History and Some Latest Developments of Precedence Diagramming Method

Miklós Hajdu
Szent István University, Gödöllő, Hungary; Department of Architecture; Facuty of Architecture and Civil Engineering;
miklos.hajdu63@gmail.com

DOI 10.5592/otmcj.2015.2.5
Research paper

Keywords

Precedence Diagramming Method, Activity overlapping, Point-to-point relations

ALTHOUGH THE BIRTH OF PRECEDENCE DIAGRAMMING METHOD (PDM) IS NOT AS RADIANT AS OF CPM OR PERT, BUT IT IS DEFINITELY THE PREVAILING SCHEDULING TECHNIQUE OF OUR TIMES. THIS POPULARITY IS DUE TO ITS MODELING FLEXIBILITY OVER OTHER TECHNIQUES AND THE EASY-TO-UNDERSTAND MATHEMATICAL MODEL BEHIND THE TECHNIQUE. However, even this technique has its own limitations; modeling overlapping activities in a proper way seems to be a never-ending debate. The reason of this can be found in the fundamentals of PDM technique; the four precedence relationships that form connections between the end-points of the activities, and the activities that are assumed to be continuous with constant production speed. These fundamentals of PDM have their own consequences to scheduling practice; it is more and more apparent among professionals that activity overlapping in PDM cannot be modeled adequately. Different solutions were proposed for solving this problem from the application of negative lag, through the combination of SS and FF relations to the fragmentation of activities. All these solutions have their shortcomings. Probably the fragmentation technique has led to the development of point-to-point type of relation that can connect any arbitrary points of the dependent activities. The objective of this paper is to analyze the pros and cons of different solutions that are used for modeling overlapped activities, then to show how newly defined point-to-point relations can be used for this purpose. Algorithms that handles point-to-point relations with minimal and maximal lags are also presented. The main finding of the paper is that newly developed point-to-point relations are better from theoretical and practical point of view than the solutions based on traditional precedence relationships, but they still cannot provide theoretically perfect solution for overlapping. This paper is the fully extended version of a paper building on the results already presented on the Creative Construction Conference [1].

INTRODUCTION

Widely used network techniques are more than half a century old. The results of Fondahl, [2]; Roy [3], [4]; IBM [5] and many others have led to the present form of the Precedence Diagram Method (PDM), the prevailing network technique of our times. PDM has hardly changed during the decades in spite of the critiques it has received about its modeling capabilities. One of the ever-returning critiques is about modeling overlapping activities. Proper modeling of overlapping activities seems to be a never-ending debate when traditional precedence relationships are used [6]. For the sake of obviousness: two activities are called overlapping if the successor starts before the finish of the predecessor. The logic in case of overlapping can be defined such as: a) at least x meter distance must be kept between the works carried out at the same moment or b) at least y day must be elapsed between all the points of the activities that correspond to the same location. This logic is hard to model with the traditional precedence relationships (SS, FF, FS and SF) as they are simply not suitable for describing this kind of logic. These relations control only the end-points of the activities and anomalies during the execution phase will not be noticed if relations are satisfied between the related end-points. Different solutions have been proposed using the traditional precedence relations; using negative lag or the combined application of FF and SS relations but fragmentation of activities and developments based on this idea [7] seem to be the best theoretical solution despite the arising practical problems, namely the multiplication of the number of activities and precedence relations. Probably the fragmentation technique has given the idea of connecting the inner points of the activities, which will be discussed in this paper. These point-to-point relations connecting the internal points of the activities seem to be theoretically more suitable for modeling overlapping activities – especially if continuous activities are assumed - as multiple relations are allowed between the activities. This allows planners to control as many points during the execution phase as it thought to be necessary. To the best of our knowledge four seminal partly parallel works on point-to-point relations can be identified in the literature: Kim [8], [9] calls his new relations bee-line relations and the graphical representation Bee-line Diagram (BDM), Francis and Miresco [10], [11] call their new relations temporal functions and they call their graphical representation method chronographic approach. Plotnick [12] calls his method Relationship Diagramming method (RDM) using the term of 'event' for the internal points. Ponce de Leon [13] uses the term Graphical Diagramming Method (GDM) and connected internal points are called embedded nodes.

Despite the differences in terminology and definitions the concept behind all these works is the same. Point-to-point relations connect any two points of the related activities, by defining the connected points and the necessary minimal or maximal time to elapse between these points. (E.g. (100 m, 0 m, 2 days) between activity A and B means that activity B can start 2 days after the finish of the first 100 meter of A.) Points can be the end points of the activities (e.g. finish or start) therefore point-to-point relations can substitute the existing traditional precedence relations (FS, FF, SS, SF). In other words, traditional precedence relations form a subset of point-to-point relations: in these cases the end points of activities are connected. From now on traditional precedence relation will be also referred as end-point relations.

Objective of the study

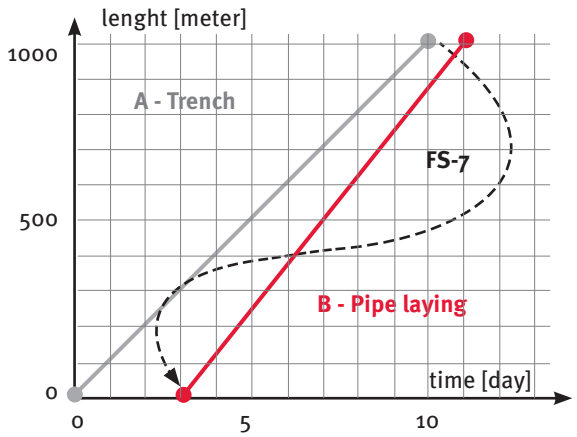
Complete description of point-to-point relations with standardized nomenclature, the mathematical model and the algorithm can be found in Hajdu's work. [14] The goal of this paper is to explore the modeling possibilities of point-to-point relations and compare it with the possibilities of the traditional precedence relations.

Modeling overlapped activities with end-point relations

Using negative lag time for modeling overlapped activities

There is a debate among practitioners whether the use of FS with a negative lag or the combination of SS and FF with the same lag are better for modeling overlapping. Each of them can be criticized; here the method of using negative lag is investigated first. Fig. 1 shows two activities of a pipe laying project in a 1 km long street: trench making (activity A) and pipe laying (activity B). Earthwork is planned for 10 days; pipe laying is planned for 8 days. Pipe laying can overlap with earthwork, however due to some technological or safety reasons at least one day overlapping is necessary to ensure between the two activities. Some planners arguing on the application of FS relation with negative lead such as using FS-7 days relation (see Figure 1a). Now let's suppose that activity A will be accomplished as it was planned, and activity B starts 7 days before the planned finished. Fig 1(b) shows a situation when activity B is going forward with a greater speed compared to the originally planned and finishes on day 8.

This is obviously an impossible situation – pipe laying is taking place where there is no trench (between 600-1000m), however it is allowed because the FS-7 days relation is satisfied. If a situation which is impossible in the real life is accepted in the model, then either the model or the logic is



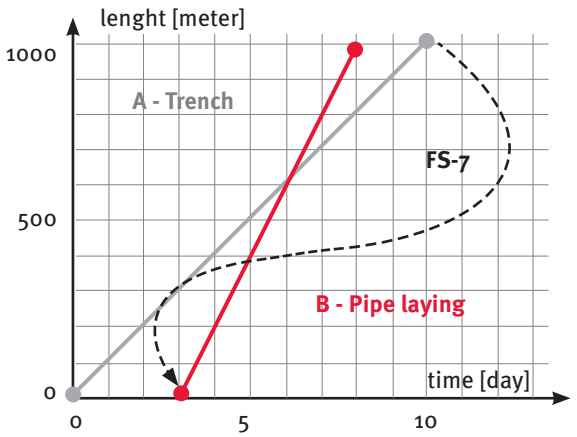


Figure 1: **(a) FS-7 is used to model overlapped activities** **(b) Impossible situation, while FS-7 is satisfied**

wrong. In this very case the applied precedence relation does not reflect the real logic and the planners do not get any warning from the model if such weird situations arise, as precedence relations are satisfied.

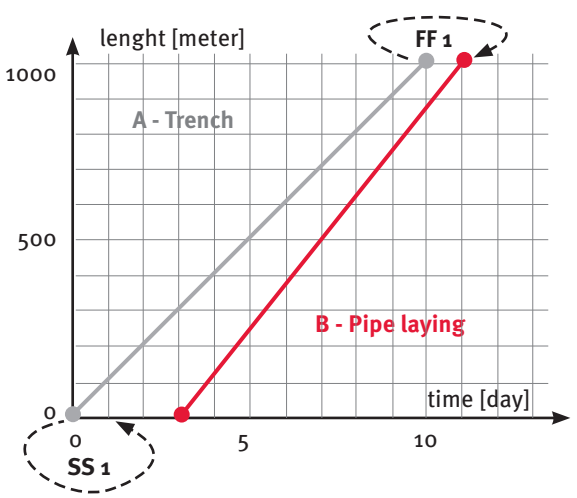
Overlapping with FF and SS relations

Fig. 2 shows another frequently used practice for the same situation. The project is the same, but this time SS1 and FF1 relations are used for describing the logic between the two activities. Let's suppose that the machine that is working on activity A breaks down after a couple of minutes and five days are necessary before it can start again. Having it fixed the work continues with double speed due to the doubled shift applied and finishes according to the plan. Activity A starts as was planned, (SS1 is satisfied!), and goes ahead according to the plan. FF1 will satisfy at the end.

This means that between sections 0-666 m the pipe laying advances the earthwork. This is obviously impossible, but not against the logic described in the network, therefore to model won't send warning signs during the execution phase about this anomaly, therefore the combination of SS and FF relations is not the proper way for modeling overlapping activities. This combination works, and only works if activities are executed with the same intensity without interruptions that is as they were planned. Unfortunately this hypothesis almost never fulfils in the construction industry.

Fragmenting activities

Fragmenting that is dividing activities into small sub-activities and using FS0 relations between them is also a frequently applied practice. From practical points of view this seems to be the best solution that can be achieved with the existing precedence relations. It can be seen on Fig. 3 that this practice divides the related activities into sub-activities in the necessary number and uses FS0 relations between the corresponding segments



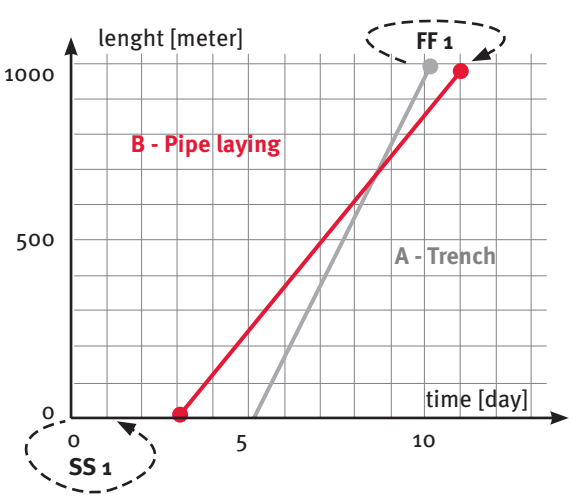


Figure 2: **(a) Planned situation** **(b) Impossible situation that satisfies the relationships**

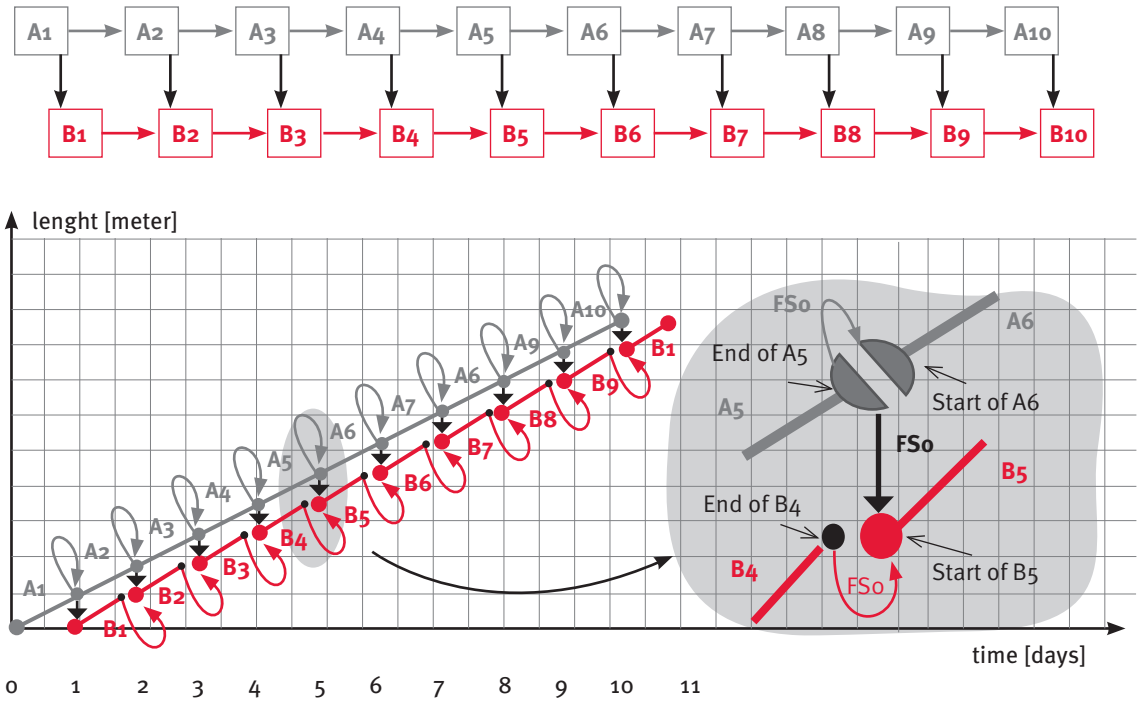


Figure 3: **Modeling overlapping activities using fragmentation and FSo relationships**

(Work on B can start after the first 100 meter is finished on A, after the finish of 200 meter on A the second 100 m can start, etc.) Network representation with Precedence Diagram Method and LSM representation is also shown on Fig. 3 for the same trench-making-pipe-layout project. (All the precedence relations are FSo relations.)

This solution can be acceptable in most of the practical cases. However this solution also has couple of drawbacks:

- Fragmentation increases the number of activities and the number of precedence relations – in this case 20 activities and 28 precedence relations - and this can make the work really uncomfortable in case of large scale projects.
- Fragmentation results in non-continuous B (and non-continuous A too). If continuity is assumed for both activities then maxFSo relations have to add in between the consecutive segments of A (9 relations) and between the consecutive segments of B (another 9 relations) . The result of this is an extra 18 (maximal) precedence relations (altogether 46!), and a time consuming algorithm handling maximal relations, which is not supported by the majority of the scheduling software. (See Fig. 4)
- Fragmentation theoretically is not an acceptable solution for modeling overlapping activities. (There is less than 100 m everywhere except the finish of the segments of A and the start of the segments of B. See Fig. 3). The problem is the same what was described above: only the end points of the segments are controlled.

Point-to-point relations

The Mathematical model

Let a directed acyclic graph be given with one start (s) and one finish node (f). Let **N** = {1,2,...i...j...n} stand for the set of nodes also called activities. **A** will define the set of arcs, also called precedence relations. Point-to-point relations defined later can have minimal or maximal lags, therefore $\mathbf{A}_{min}$ and $\mathbf{A}_{max}$ subsets are introduced for differentiating relations with minimal and maximal lags. In the algorithm, relations with maximal lags will be transformed into relations with minimal lags. In this case **A*** denotes the set of relations . An arbitrary activity i is defined by its start and finish points (S^i and F^i), or by any of the two aforementioned points and its duration d^i. Additional points of the activities can also be defined. P^i_k stands for the k^{th} point of activity i. The relative place of the k^{th} point of activity i is defined by the time span (t^i_k) from the start point of activity i. A relation can be defined between any l^{th} internal point of activity j and any k^{th} internal point of activity i by defining the time that must elapse between the two points ($z^{i\ j}_{k\ l}$). Therefore a point-to-point precedence relation can be defined either by the points and the lag as (P^i_k;P^j_l, $z^{i\ j}_{k\ l}$) or by the relative positions of the points and the lag (t^i_k; t^j_l, $z^{i\ j}_{k\ l}$). Explanation can be seen in Fig. 5.

Using the notations the extended model of PDM can be seen below. The first two conditions tell that all

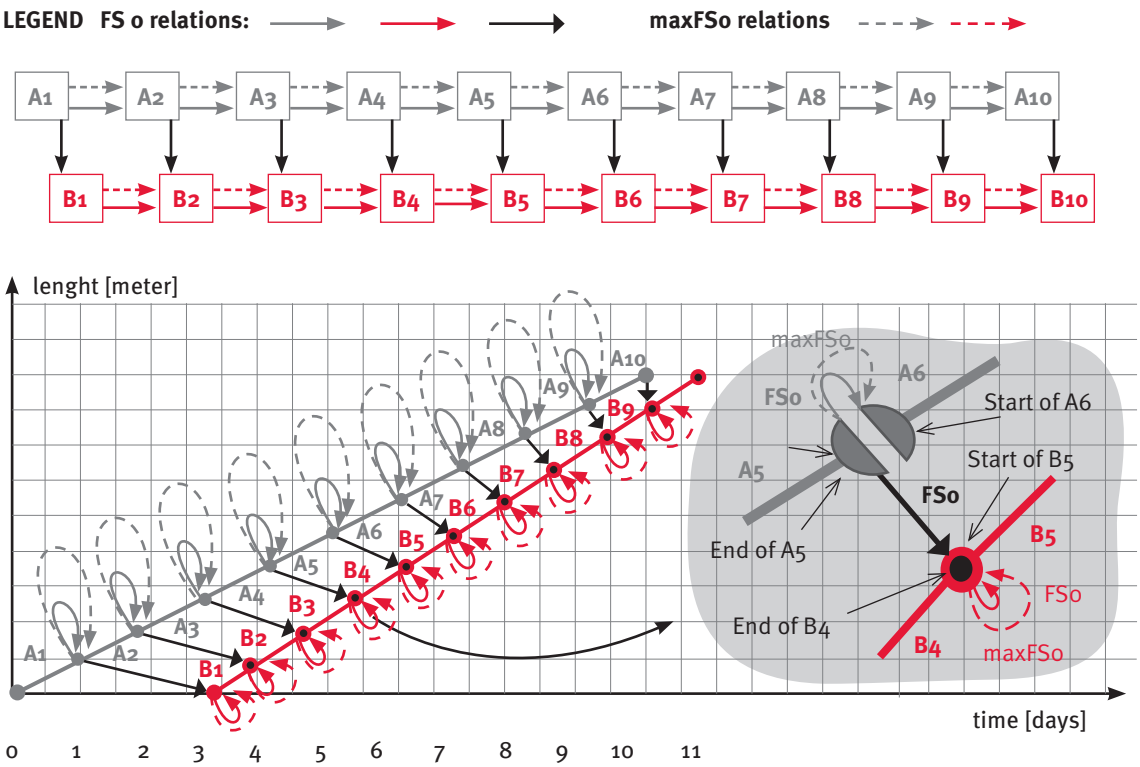


Figure 4: **Modeling overlapped activities using fragmentation technique. Continuity is assumed for both activities**

precedence relations must be satisfied. Eq. 1 describes the precedence relations with minimal lags, while (2) describes the precedence relations with maximal lags.

$$T^j_l - T^i_k \geq z^{ij}_{kl} \qquad \forall (P^i_k ; P^j_l) \in A_{min} \tag{1}$$

$$T^j_l - T^i_k \leq z^{ij}_{kl} \qquad \forall (P^i_k ; P^j_l) \in A_{max} \tag{2}$$

By definition $T^i_k = T^i_s + t^i_k$ and $T^j_l = T^j_s + t^j_l$ therefore (1) and (2) can be modified as:

$$T^j_s - T^i_s \geq z^{ij}_{kl} - t^j_l + t^i_k \qquad \forall (P^i_k ; P^j_l) \in A_{min} \tag{1*}$$

The finish of the activities can be

$$T^j_s - T^i_s \leq z^{ij}_{kl} - t^j_l + t^i_k \qquad \forall (P^i_k ; P^j_l) \in A_{max} \tag{2*}$$

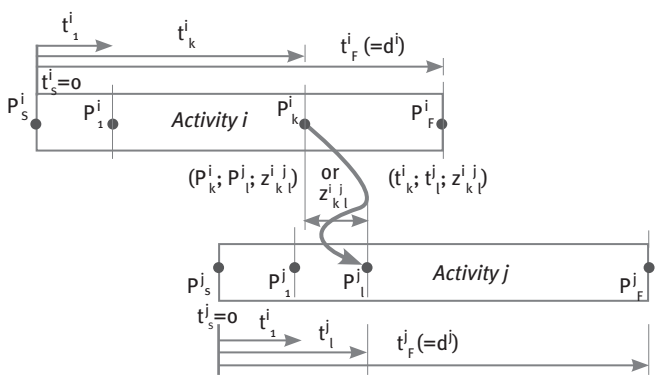


Figure 5: **Explanation of notations and the point-to-point relation (t^i_k; t^j_l; z^{ij}_{kl})**

calculated according to (3) Activities are assumed to be continuous (4). Let's set the start of the project to zero (5).

$$T^i_s + d^i = T^i_F \qquad \forall\, i \in N \tag{3}$$

$$T^i_k - t^i_k = T^i_s \qquad \forall\, i \in N \text{ and } \hat{} P^i_k \quad (P^i_k \text{ exist}) \tag{4}$$

$$T^s_s = o \tag{5}$$

The T policy that satisfies (1*), (2*), (3), (4) and (5) is called a feasible time policy. An infinite number of feasible time policies exist, but the objective of the model is to find that/those time policy/policies where the project duration is the minimum, that is

$$T^f_F - T^s_s \rightarrow \min \quad \text{that is}$$

$$T^f_F - o \rightarrow \min \quad \text{that if } T^f_F \rightarrow \min \tag{6}$$

This model is an LP model, so any LP solver can be used for solving it,

furthermore, based on the simplistic structure of this LP problem different efficient primal dual algorithms can be developed. The solution shown below is based on the modification of the simplistic and widely used CPM/PDM time analysis. This approach is probably the easiest to digest for planning engineers.

Algorithm in case of minimal point-to-point relations

The goal of the algorithm is to find two specific optimal time policies out of the infinite exsisting policies, the earliest and the latest. The earliest optimal time policy is denoted by E^i_S and E^i_F. The latest optimal policy is denoted by L^i_S and L^i_F. The applied algorithm has two phases. The result of the first phase is the earliest optimal time policy, while the result of the second phase is the latest optimal time policy.

Let's suppose for the sake of simplicity that only relations with minimal lags are allowed in the network. In this case, the earliest start and finish of an activity j can only be calculated, if the earliest start dates for all its predecessors are known. As all precedence relations must be satisfied, the early start of a given j can be defined by the maximum of the shifts caused by the preceding relations of activity j, that is:

$$E^j_S = \max \{ E^i_S + t^i_k + z^{ij}_{kl} - t^j_l \quad \forall (P^i_k; P^j_l) \in A_{min} \} \qquad (7)$$

To start, an activity with known predecessors has to be found. In the beginning, only the start activity satisfies this condition: all of its predecessors are known because it does not have any. After these introductory thoughts, the steps of the first phase, that is the steps aiming to find the earliest time policy, can be summarized as follows:

Step 1

Let $E^i_S = -\infty$ and $E^i_F = -\infty \; \forall i \in N$; Let $E^s_S = 0$ Let g:=1

Step 2

REPEAT

g:=g+1

Choose an activity j from the unknowns ($E^i_S = -\infty$) with known predecessors only. IF there is no such activity then GO TO Step 3

$$E^j_S = \max \{ E^i_S + t^i_k + z^{ij}_{kl} - t^j_l \quad \forall (P^i_k; P^j_l) \in A_{min} \};$$

$$E^j_F = E^j_S + d^j$$

UNTIL g=n

Step 3

IF g<n

THEN

STOP

(There is a loop in the network.)

ELSE

$p = E^f_F$

(Project duration is the same as the early finish of the finish activity.)

During the backward pass, the latest optimal time policy will be defined. It is completed by working from the terminal activity to the initial activity in reverse direction of the arrows. It is based on the observation that the late activity times of an activity can only be calculated, if these dates are known for all of its successors. As all successor relations must be satisfied, the late start of a given i can be defined by the maximum of the shifts caused by the succeeding relations of activity i, that is:

$$L^i_S = \min \{ L^j_S + t^j_k - z^{ij}_{kl} - t^i_l \quad \forall (P^i_k; P^j_l) \in A_{min} \} \qquad (8)$$

The rules of the backward pass can be summarized as follows:

Step 1

Let $L^i_S = \infty$ and $L^i_F = \infty \; \forall_i \in N$;

Let $L^{i=F}_S = p - d^f$; Let g:=1

Step 2

REPEAT

g:=g+1

Choose an activity i from the unknowns ($L^i_S = \infty$) with known successors only.

$$\{ L^j_S + t^j_k - z^{ij}_{kl} - t^i_l \quad \forall (P^i_k; P^j_l) \in A_{min} \}; \quad L^i_F = L^i_S + d^i$$

UNTIL g=n

(Note: Loop detection is not necessary during the backward pass.)

Algorithm with mixed (minimal and maximal) lags

Calculations with mixed lags require more computational steps. Maximal relations have to be transformed into minimal relations first. Comparing conditions (1) and (2), it can be seen that the difference between a relation with minimal or maximal lags lies in the direction of the operand. Transforming a relation with maximal lag into a relation with minimal lag requires a simple multiplication by -1.

$$T^j_l - T^i_k \le z^{ij}_{kl} \qquad \forall (P^i_k; P^j_l) \in A_{max} \quad / \; *(-1)(2)$$

$$T^i_k - T^j_l \ge -z^{ji}_{lk} \qquad \forall (P^i_k; P^j_l) \in A_{max} \qquad (9)$$

This is nothing else but a relation from j to i with a negative minimal lag (see Fig. 6).

Traditional precedence relations with maximal lags, and their transformed equivalent minimal lags can be found in Table 1.

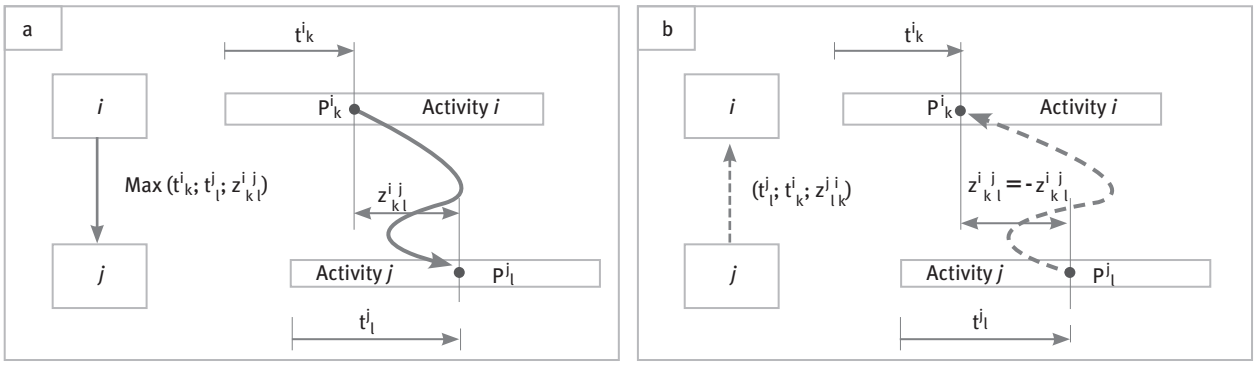


Figure 6: **Point-to-point relation with maximal lag a) and its minimal equivalent b)**

Traditional precedence relations with maximal lags, and their transformed equivalent minimal lags can be found in Table 1.

Traditional precedence relations with maximal lag	Equivalent point-to-point relation with maximal lag	Transformed equivalent precedence relations with minimal lag*	Transformed equivalent point-to-point relation with minimal lag*
maxSSz	max(0; 0; z)	SS-z	(0; 0; -z)
maxFSz	max(d^i; 0; z)	SF-z	(0, d^j; -z)
maxFFz	max(d^i; d^j; z)	FF-z	(d^i; d^j; -z)
maxSFz	max(0, d^j; z)	FS-z	(d^i; 0; -z)

* Transformed equivalents go in the opposite direction

Converting relations with maximal lags into their equivalent minimal relations can result in so-called transformation loops. Fig. 7 shows a loop between B and C.

Table 1: **Traditional precedence relations and their equivalent minimal versions.**

The simple time analysis presented in 4.2 cannot be used in case of loops. One can easily check it by following the activity selection process of the algorithm presented in 4.2. E.g. during the forward pass activity A has to be chosen first. After that none of the activities can be selected, B has two predecessors of which A is known but C is not; C has two predecessors but only A is known while B is not, so the algorithm will stop here.

In case of loops, different algorithms exist to find the longest path. We will use the modified version of the algorithm developed by Bellman [15] and Ford [16] for finding the shortest path between any two points of a cyclic graph. The algorithm is based on the idea that during the forward pass all activities are calculated using the dates of their predecessors even if those have not taken their final dates yet. When all activities have been calculated this way, it has to be checked whether there have been changes in activity dates or not. If the answer is yes, then the entire calculation must be repeated again and again, until we come to the results we had in the previous iteration. In every iteration at least one activity takes its final value, so after maximum n iterations we get the results. Usually much less iterations are necessary. If the nth iteration still brings changes, then the

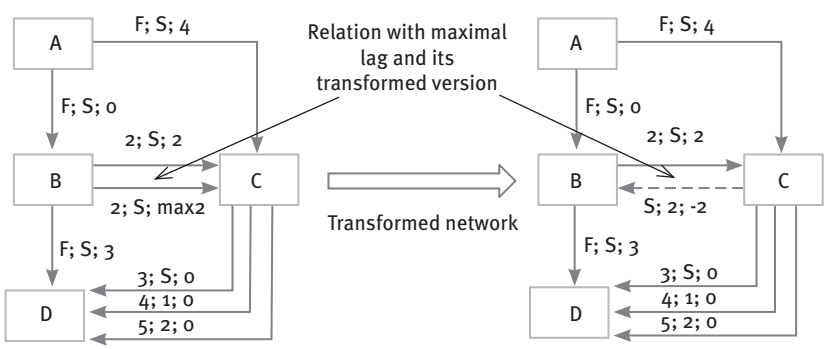


Figure 7: **Transformation of relations with maximal lag into their minimal equivalent can result in loops.**

network cannot be solved. In this case maximal and minimal relations contradict to each other resulting a situation that cannot be solved. E.g. imagine that B can start minimum five days after the finish of A (F;S; 5) but another relation describes that B should start maximum 4 days after the finish of A (F;S;max4).

This obvious contradiction cannot be solved. In this case, the value of the transformation loop will be positive, and the result for the project duration will increase by at least this value in every iteration even after the nth iteration. The steps below summarize the forward pass in case of mixed (minimal and maximal) lags:

Step 1

Let $E^i_s = -\infty$ and OLD $E^i_s = -\infty$

$\forall\ i \in N$; Let $E^{j=s}_s = 0$ and

OLD$E_^{j=s}_s = 0$; Let *h*=0;

Let *No_of_Iter*=0

Step 2

REPEAT

There_were_changes:=FALSE;
No_of_Iter: = No_of_Iter+1

REPEAT

h:=h+1

Select any *j* activity that has not been selected in this iteration yet

$E^j_s = \max\{\text{OLD\_}E^j_s;$

$(E^i_s + t^i_k + z^{i\ j}_{k\ l} - t^j_l \quad \forall (P^i_k;P^j_l) \in A^*)\};$

$E^j_F = E^j_s + d^j$

IF

$E^j_s > \text{OLD\_}E^j_s$

THEN *There_were_changes:=TRUE*

UNTIL *h=n*

*Let OLD_* $E^i_s = E^i_s\ \forall i \in N$

UNTIL No_of_iter>n or There_were_changes:=FALSE

Step 3

IF No_of_Iter>n THEN There is no solution.
(There is a loop with positive value.)
IF *There_were_changes:=FALSE THEN* we arrived to a feasible optimal time policy. (All activity dates remained unchanged after two iterations.)

The rules of backward pass can be summarized as follows:

Step 1

Let $L^i_s = -\infty$ and OLD_$L^i_s = -\infty$

$\forall\ i \in N$; Let $L^{j=s}_s = E^{j=s}_s$ and

OLD_$L^{j=s}_s = L^{j=s}_s$; Let *h=n*;

Step 2

REPEAT

There_were_changes:=FALSE;

REPEAT

Select any i activity that has not been selected in this iteration yet

$L^i_s = \min\{\text{OLD\_}L^i_s;\ (L^j_s + t^j_k - z^{i\ j}_{k\ l} - t^i_l \quad \forall (P^i_k;P^j_l) \in A^*)\};$

$L^i_F = L^i_s + d^i$

IF $L^i_s > \text{OLD\_}L^i_s$

THEN *There_were_changes:=TRUE*

h:=h − 1

UNTIL h=0

Let OLD_$L^i_s = L^i_s\ \forall_i \in N$

UNTIL There_were_changes:=FALSE

Notes to the algorithm:

- Loop detection was done during the forward pass, so there is no need for that during the backward pass.
- Any order of activities can be used during the algorithm, which can largely modify the number of iterations. Here we used the ascending order of activities during the forward pass and the descending order during the backward pass. In the optimal case, the first iteration presents the results and the second iteration will validate this, in the pessimistic case, n+1 iterations are necessary.
- Following the optimal order of the forward pass will be the worst during the backward pass, and vice versa.

Sample project: only minimal relations are allowed

A small sample project is shown in Fig. 8 a) consisting only mimimal relations. Results can be seen in Figure 8 b). Calculations can be tracked below.

Forward pass:

Only activity A can be selected.

$E^A_S = 0;\ E^A_F = E^A_S + d^A = 0 + 6 = 6$

Only activity B can be selected.

$E^B_S = \{(E^A_S + t^A_k + z^{A\ B}_{3\ 0} - t^B_0)\} =$

$\{(0+3+0-0)\} = 3;$

$E^B_F = E^B_S + d^B = 3+6 = 9$

Only activity C can be selected.

$E^C_S = \max\{(E^A_S + t^A_6 + z^{A\ C}_{6\ 0} -$

$t^C_0);(E^B_S + t^B_2 + z^{B\ C}_{2\ 0} - t^C_0)\}$

$=\{(0+6+4-0);(3+2+2-0)\} = 10;$

$E^C_F = E^C_S + d^C = 15$

Only activity D can be selected.

$E^D_S = \max\{(E^B_S + t^B_6 + z^{B\ D}_{6\ 0} - t^D_0);$

$(E^C_S + t^C_3 + z^{C\ D}_{3\ 0} - t^D_0);$

$(E^C_S + t^C_4 + z^{C\ D}_{4\ 1} - t^D_1);$

$(E^C_S + t^C_5 + z^{C\ D}_{5\ 2} - t^D_2)\} =$

$\{(3+6+3-0);(10+3+0-0);$

$(10+4+0-1);(10+5+0-2)\} =$

$\{12;13;13;13\} = 13;$

$E^D_F = E^D_S + d^D = 17$

Early dates for all activities have been calculated, the forward pass is finished. (See Fig. 8)

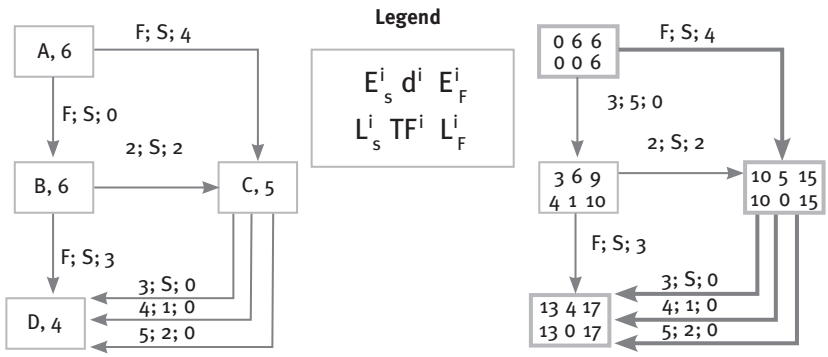


Figure 8:
a) Sample project with minimal lags. **b) Results of the calculations.**

Backward pass:

Only activity *D* can be selected.

$L^D_F=17;\ L^D_S=L^D_F-d^D=17-4=13$

Only activity C can be selected.

$L^C_S=\min\{(L^D_S+t^D_2-z^{C\ D}_{5\ 2}-t^C_5);$

$(L^D_S+t^D_1-z^{C\ D}_{4\ 1}-t^C_4);\ (L^D_S+t^D_0-$

$z^{C\ D}_{3\ 0}-t^C_3)\}=\min\{(13+2-0-5);$

$(13+1-0-4);$

$(13+0-0-3)\}=\min$

$\{(10;10;10)\}=10$

$L^C_F=L^C_S+d^c=10+5=15$

Only activity *B* can be selected.

$L^B_S=\min\{(L^D_S+t^D_0-z^{C\ D}_{6\ 0}-$

$t^C_6);(L^C_S+t^C_0-z^{B\ C}_{2\ 0}-t^B_2)=$

$\min\{(13+3-3-6);$

$(10+0-2-2)\}=\min\{(4;6)\}=4$

$L^B_F=L^B_S+d^B=4+6=10$

Only activity *A* can be selected.

$L^A_S=\min\{(L^B_S+t^B_0-z^{A\ B}_{3\ 0}-t^A_3);$

$(L^C_S+t^C_0-z^{A\ C}_{6\ 0}-t^A_6)=\min$

$\{(4+0-0-3);(10+0-4-6)\}$

$=\min\{(1;0)\}=0$

$L^A_F=L^A_S+d^A=0+6=6$

Late dates for all activities have been calculated, the backward pass is finished. (See Fig. 8)

Sample project: both minimal and maximal relations are allowed

A small sample project is shown on Fig. 9. a) consisting of relations with minimal and maximal lags. The network with the transformed maximal relation and with the results is shown in Fig. 9. b). Due to the transformation loop, the iterative algorithm has to be used.

Any order of the activities can be used. Here we use the A;B;D;C order for the forward pass. The calculations can be followed on Figure 10. Boxes of those activities that have been changed during the iteration are filled with grey.

Iteration #1

$E^A_S=\max\{OLD\_E^A_S\}=0;$

$E^A_F=E^A_S+d^A=0+6=6$

$E^B_S=\max\{OLD\_E^B_S;$

$(E^A_S+t^A_3+z^{A\ B}_{3\ 0}-t^B_0);$

$(E^C_S+t^C_0+z^{C\ B}_{0\ 2}-t^B_2)\}=$

$\{-\infty;(0+3+0-0\};$

$(-\infty+0-2-2)\}=3$

$E^B_F=E^B_S+d^B=9$

$E^D_S=\max\{OLD\_E^D_S;$

$(E^B_S+t^B_6+z^{B\ D}_{6\ 0}-t^D_0);$

$(E^C_S+t^C_3+z^{C\ D}_{3\ 0}-t^D_0);$

$(E^C_S+t^C_4+z^{C\ D}_{4\ 1}-t^D_1)$

$(E^C_S+t^C_5+z^{C\ D}_{5\ 2}-t^D_2)\}=$

$\{-\infty;(3+6+3-0\};$

$(-\infty+3+0-0);$

$(-\infty+4+0-1);$

$(-\infty+5+0-2)\}=12$

$E^D_F=E^D_S+d^D=16$

$E^C_S=\max\{OLD\_E^C_S;$

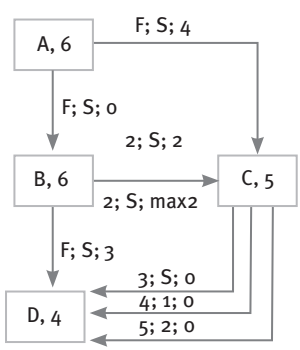
$(E^B_S+t^B_2+z^{B\ C}_{2\ 0}-t^C_0);$

$(E^A_S+t^A_6+z^{A\ C}_{6\ 0}-t^C_0)=$

$\{-\infty;(3+2+2-0\};$

$(0+6+4-0)\}=10$

$E^C_F=E^C_S+d^C=15$



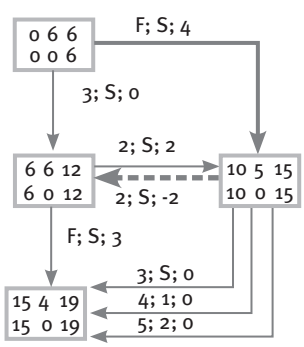


Figure 9:
a) Sample project with mixed lags **b) Transformed network with the results**

Iteration #2

E^A_S= max { OLD_E^A_S}=0;

E^A_F=E^A_S+d^A=0+6=6

E^B_S=max {OLD_E^B_S;

(E^A_S+t^A_3+$z^{A\ B}_{3\ 0}$−t^B_0);

(E^C_S+t^C_0+$z^{C\ B}_{0\ 2}$−t^B_2)}=

{3;(0+3+0−0};

(10+0−2−2)}=6

E^B_F=E^B_S+d^B=12

E^D_S=max {OLD_E^D_S;

(E^B_S+t^B_6+$z^{B\ D}_{6\ 0}$−t^D_0);

(E^C_S+t^C_3+$z^{C\ D}_{3\ 0}$−t^D0);

(E^C_S+t^C_4+$z^{C\ D}_{4\ 1}$−t^D_1)

(E^C_S+t^C_5+$z^{C\ D}_{5\ 2}$−t^D_2)}=

{12; (6+6+3−0}; (10+3+0−0);

(10+4+0−1); (10+5+0−2)}=15

E^D_F=E^D_S+d^D=19

E^C_S=max {OLD_E^C_S;

(E^B_S+t^B_2+$z^{B\ C}_{2\ 0}$−t^C_0);

(E^A_S+t^A_6+$z^{A\ C}_{6\ 0}$−t^C_0)=

{10;(6+2+2−0};

(0+6+4−0)}=10

E^C_F=E^C_S+d^C=15

Iteration #3

E^A_S= max { OLD_E^A_S}=0;

E^A_F=E^A_S+d^A=0+6=6

E^B_S=max{OLD_E^B_S;

(E^A_S+t^A_3+$z^{A\ B}_{3\ 0}$−t^B_0);

(E^C_S+t^C_0+$z^{C\ B}_{0\ 2}$−t^B_2)}=

{3;(0+3+0−0};

(10+0−2−2)}=6

E^B_F=E^B_S+d^B=12

E^D_S=max{OLD_E^D_S;

(E^B_S+t^B_6+$z^{B\ D}_{6\ 0}$−t^D_0);

(E^C_S+t^C_3+$z^{C\ D}_{3\ 0}$−t^D_0);

(E^C_S+t^C_4+$z^{C\ D}_{4\ 1}$−t^D_1)

(E^C_S+t^C_5+$z^{C\ D}_{5\ 2}$−t^D_2)}=

{12; (6+6+3−0};

(10+3+0−0);

(10+4+0−1); (10+5+0−2)}=15

E^D_F=E^D_S+d^D=19

E^C_S=max{OLD_E^C_S;

(E^B_S+t^B_2+$z^{B\ C}_{2\ 0}$−t^C_0);

(E^A_S+t^A_6+$z^{A\ C}_{6\ 0}$−t^C_0)=

{10;(6+2+2−0};

(0+6+4−0)}=10

E^C_F=E^C_S+d^C=15

No activity dates have changed in the course of iteration #3; therefore the forward pass is finished. For the backward pass the D; B; C; A sequence is selected. Calculations can be followed below.

Iteration #1

L^D_S=min {OLD_L^D_S; (nil)}=15;

L^D_F=L^D_S+d^D=19

L^B_S=min {OLD_L^B_S;

(L^D_S+t^D_0−$z^{B\ D}_{6\ 0}$−t^B_6);

(L^C_S+t^C_0−$z^{B\ C}_{2\ 0}$−t^B_2)}=min{∞;

(15+0−3−6);

(∞+0−2−2)}=6

L^B_F=L^B_S+t^B=12

L^C_S=min {OLD_L^C_S;

(L^B_S+t^B_2−$z^{C\ B}_{0\ 2}$−t^C_0);

(L^D_S+t^D_0−$z^{B\ D}_{3\ 0}$−t^B_3);

(L^D_S+t^D_1−$z^{B\ D}_{4\ 2}$−t^B_4)

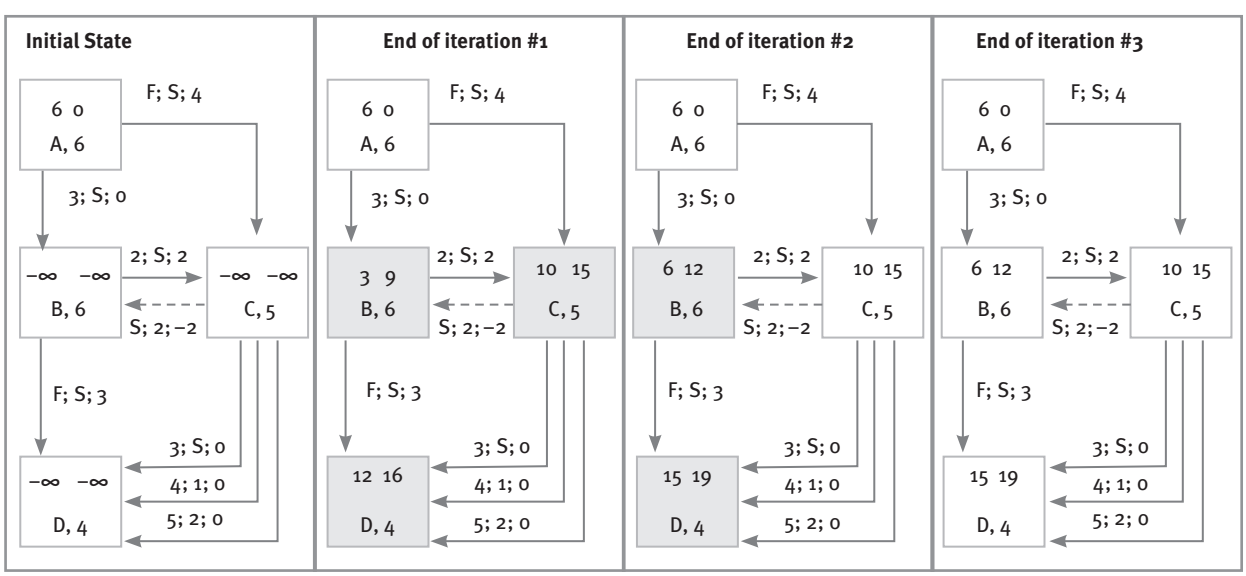


Figure 10: **Forward pass computation.**

$(L^D_S+t^D_2-z^{B\ D}_{5\ 2}-t^B_5)\}=$

$\min\{\infty;(6+2-(-2)-0);$

$(15+0-0-3);$

$(15+1-0-4);(15+2-0-5)\}=10$

$L^C_F=L^C_S+t^C=15$

$L^A_S=\min\{OLD\_L^A_S;$

$(L^C_S+t^C_0-z^{A\ C}_{6\ 0}-t^A_6);$

$(L^B_S+{}^B_{t\ 0}-z^{A\ B}_{3\ 0}-t^A_3)\}=$

$\infty\ \min\{;(10+0\text{-}4\text{-}6);(6+0\text{-}$

$0\text{-}3)\}=0\quad L^A_F=L^A_S+t^A=6$

Iteration #2

$L^D_S=\min\{OLD\_L^D_S;\ (nil)\}=15;$

$L^D_F=L^D_S+d^D=19$

$L^B_S=\min\{OLD\_L^B_S;$

$(L^D_S+t^D_0-z^{B\ D}_{6\ 0}-t^B_6);$

$(L^C_S+t^C_0-z^{B\ C}_{2\ 0}-t^B_2)\}=\min$

$\{6;(15+0-3-6);$

$(10+0-2-2)\}=6$

$L^B_F=L^B_S+t^B=12$

$L^C_S=\min\{OLD\_L^C_S;$

$(L^B_S+t^B_2-z^{C\ B}_{0\ 2}-t^C_0);$

$(L^D_S+t^D_0-z^{B\ D}_{3\ 0}-t^B_3);$

$(+t^D_1-z^{B\ D}_{4\ 2}-t^B_4)$

$(L^D_S+t^D_2-z^{B\ D}_{5\ 2}-t^B_5)\}=$

$\min\{10;(6+2-(-2)-0);$

$(15+0-0-3);$

$(15+1-0-4);$

$(15+2-0-5)\}=10$

$L^C_F=L^C_S+t^C=15$

$L^A_S=\min\{OLD\_L^A_S;$

$(L^C_S+t^C_0-z^{A\ C}_{6\ 0}-t^A_6);$

$(L^B_S+t^B_0-z^{A\ B}_{3\ 0}-t^A_3)\}=\min$

$\{0;(10+0\text{-}4\text{-}6);(6+0\text{-}0\text{-}3)\}=0$

$L^A_F=L^A_S+t^A=6$

Activity dates have not changed during the iteration. Calculations are finished. Results of the backward pass (and the forward's as well) can be seen in Fig. 10. b.

Modeling overlapped activities with point-to-point relations

Probably the fragmentation technique has given the idea of connecting the inner points of the activities. The common characteristic of the point-to-point relations is that any points of the related activities can be connected. (E.g. B can start after the finish of the first 100 meter is finished on activity A; the second 100 meters on B can start as the first 200 meters of A has finished etc.; or B can start after the finish of the first day work on A, the second day work on B can start after the finish of the second day work on A, etc. These are practically the same relations that are used at the fragmentation technique but without fragmenting activities. As it was shown earlier three data is necessary to define a point-to-point relation: the definition of the connected points, and the lag time between the points. Points can be defined by their distance from the start point of the activities using time and volume units. (e.g. (100 m; 0 m; 2 days) means that the successor can start 2 days after the finishing of the first 100 meters on the predecessor, (2 days, 1 day, 0 day) means that after finishing the second day work on the predecessor the second day work on the successor can start.) Any arbitrary points of the connected activities can be defined even the start or finish points. This means that point-to-point relations can be used to define the traditional precedence relations as well. Table 1 shows the traditional precedence and their equivalent point-to-point relations.

Figure 11 shows our sample project using point-to-point relations with zero lag. Instead of 20 activities and 46 precedence relations only 2 activities and 10 point-to-point precedence relations are necessary, a considerably smaller amount of data for the same logic. Calculations with point-to-point relations are very similar to those that are used in case of traditional end-point relations. The only difference is the function applied in calculating the early/late dates based on the predecessor/successor activities.

Problems with point-to-point relations

Point-to-point relations are better in modeling overlapped activities than traditional end-point relations. Its advantage compared to traditional relations will be apparent during to

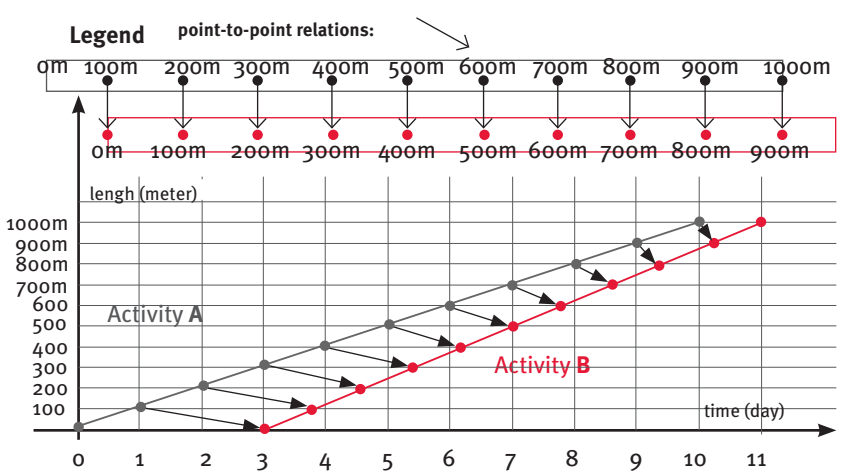


Figure 11. **Modeling overlapped activities with point-to-point relations**

Traditional precedence relations	Equivalent point-to-point relations between activity *i* and *j* (point are defined as volume (V))	(points are defined as time)
Start-to-Start-*t* *(SSt)*	(0 ; 0 ; *t*)	(0; 0; *t*) or (S; S; *t*)
Finish-to-Start-*t* *(FSt)*	(V^i ; 0 ; *t*)	(d^i ; 0 ; *t*) or (F; S; *t*)
Finish-to-Finish-*t* *(FFt)*	(V^i ; V^j ; *t*)	(d^i ; d^j ; *t*) or (F; F; *t*)
Start-to-Finish-*t* *(SFt)*	(0 ; V^j ; *t*)	(0 ; d^j ; *t*) or (S; F; *t*)

* V^i and V^j are the quantity of work on activity *i* and *j*, d^i and d^j are the durations of activity *i* and *j*

Table 2: **Point-to-point relations can be used instead of traditional precedence relations**

control of the execution phase. While end-point relations control only the end-points and due to this no warning message is received in case of anomalies, point-to-point relation can control as many points as it thought to be necessary. But it can still not be enough! Figure 12. shows a situation which can easily arise during the execution phase. In the course of the second and third day the distance between A and B is less than the necessary safety distance. However, no warning will come from the system – even if execution data are monitored in every hour – because precedence relations are satisfied.

Discussions and further research

Modeling capabilities of traditional (end-point) precedence relations have been compared with recently developed point-to-point precedence relationship. It has been shown that the existing practices based on traditional precedence relations are theoretically wrong. Point-to-point relations and fragmentation assuming continuity are identical from theoretical point of view, but point-to-point relations require much less input data.

It also has been shown that the traditional precedence relations (SS, SF, FS and FF with either minimal or maximal lags) could be derived from the new relation. Point-to-point relations affect the very fundaments of network techniques, therefore all definitions, generalizations, problems based on the 'old' precedence relations must be checked and modified accordingly, if necessary, including the definitions and calculations of floats, the definition of the critical path, the classification of critical activities [17], [6], [18], the algorithms for resource optimization etc. To our best knowledge, this work has not been done yet.

The problem with point-to-point relations has been discussed in the previous section. This problem can be solved by increasing the number of segments and the point-to-point relations between activities. Controlling all the corresponding points between the related activities requires an infinite number of segments and infinite number of point-to-point relations. This leads to a new type of precedence relation which can be called as continuous precedence relation.

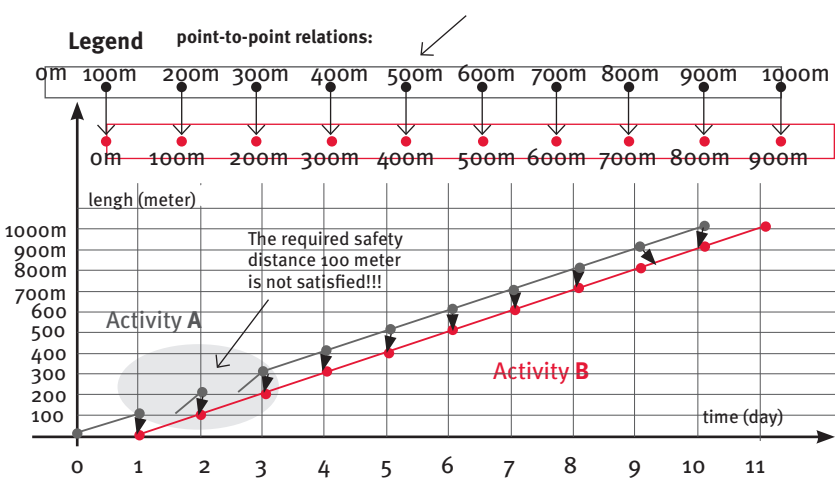


Figure 12. **Point-to-point relations can also lead to anomalies**

References

[1] Hajdu, M., 2015. Point-to-point Versus Traditional Precedence Relations for Modeling Activity Overlapping, Procedia Engineering Vol 123 pp 208-215 2015. ISSN: 1877-7058 doi:10.1016/j.proeng.2015.10.079

[2] Fondahl, J.W 1961. A non-computer approach to the critical path method for the construction industry. Technical Report #9 The Construction Institute, Department of Civil Engineering, Stanford University, Stanford California

[3] Roy, G.B., 1959 Théorie des Graphes: Contribution de la théorie des graphes á l1 étude de certains problémes linéaries, Comptes rendus des Séances de l1 Acedémie des Sciences. séence du Avril 1959, s2437-2449, 1959

[4] Roy, G.B. 1960. Contribution de la théorie des graphes à l'étude de certains problems d'ordonnancement", Comptes rendus de la 2ème conférence internationale sur la recherché opérationnelle, Aix-en-Provence, English Universities Press, Londres 171-185.

[5] IBM 1964. Users Manual for IBM 1440 Project Control System (PCS). 1964

[6] Douglas, E.E.; Calvery T. T. McDonald, D.F.; Winter, R.M. 2006. The Great Negative Lag Debate. 2006 AACE International Transactions PS 02.01.- PS 02.07

[7] Hajdu, M. 1996. Network Scheduling Techniques For Construction Project Management. Kluwer Academic Publishers, ISBN 0-7923-4309-3

[8] Kim, S. 2010. Advanced Networking Technique Kimoondang, South Korea 2010

[9] Kim, S. 2012. CPM Schedule Summarizing Function of the Beeline Diagramming Method. Journal of Asian Architecture and Building Engineering, 11(2) November 2012; 367-374

[10] Francis, A., Miresco, E.T. 2000. Decision Support for Project Management Using a Chronographic Approach. Proceedings of the 2nd International Conference on Decision Making in Urban and Civil Engineering, 2000 Lyon, France, 845-856.

[11] Francis, A., Miresco, E.T. 2002. Decision Support for Project Management Using a Chronographic Approach. Journal of Decision Systems, Special issue JDS-DM in UCE: Decision Making in Urban and Civil Engineering, 11(3-4): 383-404.

[12] Plotnick FL, 2004 Introduction to Modified Sequence Logic, Conference Proceedings, PMICOS (first annual) Conference, April 25, 2004, Montreal, Canada

[13] Ponce de Leon, G. 2008 Graphical Planning method. PMICOS Annual Conference, Chicago, IL, 2008

[14] Hajdu, M., One relation to rule them all: The point-to-point precedence relation that substitutes the existing ones. 5th International/11th Construction Specialty Conference 5e International/11e Conférence spécialisée sur la construction, Vancouver, British Columbia June 8 to June 10, 2015 / 8 juin au 10 juin 2015

[15] Bellman, R.E. 1958 On a Routing Problem. Quarterly of Applied Mathematics 16 1959. 87-90

[16] Ford, L.R. 1956. Network Flow Theory. The Rand Corporation 1956.

[17] Wiest, J.D. 1981 Precedence diagramming method: Some unusual characteristics and their implications for project managers, Journal of Operations management, Volume 1/3 Feb. 1981. 121-130

[18] Valls, V and Lino, 2001. Criticality Analysis in Activity-on-Node Networks with Minimal Time Lags. Annals of Operations Research, 102(1-4) 17-37