will be addressed in future research.

8. RESEARCH METHODOLOGY

To achieve the objectives of this research a detailed methodology will be applied. A literature review of existing scheduling techniques and their limitations, including PSM (Lucko 2008b), will provide inspiration on how to represent the buffers with singularity functions and the best sequence of steps that the algorithm should take to handle minimum and maximum constraints in conjunctions. The algorithm will be modular, including inputs to define schedule elements and constraints, sequentially assembling the schedule, adjusting activities if needed if their initial constellation violates a constraint, and any others that may become necessary. Such approach is
called “divide-and-conquer” in computer coding literature, for it “decompose[s] the problem into subproblems that resemble the original problem on a reduced scale” (Reingold 1996, p. 19).

After each module is designed and verified, it will be examined how it interacts with others, noting patterns, loops, and any alternative paths within the algorithm that a solution may use. It will employ test schedules of sufficient diversity in productivity and buffers. Two remedies for violations will be explored, changing productivity or interrupting the predecessor or successor.

9. MAXIMUM CONSTRAINT: PROVISIONS

A set of basic provisions – “a statement in an agreement... that a particular thing must happen or be done” (Cambridge Dictionaries Online 2015) – has been established to guide determining feasible constellations of buffers, i.e. the algorithm will employ them to assess whether activities in the schedule already fulfill minimum and maximum constraints or still require modification.

- **Provision I:** \( y(x)_p^{\text{max}} \leq y(x)_p^{\text{max}} \forall x \in [0, x_{\text{max}}] \). Minimum predecessor buffer must be less than or equal to maximum predecessor buffer for any \( x \) within valid range. Inputs should fulfill this intuitive rule, otherwise the inputs may be marked as instantly infeasible and thus erroneous;

- **Provision II:** \( y(x)_s \geq y(x)_p^{\text{max}} \forall x \in [0, x_{\text{max}}] \). Successor must be larger than or equal to minimum predecessor buffer for any \( x \) within valid range. This establishes the lower bound;

- **Provision III:** \( y(x)_s \leq y(x)_p^{\text{max}} \forall x \in [0, x_{\text{max}}] \). Successor must be less than or equal to maximum predecessor buffer for any \( x \) within valid range. This establishes the upper bound.

This guidance for the algorithm to generate a feasible solution fulfills Research Objective 1.

10. ALGORITHM

The following algorithm of Figure 6 considers pairwise activities pairwise in one cycle per each, while remedying any potential violations. Initially, it will perform up to five steps, which later may vary depending on the complexity of the schedule. It also queries the user in choosing which activity to alter (predecessor or successor) and how (interrupting or changing productivity) to resolve a conflict. These options will be described in more detail in the following sections.

**<INSERT FIGURE 6 HERE>**

Activities with multiple successors or predecessors must be handled carefully. Every time a predecessor changes, all successors must be tested by Provision III. Also, if a successor changes its pattern, either productivity or interruption, Provision III must be applied to all predecessors. This approach guarantees that in the end the schedule will fulfill all buffers without any conflicts.

The algorithm distinguishes Cases I through IV, depending on whether the user wishes a predecessor or successor to be change productivity or be interrupted. For example, at least one hour must remain between two tunneling crews to avoid interference (minimum constraint), but soil can never be unsupported for more than two hours due to intruding groundwater (maximum constraint). Boring can produce 4.5 m/h (predecessor \( P \)), but supporting (successor \( S \)) can only build 1.5 m/d. The short tunnel is 9 m long. A linear schedule per Figure 7 is clearly infeasible.

**<INSERT FIGURE 7 HERE>**

10.1. Module A: Define Inputs
In Module A the user sets these inputs: Predecessor productivity \( P(P) \); successor productivity \( P(S) \); minimum and maximum buffers of \( P \) (\( \text{minBuffer} \), \( \text{maxBuffer} \)); and work quantity \( (x_{\text{max}}) \). Each step in this module refers to one or two inputs. The very first decision restricts the process to activities whose productivities are constant throughout the work amount. Provision I is fulfilled with proper input for minimum and maximum buffers. Continuing the tunneling example, \( P(P) = 9/2 \), and \( P(S) = 9/6 \) and the buffers are \( \text{minBuffer} = 1 \cdot (x - 0)^0 \) and \( \text{maxBuffer} = 2 \cdot (x - 0)^0 \) and the work quantity is \( x_{\text{max}} = 9 \). Units are omitted here for brevity.

10.2. Module B: Perform PSM with Minimum Constraints

Invented by Lucko (2008b), PSM contains 7 steps that are guaranteed to yield the minimum project duration for a linear schedule with minimum constraints. A linear schedule diagram can illustrate the mathematics, but is not required. In PSM the vertical axis measures time (which must be minimized) and the horizontal axis shows work quantity or location. To add maximum constraints, one must first consider that minimum and maximum constraints together may create constellations in which the successor must be modified before it becomes feasible. This can be accomplished either by changing its productivity or by interruption. Continuing the tunneling example of Figure 7, the successor violates the constraint and thus presents a collapse hazard.

Module B now expands PSM to being mathematically able to handle maximum constraints:

- **Step 1: Initial Equation**

  Equation 2 is the singularity function for the first activity with its intercept and slope terms.
  \[
  y(x)_P = 0 \cdot (x - 0)^0 + 2/9 \cdot (x - 0)^1 
  \]  
  Eq. 2

- **Step 2: Buffer Equations**

  Equations 3 and 4 establish minimum and maximum buffers. Two approaches are possible. One can begin at the origin and describe each slope. More efficient is to add the time buffer to the activity equation. Work buffers would have an extra term at their end as has been described earlier and illustrated in Figure 3. Since the successor is the last activity, its needs no buffers.
  \[
  y(x)_P^{\text{min}} = 1 \cdot (x - 0)^0 + 2/9 \cdot (x - 0)^1 
  \]  
  Eq. 3
  \[
  y(x)_P^{\text{max}} = 2 \cdot (x - 0)^0 + 2/9 \cdot (x - 0)^1 
  \]  
  Eq. 4

- **Step 3: Initial Stacking**

  In the order of precedence, activities and buffers are alternatingly stacked. Equation 5 uses an initial convservative assumption of CPM to start a successor only at the finish of its predecessor.
  \[
  y(x)_S_{\text{initial}} = 3 \cdot (x - 0)^0 + 6/9 \cdot (x - 0)^1 
  \]  
  Eq. 5

- **Step 4: Minimum Differences**

  Subtraction in Equation 6 yields the minimum distance \( \alpha \) from the successor activity to its predecessor buffer. Its slope is positive, i.e. said distance increases from a minimum of 2 at \( x = 0 \).
  \[
  \alpha = y(x)_S - y(x)_P^{\text{min}} = 2 \cdot (x - 0)^0 + 4/9 \cdot (x - 0)^1 
  \]  
  Eq. 6

- **Step 5: Differentiation**

  If needed, said difference can be differentiated to confirm the nature of points of proximity. For non-complex cases like the current example, this step can often be omitted for brevity.

- **Step 6: Consolidation**
The minimum distance is permissible move downward, i.e. earlier on the time axis. Equation 7 subtracts it from the intercept of the initial successor equation. Further activities and their buffers in a larger linear schedule would be treated in the same manner by repeating the steps. Provision II is fulfilled automatically by following the previous steps.

\[ y(x)_S = 1 \cdot (x - 0)^0 + 6/9 \cdot (x - 0)^1 \]

**Step 7: Criticality Analysis**

The critical path is determined. It extends from the start to the finish of the entire project through activities or their segments and can branch in some constellations. For brevity, this is omitted. This approach has generated final equations for all activities, which are guaranteed to be optimum for the given minimum constraints. As described in this section, the approach has potential, but has failed in two ways: First, PSM was conceived with only minimum buffers, so that systematic examination and extension is required. The maximum constraint in the tunneling example was not explicitly used to generate a feasible successor equation. An algorithm is therefore needed to formalize it for all possible constellations of activities and enable computerization. Both advancements will addressed next.

Figure 7 displays the shaded area as the feasible zone for the successor. It must be performed fully between its lower and upper bounds it so that no constraint is violated. It dictates a range of productivity within which it can be performed, which may or may not be possible for the initially planned crew. In the following section, PSM is newly extended to handle maximum constraints and applied to the example, where P and S are predecessor and successor.

10.3. Module C: Testing Against Provision III

Module C tests pairs of directly adjacent activities versus Provision III, and defines a variable \( \delta(x) \) of Equation 8 as the difference of successor \( S \) minus maximum buffer of predecessor \( S \). The module asks ‘is the maximum buffer of \( P \) greater than or equal to \( S \)?’ If all pairs pass muster, then the algorithm will stop. Otherwise, the schedule in deemed infeasible. The exact location of the project is identified. Continuing the example, Equation 8 is less than zero for any \( x \geq 9/4 \), which is part of the valid \( x \)-range of \( \{0 \text{ to } x_{max} = 9\} \). The maximum constraint has thus been identified as being violated. This module fulfills Research Objective 2.

\[ \delta(x) = y(x)_S^{max} - y(x)_S = 1 \cdot (x - 0)^0 - 4/9 \cdot (x - 0)^1 \]

10.4. Module D: Definition for Divergence

Modules D and further ones are only active if Provision III is not fulfilled. The variable \( FL(x) \) of Equation 9 is the Free Float of the successor \( S \). Lucko and Peña Orozco (2009) computed free float in linear schedules as the successor equation minus the predecessor minimum buffer equation, which is used here calculate float. A distinction into Early Float (EF) or Late Float (LF) is automatically made when the float is analyzed. Taking the derivative of the float equation identifies whether the activities converge or diverge for a negative and positive derivative, respectively. This relies upon the assumption of constant buffer values that follow the shape of their host activity. In the example, Equation 10 finds the values to be positive, so the activities diverge as Figure 7 shows. Cases I and II of Figure 6 are considered for this project. Cases III and IV do not apply here, but the algorithm would handle convergence analogously.

\[ FL(x) = 0 \cdot (x - 0)^0 + 4/9 \cdot (x - 0)^1 \]
10.5. Module E: Interruptability or Variable Productivity

Module E requires user input to determine if an activity should be planned to be interrupted or its productivity changed by modifying a crew in its number of laborers, or the number, size, or capacity of machinery. Note that any interruption is applied within the minimum and maximum constraints of predecessor activity as a whole, which is automatically also fulfilled for any tasks or segments within it. The subsequent example will illustrate this particular beneficial feature of the new algorithm. Finally, for this module four different outputs are possible. User expertise is vital in this module. For better illustration, the user here wishes to consider both Cases I and II.

10.6. Case I: Activities Diverge; Variable Productivities

Another variable RR, which is Required Rotation (i.e. slope change), is defined for Equation 11 to achieve a desired productivity in either activity that fulfills the maximum constraint. It will be applied to either P or S as determined by the user. After rotation per Figure 8, if needed, buffers are updated for the predecessor with the same equations per the description of Module B.

In the example, the minimum delta \( \delta(x) \) is equal to -3 at \( x = 9 \). The RR therefore is 1/3 per Equation 11. The user selects that increasing the productivity of \( S \) is best. This gives its modified singularity function in Equation 12. To reach a feasible linear schedule, it thus will be necessary to increase its productivity from 1.5 m/h to at least 3 m/h per Figure 9. In practice, this could be achieved by e.g. (a) increasing crew size or (b) selecting higher capacity supporting equipment.

\[
RR = \left| -\frac{3}{9} \right| \cdot \langle x - 0 \rangle^1 = \frac{1}{3} \cdot \langle x - 0 \rangle^1
\]

\[
\gamma(x)_s = 1 \cdot \langle x - 0 \rangle^0 + \frac{1}{3} \cdot \langle x - 0 \rangle^1
\]

10.7. Case II: Activities Diverge; Interruptible Activities

Equation 13 calculates the point from which onward \( S \) violates the maximum buffer of \( P \), i.e. it finds a changeover. Equation 8 gives \( \delta(x) = 0 \) at this work quantity. Effectively, the activity splits into two segments here; one compliant, the other in violation. This is the default coordinate where interruptability could be applied beneficially. The possible interruption duration is given by the float at this coordinate. If the user permits interruptability, then it is added to \( \delta(x) \) as part of the modified activity in Equation 14. Of course it is also subtracted from the float equation in Equation 15. This process automatically repeats while constraint violations exist, i.e. while any Changeover is less than the maximum work amount \( x_{\text{max}} \). Per Figure 10, the user can select whether the calculated optimum for interruption should be used, which may be fractional, or whether the interruption should be applied already at a rounded down integer work quantity. This can be more practicable if the linear schedule is planned and controlled primarily in discrete integer work units. Note that if the predecessor is interrupted, the interruption itself will be positive as Figure 11 shows. If the successor it interrupted, the interruption is negative, which means that more than one instance of the same activity will be active at the same time. Such overlap will require more crews and can only be implemented if they are available in practice.

\[
x_{\text{changeover}} = \delta^{-1}(0) = 9/4 < x_{\text{max}}
\]
\[
\delta(x) = 1 \cdot \langle x - 0 \rangle^0 - 4/9 \cdot \langle x - 0 \rangle^1 + 1 \cdot \langle x - 9/4 \rangle^0
\]
\[
FL(x) = 0 \cdot \langle x - 0 \rangle^0 + 4/9 \cdot \langle x - 0 \rangle^1 - 1 \cdot \langle x - 9/4 \rangle^0
\]
Eq. 14
Eq. 15

For the example this generates final Equations 17 for \(\delta(x)\), 18 for \(FL\), and 19 for \(x_{\text{changeover}}\).

\[
\delta(x) = 1 \cdot \langle x - 0 \rangle^0 - 4/9 \cdot \langle x - 0 \rangle^1 + 1 \cdot \langle x - 9/4 \rangle^0 + 1 \cdot \langle x - 9/2 \rangle^0 + 1 \cdot \langle x - 27/4 \rangle^0
\]
\[
FL(x) = 0 \cdot \langle x - 0 \rangle^0 + 4/9 \cdot \langle x - 0 \rangle^1 - 1 \cdot \langle x - 9/4 \rangle^0 - 1 \cdot \langle x - 9/2 \rangle^0 - 1 \cdot \langle x - 27/4 \rangle^0
\]
Eq. 16
Eq. 17
Eq. 18

\[
x_{\text{changeover}} = \delta^{-1}(0) = 9 = x_{\text{max}}
\]

The algorithm now knows all variables that are needed to calculate interruptions for a given activity. Interrupting a predecessor would stop it at a certain coordinate and return to work at another. On the other hand, interrupting a successor means that over a certain range the work of multiple crews will overlap. In the example, the former case of interrupting \(P\) is used. Equation 19 is generated for the activity itself. For its minimum buffer, the equation here would have an intercept of 1; for the maximum constraint an intercept of 2. Predecessor \(P\) continues with the same productivity as before, but with three interruptions at \(x = \{9/4, 9/2, x = 27/4\}\) per Figure 11.

\[
y(x)_p = 0 \cdot \langle x - 0 \rangle^0 - 2/9 \cdot \langle x - 0 \rangle^1 + 1 \cdot \langle x - 9/4 \rangle^0 + 1 \cdot \langle x - 9/2 \rangle^0 + 1 \cdot \langle x - 27/4 \rangle^0
\]
Eq. 19

10.8. Case III: Activities Converge, Variable Productivities

The case that Figure 12 shows starts like Case I. As either activity may change productivity, the necessary rotation is defined as the ratio between the minimum \(\delta(x)\) and the work quantity \(x_{\text{max}}\). After a user decides on whether to increase productivity of \(P\) or decrease \(S\), the algorithm either subtracts the rotation from \(P\) or adds it to \(S\) to bring it to an earlier start. In case of rotating \(P\), the buffers must be updated by performing the calculations as described for Module B again.

10.9. Case IV: Activities Converge, Interruptible Activities

Case IV employs the assumption that it is best to modify the start of \(S\) to its latest possible time by considering the maximum buffer of \(P\). This is the first output in Figure 13. Afterward the users inputs which activity to interrupt. New \(\delta(x)\) and \(FL\) equations are computed and the changeover \(x_{\text{changeover}}\) is determined. Different from before, here the constraint is the minimum buffer, because \(S\) was previously modified so that no maximum constraint is violated. The changeover is thus calculated as the one point where \(FL(x) = 0\). Same as before, the user can elect to interrupt the activity at the optimum coordinate or an earlier integer work quantity for practicability. Then it is checked if the changeover is smaller or larger than the maximum work quantity \(x_{\text{max}}\) for whether the algorithm must continue or stops. These algorithm steps are executed at least once. Afterward, an interruption is inserted at the changeover. Its value is the \(\delta(x)\) at the changeover, which is the maximum interruption possible. If \(P\) is modified, then the interruption is applied to its equation repeatedly until all constraints are fulfilled; new equations for buffers, \(\delta(x)\) and \(FL\) are computed each time. Otherwise, the interruption is assigned to \(S\), and
the process steps are repeated until no constraint is violated. With the conclusion of this algorithm, Research Objective 3 is fulfilled.

<INSERT FIGURE 13 HERE>

11. EXAMPLE: PIPELINE PROJECT

The functioning of the newly developed algorithm will be validated by solving an example of a pipeline project presented by Damci et al. (2013). It consists of seven sequential activities per Table 1, with a total length of 25 km, and their productivities, buffers, and starts and finishes.

<INSERT TABLE 1 HERE>

To plan the pipeline, this research assumes the following maximum constraints: For safety, shoring (C) must occur within 4 days after excavating (B). After testing (E), backfilling (F) must occur within 6 days so that the tested pipes are not damaged. Heavy rain can cause trenches improper settlement; therefore compacting (G) must be performed within 10 days of backfilling (F). Equations 20, 21, and 22 are three examples of equations before the algorithm is applied. But Provision III is not fulfilled by these three equations, they violate their maximum constraints.

First, the initial equation of C is tested. It violates Provision III; the algorithm checks whether B and C diverge or converge via the float derivative. Since it is negative, they diverge (Module D), so the options of interrupting or changing productivity exist. Here interrupting the successor is chosen for Module E. The algorithm sets the successor start to the start of the maximum buffer of the predecessor (first step of Case IV), and from there finds $x_{\text{changeover}}$ (float equation reaches zero). Interruptions are selected to occur at integers only, so the results are rounded down. Its value is the maximum allowed, which is the difference between maximum and minimum buffer. A similar solution process occurs for increasing the productivity of $F$, which the user desires here. Factors for such decision are the scope of work, cost of interruptions, resource availability, time of the year, weather, experience, etc. The maximum constraint of $E$ is violated. Per the float derivative, the activity pair diverges. The algorithm determines how much the productivity must change (Case I) and subtracts this from the successor. That implies that it is performed faster and in practice must either use e.g. a longer daily shift, a larger crew, or higher capacity equipment.

The third pair, $F$ and $G$, undergo the same process. The maximum buffer is violated and the activities diverge. The user elects to interrupt the successor again. Since this pair diverges, not converges like $B$ and $C$, the result for Case II will be different. First the algorithm finds $x_{\text{changeover}}$, which is where the maximum constraint is violated and $\delta(x)$ reaches 0. Here $x_{\text{changeover}} = 11.59$, which is rounded down to 11. The interruption itself is found from the float function at $x_{\text{changeover}}$. Then the successor equation is modified using the maximum buffer and $\delta(x)$. Two interruptions must be inserted, the second one at $x = 22$. Therefore crews overlap and three crews must work to execute this modified activity. Equations 20 through 25 list the initial and modified equations.

$$y(x)_{C \text{ before}} = 59/3 \cdot (x - 0)^0 + 1/3 \cdot (x - 0)^1$$

$$y(x)_{C \text{ after}} = 395/12 \cdot (x - 0)^0 + 2/3 \cdot (x - 0)^1$$

$$y(x)_{G \text{ before}} = 419/12 \cdot (x - 0)^0 + 5/4 \cdot (x - 0)^1$$

$$y(x)_{G \text{ after}} = 111/6 \cdot (x - 0)^0 + 1/3 \cdot (x - 0)^1 + 3 \cdot (x - 18)^0$$

Equation 20
Equation 21
Equation 22
Equation 23
\[ y(x)_{F_{\text{after}}} = 127/4 \cdot (x - 0)^0 + 14/25 \cdot (x - 0)^1 \quad \text{Eq. 24} \]
\[ y(x)_{G_{\text{after}}} = 135/4 \cdot (x - 0)^0 + 5/4 \cdot (x - 0)^1 - 759/100 \cdot (x - 11)^0 - 759/100 \cdot (x - 22)^0 \quad \text{Eq. 25} \]

Figure 14 shows the linear schedules before and after maximum constraints considerations. If maximum constraints are considered, a different and possibly more efficient schedule can result. The initial version ignored some important considerations that a real-world project may incur.

<INSERT FIGURE 14 HERE>

12. CONTRIBUTIONS TO THE BODY OF KNOWLEDGE

The importance of this research lies in finally overcoming a conceptual limitation of handling only minimum constraints in LSM as well as PSM, which had described how to analyze linear schedules with singularity functions (Lucko 2008b). This may have hindered the use of linear schedules in practice, because important safety, contractual, and technical considerations that require maximum constraints could not be modeled. Besides raising awareness of the importance of maximum constraints, this research has derived an algorithm to not just identify any constraint violations, but resolve them per user preferences. Contributions to the body of knowledge are:

- This research has brought awareness to the importance of maximum constraints and enabled their use in modeling and improving linear and repetitive schedules;
- This research has enabled verifying the feasibility of linear and repetitive schedules in terms of possible violations of the maximum constraints that they may contain;
- This research has given the user the ability to determine which approach (interruption or productivity change) shall be taken to automatically generate feasible schedules.

13. CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE RESEARCH

Most methods that had been published in the literature ignored maximum constraints when calculating precedence relations between activities. In particular, the Productivity Scheduling Method (Lucko 2008b) had omitted such relations in its approach. The new algorithm accurately handles maximum constraints for linear schedules whose productivity is constant. It has five modules with four cases, which differ in relative productivities of activity pairs and user selection on preferring productivity changes or interruptability, which lead to different outputs.

The algorithm activates modules C and others if checking against the Provision finds that any constraint is violated and therefore a modification must be generated. All mathematics are solved with singularity functions and only require knowledge of geometry and a bit of calculus. Of course, it is not intended that users apply the algorithm manually in practice, but could do so to verify the output from a computer implementation. While not required, graphically representing the input and output is helpful to visualize the linear schedule and how user decisions affect it. Allowing multiple productivities in such analysis and optimization, which has been a limiting assumption in the present research, is an important extension that future research should explore.

ACKNOWLEDGEMENTS

Financial support by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Coordination for the Improvement of Higher Education Personnel – CAPES Foundation), Ministry of Education, Brazil from 2014-2015 for the first author is gratefully acknowledged.
REFERENCES


Figure 1a: Point-to-Point Relation with Arrow

Figure 1b: Continuous Relation with Buffer
Figure 2: Linear Schedule with Buffers (modified after Hinze 2008, p. 301)
Figure 3: Conversion of Time and Work Buffers
Figure 4: Node Diagram (adapted from Battersby 1970, p. 91)
Figure 5: Logical Loop (adapted from Battersby 1970, p. 92)
Figure 6: Phase I of Algorithm
Figure 7: PSM Example I
Figure 8: Algorithm Part for Case I
Figure 9: PSM with Maximum Constraints, Solution for Case I
Figure 10: Algorithm Part for Case II
Figure 11: PSM with Maximum Constraints, Solution for Case II
Figure 12: Algorithm Part for Case III
Figure 13: Algorithm Part for Case IV
Figure 14a: Pipeline Example Before Considering Maximum Constraints

Figure 14b: Pipeline Example After Considering Maximum Constraints
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Start</td>
<td>Finish</td>
</tr>
<tr>
<td>A</td>
<td>Loading and clearing</td>
<td>1</td>
<td>2</td>
<td>-</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>B</td>
<td>Excavating</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>29/2</td>
<td>27</td>
</tr>
<tr>
<td>C</td>
<td>Shoring and laying aggregate</td>
<td>3</td>
<td>1</td>
<td>-</td>
<td>59/3</td>
<td>28</td>
</tr>
<tr>
<td>D</td>
<td>Laying pipes</td>
<td>4/3</td>
<td>3/2</td>
<td>-</td>
<td>62/3</td>
<td>473/12</td>
</tr>
<tr>
<td>E</td>
<td>Testing</td>
<td>5/2</td>
<td>2</td>
<td>6</td>
<td>371/12</td>
<td>491/12</td>
</tr>
<tr>
<td>F</td>
<td>Backfilling</td>
<td>3/2</td>
<td>2</td>
<td>10</td>
<td>395/12</td>
<td>595/12</td>
</tr>
<tr>
<td>G</td>
<td>Compacting</td>
<td>4/5</td>
<td>-</td>
<td>-</td>
<td>419/12</td>
<td>397/6</td>
</tr>
</tbody>
</table>

Table 1: Activity List (modified from Damci et al. 2013)
FIGURE CAPTIONS
Figure 1a: Point-to-Point Relation with Arrow
Figure 1b: Continuous Relation with Buffer
Figure 2: Linear Schedule with Buffers (modified after Hinze 2008, p. 301)
Figure 3: Conversion of Time and Work Buffers
Figure 4: Node Diagram (adapted from Battersby 1970, p. 91)
Figure 5: Logical Loop (adapted from Battersby 1970, p. 92)
Figure 6: Phase I of Algorithm
Figure 7: PSM Example I
Figure 8: Algorithm Part for Case I
Figure 9: PSM with Maximum Constraints, Solution for Case I
Figure 10: Algorithm Part for Case II
Figure 11: PSM with Maximum Constraints, Solution for Case II
Figure 12: Algorithm Part for Case III
Figure 13: Algorithm Part for Case IV
Figure 14a: Pipeline Example Before Considering Maximum Constraints
Figure 14b: Pipeline Example After Considering Maximum Constraints

TABLE CAPTIONS
Table 1: Activity List (modified from Damci et al. 2013)
PRODUCTIVITY SCHEDULING METHOD WITH MAXIMUM CONSTRAINTS

By Jonathas Sousa Reis 1 and Gunnar Lucko, Ph.D. 2

1 Undergraduate Research Assistant, Construction Engineering and Management Program, Department of Civil Engineering, Catholic University of America, Washington, DC 20064, 1reis@cardinalmail.cua.edu, affiliated with Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES Foundation), Ministry of Education, CEP 70.040-020 – Brasília-DF, Brazil.

2 Associate Professor of Civil Engineering and Director of Construction Engineering and Management Program, Department of Civil Engineering, Catholic University of America, Washington, DC 20064, lucko@cua.edu.

ACKNOWLEDGEMENT

This work was supported by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Coordination for the Improvement of Higher Education Personnel – CAPES Foundation), Ministry of Education, Brazil from 2014-2015 for the first author.