AN ACTIVITY AND ARROW ARRANGING ALGORITHM FOR CLARITY IN SCHEDULE NETWORK DIAGRAMS

Gunnar Lucko[1]

ABSTRACT

Quality schedule network diagrams are characterized by containing the entire time and logic information of the schedule in a highly legible format. This working paper presents an algorithm with which schedulers can improve the clarity of the graphical representations of schedule networks that they create. The algorithm has been designed to emulate the approach that a human scheduler takes when being asked to prepare a manually drawn activity-on-the-node diagram. Activities are automatically placed into a grid so that closely dependent activities are located near each other and the number of logic link crossings within the schedule network is kept low. An illustrated step-by-step example assists with using the algorithm, which itself is written in intuitive pseudo-code. Various considerations for good graphics design of schedule are provided. Under ongoing research into schedule network analysis and optimization it is planned to investigate more powerful arranging algorithms and to develop a software implementation.

KEY WORDS

Network schedules, algorithm, logic links, arranging, graphics design

INTRODUCTION

In past decades, most experienced schedulers learned their trade in the old-fashioned way with paper and pencil to create activity lists for projects, draw bar charts, activity-on-the-arrow (AOA) and activity-on-the-node (AON, also called precedence diagramming method, PDM) diagrams – which do not require "dummy" links to correctly display the logic and have gained much popularity over AOA – and to perform the forward pass, backward pass, and float calculations as per the widely known Critical Path Method (CPM). Drafting skills for manually drawn diagrams were included in their education and equipped these professionals with a sense of the visual clarity needed to achieve quality drawings.

Nowadays, schedulers have powerful tools at their disposal in form of commercial scheduling software packages that are used to store and analyze the vast amounts of information describing the time, logic, and resource-related aspects of construction projects. Scheduling software is routinely used both during the planning and during the execution stages to ensure the smooth flow of operations by developing an optimum sequence of activities, coordinating among their required resources, and reacting to any changes that

[1] Assistant Professor, Department of Civil Engineering, Pangborn Hall Room G-17, The Catholic University of America, Washington, DC 20064, Phone +1 202/319-4381, FAX 202/319-6677, lucko@cua.edu

might be encountered. Since schedule diagrams that are used in practice are not carefully drawn by hand anymore but rather are generated by a machine, this paper examines their graphical quality with a particular focus on displaying the schedule logic.

SOFTWARE-GENERATED SCHEDULES

Any graphical representation of a schedule includes two basic elements, time information and logic information (Lucko 2005). Schedules are composed of discrete activities of a specified duration which are connected by logic links that represent the dependency structure and sequence over time of said activities. Users enter this input in software packages such as Microsoft® Project, Primavera Project Planner® (P3®), or Primavera SureTrak® Project Manager. The software automatically generates bar charts and AON diagrams from the activity list. Some scheduling software packages use a different terminology, such as e.g. referring to AON diagrams as PERT, which actually denotes the Program Evaluation and Review Technique, a form of scheduling calculations developed by the U.S. Navy that uses probabilistic duration estimates rather than fixed durations for each activity. The following sections discuss the basic diagram types generated by scheduling software.

BAR-CHARTS

Bar charts, also known as Gantt charts after their 1917 inventor Henry L. Gantt, are the most widely known form of graphical schedule representation. They feature a horizontal time axis and a vertical activity list. Each activity is shown in the diagram area as a bar whose width is directly proportional to its duration, making them fully time-scaled. Despite their widespread use bar charts suffer from the severe setback that they focus almost exclusively on activities and only imply the logic indirectly, whether through consecutive naming and numbering or by the "stepped-down" arrangement of the bars that may indicate a dependency. In other words, whether an activity further down the list and further over to the right of the time axis is dependent on the activities before it on the list and on the time axis remains unknown. Such information may be derived with experience and knowledge of the actual construction process that is captured in the schedule, yet uncertainty remains.

Fenced bar charts as introduced by Melin and Whiteaker (1981) in their seminal paper curiously have never caught on in the scheduling community and among the software developers as strongly as their intuitive display of logic might suggest. Fenced bar charts augment the traditional plain bar (or Gantt) charts with vertical solid "fences" and horizontal flexible "links" that together capture the entire logic information. Lucko (2005) has argued to revive this mechanistic view and its intuitive understanding. Basu (1990) has described how a physical model of a schedule can be built for teaching purposes on an abacus-like calendar board where wooden blocks represent activities that can "push" subsequent activities further down in time if a delay occurs and where the logic is captured by wire fences with fixed intermissions (e.g. for curing concrete) or with springs (Lucko 2005) showing free float.

Software-generated bar charts show logic links – if they are displayed whatsoever – as arrows without distinguishing "soft" and "hard" links as fenced bar charts do. Their layout is oriented toward the activity bars and the arrows are squeezed in between, in some cases overlapping so that exact dependencies are no longer recognizable, and even running in a

zigzag between directly succeeding activities due to a lack of space for drawing the head of the arrow. Lucko (2005) has provided vivid examples of a bar chart and of an AON diagram that were generated with a scheduling software package in its normal printout resolution to illustrate how poor default layout can falsely imply incorrect definitions of start and finish dates, multiple non-existing dependencies, and even negative float.

ACTIVITY-ON-THE-NODE DIAGRAMS

As mentioned earlier, AON diagrams have become the predominant form of schedule network diagrams that are used in practice. The activities are represented by rectangles or circles and the logic links are arrows between the activities showing their dependencies. Activities all have the same size regardless of their actual duration. In some software applications a calendar bar is provided along the horizontal axis that indicates what range of dates an activity falls into. While AON diagrams thus are not fully time-scaled, they still show the sequence of activities over time from left to right.

In AON diagrams generated by scheduling software the activities typically are arranged on a tight grid where few positions are left empty. Arrows are drawn in the narrow spaces between activities and can be customized to be either straight, shared, or separate, yet may still overlap each other so that the actual dependency structure is hidden or hard to decipher (Lucko 2005). The layout further depends on the number of activities that are displayed in the rows and columns fitting onto one printout page as chosen by the user.

Compare this with hand-drawn AON diagrams, where the scheduler can arrange activities based on the CPM calculations such that e.g. activities that directly succeed each other in time and dependency are placed in succeeding grid columns, the critical path may be one central axis of activities around which auxiliary ones are grouped, and where the diagram displays the entire time and logic information with such graphical clarity that it is indeed a helpful visualization of the numerical CPM results. It is the purpose of this paper to provide an account of such best practices and to capture the graphical intuition of professional schedulers in a simple algorithms that assists with creating quality schedule diagrams. The following section gives graphics design guidelines to be considered when drawing a schedule diagram with its time and logic elements and annotations.

FONT AND COLOR IN SCHEDULE DIAGRAMS

Serif type fonts, e.g. Times New Roman, generally are more legible than sans-serif type fonts, e.g. Arial. Mixed capital and lower case letters are easier to read than all capitals or all lower case text. The reason for this lies in the former ones having more distinctive characteristics for the human eye than the latter ones. Common use in business applications has established the 12-point font size as the standard size and it is strongly recommended to not use any fonts smaller than 8-point in printed matters. It depends on the actual use and printout size of a schedule diagram which font sizes are appropriate, whether being e.g. for a folded letter-size printout that is part of a submittal or e.g. for a wall-sized chart on which the planning team places note stickers to develop an optimum sequence and resource allocations. While it is common practice to abbreviate activity descriptions in scheduling software applications, the shortened names should still be sufficiently long to be self-explanatory to a

user unfamiliar with the project without requiring the use of a legend. On the other hand, annotations in the body of the diagram or in the activity list should remain short and unobtrusive enough to not distract from the main information of the schedule – time and logic – in its graphical form.

Color can enhance schedules to mark important activities or logic links. Its most common application is to highlight the critical path in red and to distinguish different types of activities or the resources that they require. Whichever its use, conscious choice of a color scheme will contribute to creating schedules that quickly and easily provide their user with rich information. Color itself has the aspects of hue (color), saturation (intensity), and value (darkness). Tufte (1989 and 2001), discussing maps, recommends using color schemes derived from nature, i.e. shades of pastel tones rather than glaring neon colors that drown any non-highlighted pieces of information in their vicinity. Contrasting or dark colors, possibly combined with negative fonts (light letters on dark background), are often found in the calendar bar of scheduling software applications. They distract the eye from the body of the diagram and its actual information-loaded elements unless used with care and restraint.

SPACE AND SYMBOLS IN SCHEDULE DIAGRAMS

One of the most important considerations for any diagram is how much of its space is dedicated to actual information. Since schedules have to communicate two basic types of information, time and logic, the scheduler needs to decide how much of the available diagram area is allocated to each one of them. Lucko (2005) has shown that for a typical software-generated AON diagram there exists a discrepancy by one order of magnitude between showing time and logic. In other words, the area dedicated to the time information (containing an identifier, the duration, and the early and late start and finish dates) is 10 times the area dedicated to the logic information (linking predecessors with successors).

At the same time, space should not be wasted in a schedule diagram. Tufte's (2001) data density, defined as the "number of entries in data matrix" divided by "area of data graphic" should be maximized so that much information is communicated in the given diagram area. To avoid clutter and focusing too much on activities and too little on the arrows that connect them, this author recommends that the data density be chosen approximately equal for time and logic, i.e. the overall area is balanced between them. The scheduler should particularly ensure that sufficient space is available to funnel the arrows between the activities. It is further recommended that only essential information be provided in the body of the diagram. Extra lines or borders in the activity rectangles can be omitted if the layout of the information itself is intuitive. The duration is placed horizontally between the early and late start and finish dates to represent their difference when read from the left to the right. The total float is placed vertically between the early and the late dates to represent their difference when read from the top to the bottom. Tufte's (2001, p96 and p162) data-ink ratio, defined as the ink used to display the data divided by the "total ink used to print the graphic" should be maximized for the schedule diagram. In other words, any superfluous lines and extra decorations of the kind that he has elaborately termed chartjunk should be avoided so that they do not distract from the key message.

Arrows are used to display start-to-start, start-to-finish, finish-to-start, and finish-to-finish relationships between activities. Intermediate stages are possible when the percentage of

completion is displayed along the width of an activity rectangle. An arrow originating at the half width of an activity thus means that 50% completion has to be achieved for the successor activity to be able to start. Arrows should follow a clear path and pattern through the schedule diagram that is as short and as straight as possible, should have the same angle at bends, and should be drawn equidistantly when several arrows run parallel. They should connect predecessor and successor activities as directly as possible while being visible along their entire path and without interfering with each other or with activities.

Choosing a good format and arrangement for numeric and textual information, giving sufficient space to both time and logic information, providing a good legend, and keeping the symbols used for activities and arrows simple should already result in easily accessible schedule diagrams. An arrangement of activities and logic links that further improves the diagram is provided by using the arranging algorithm described in the following section. The algorithm has been derived from observing the nature and sequence of steps taken by a trained human scheduler who is asked to prepare a clear diagram with paper and pencil.

ARRANGING ALGORITHM

Activities and arrows are situated on an invisible grid. The grid allows activities to be arranged in rows and columns. Arrows usually connect the finishes and starts of subsequent activities, i.e. their right and left edges. While activities can theoretically be placed onto any free grid location it is highly desirable that activities that come later in time are found further on the right. Through this relative positioning of activities the AON diagram receives some time direction but will not become truly time-scaled along its horizontal axis.

Three arrangements are possible for the activities in such AON diagrams. In the "stepped-down" form the overall project start lies in the top left corner of the diagram and the activities are generally arranged around a diagonal until they reach the overall project finish in the bottom right corner. In the "full tree" form the overall project start and finish both lie on a horizontal line at approximately the middle of the list axis and other activities are drawn both above and below this spine to resemble a tree lying on its side. The spine is composed entirely of the activities of the critical path. In a "half tree" form the spine is situated horizontally along the top (or bottom) of the diagram and all non-critical activities are drawn below (or above) it. Note that the differences between these arrangements are entirely determined by the rows in which activities are placed, while their columns remain the same. Comparing "full tree" and "half tree" diagrams of the same schedule network shows that the former ones are less congested and less prone to having a large number of crossing arrows than the latter ones.

To create a schedule diagram based on the results from the CPM calculations it is first necessary to determine the column in the grid to which each activity should be assigned. For each activity its column number is calculated using Equation 1:

$$C_j = 1 + \text{maximum } P_{1 \text{ to } i} \qquad \text{Equation 1}$$

where C_i is the column number of activity j, P is the number of activities comprising a single piece of path reaching through the schedule from including activity 1 to including activity i, and activity i is a direct predecessor to activity j. Equation 1 is equivalent to

determining the sequence steps used in the resource leveling procedure that Callahan et al. (1992) describe without deriving the formula of Equation 1.

A crossing shall be defined as any single intersection between two arrows or between an arrow and an activity. Purchase (2002) provides the formula for the number of actual crossings created by k intersecting arrows as $\frac{1}{2} \cdot k \cdot (k - 1)$. Under this definition three arrows intersecting in the same point do not create only one crossing but three, which becomes apparent when the three arrows are slightly pulled apart towards a triangle. While any crossings are undesirable, crossings of more than two arrows in the same point should specifically be avoided because of their larger potential for confusion.

A bend along the length of an arrow shall be defined as any two consecutive segments that are at an angle with each other that is not equal to 180 degrees. Common angles found in bent logic links are 45º and 90º. It is recommended to use a layout convention that clearly determines whether an arrow originating from an activity shall run above or below activities that are located in the same row in columns to its right, when the arrow will run vertically from its originating row to its target row, and that arrows running parallel horizontally should be stacked according to their target rows.

The following arranging algorithm has been designed to be applied to "full tree" AON diagrams that this author recommends over "half tree" diagrams. The arranging algorithm is written in intuitive pseudo-code so that it can be easily understood and manually applied by schedulers who wish to create non-entangled schedule diagrams and so that it can also be quickly transferred into a computer programming language. Under research carried out by this author at The Catholic University of America it is planned to develop a software module that will perform optimizations of schedule network arrangement as an add-on to an existing scheduling software package.

Step 0: Apply schedule diagram layout convention.

Step 1: Perform the CPM calculations to obtain the early start, early finish, late start, and late finish dates, and the total float and free float to determine the activities of the critical path.

Step 2: For all activities calculate the column number as per Equation 1 based on their number of predecessors.

Step 3: Sort all activities by early start date in ascending order.

If two or more activities have the same early start date then sort these activities by their number of direct successors in ascending order;

Else if two or more activities have the same number of direct successors then sort these activities by their duration in ascending order;

Else if two or more activities have the same duration then sort these activities alphabetically by their name in ascending order.

Step 4: Beginning with the first column on the left of the diagram, draw into it all activities with this column number, filling the grid positions vertically from top to bottom in the order established in Step 3. Connect each activity with all of its predecessors following the layout conventions for arrows.

Step 5: Go to the next highest column number and repeat the procedure of Step 4 until all columns have been filled with their respective activities.

Step 6: Beginning with the second column on the left of the diagram, move each column with its activities vertically up or down until its activities on the critical path align horizontally in the same row (called the spine) as its neighbors in the columns on the left and right. [*Note that this only stretches arrows but does not affect the number of crossings*.]

Step 7: Count the total number of crossings.

Step 8: For each activity that is not part of the critical path generate its partial successor path in the order established in Step 3, ignoring activities on the critical path themselves. The partial successor path is written such that for each column all interconnected activities are listed with their name separated by commas. From the list of all possible partial successor paths, identify any non-overlapping paths that are mutually exclusive with respect to the activities that they contain. For each partial successor path in ascending order by the number of columns across which they stretch, check if moving that path to the other side of the spine reduced the total number of crossings;

If the number of crossings is reduced then keep the move and update the total number of crossings;

Else do not keep the move.

Step 9: For each activity in the order established in Step 3, count its numbers of direct predecessors and direct successors on the same side of the spine (DPSS and DSSS) and its numbers of direct predecessors and direct successors on the different side of the spine (DPDS and DSDS) to achieve a balanced distribution, not counting activities on the critical path themselves.

If the balance *B* equal to (DPDS+DSDS) - (DPSS+DSSS) ≥ 0 then check if moving that activity vertically in the same column to the other side of the spine reduces the total number of crossings;

If the number of crossings is reduced then keep the move and update the total number of crossings;

Else do not keep the move;

Else go to the next activity.

APPLICATION EXAMPLE

A sample schedule with 15 activities is used to demonstrate how to apply the algorithm. Table 1 contains an activity list with durations and successors of each activity, the results from the CPM calculations, and the column allocation as per Equation 1. Figure 1 show the schedule network diagram after the activities have been placed initially into their columns at the end of Step 7 in the algorithm. A total number of crossings of 24 is counted.

In Step 8 of the algorithm the new concept of partial successor paths is used to identify not only individual activities that could be shifted to the other side of the spine for a potential reduction in the total number of crossings, but a group of closely connected activities. For the sample schedule the partial successor paths are identified as -*A-_-D,J-_-* and *-E-G-_-H,K-M-*, where an asterisk denotes the beginning and end of the path that ties back to the critical path. The partial path including activities A, D, and J contains three activities and is non-overlapping with the partial path containing the five activities, E, G, H, K, and M. It is found that shifting the former to the other side does reduce the number of crossings.

Table 1: Schedule Activity List with CPM Calculation Results

Activity	Dur.	Succ.	ES	LS	EF	LF	TF	FF	Col.
Mob.	7	A, B, E	0	0	7	7	0	0	1
A	19	D, I, J	7	13	26	32	6	0	2
B	10	C	7	7	17	17	0	0	2
C	6	D, F, J	17	17	23	23	0	0	3
D	18	L	26	33	44	51	7	7	4
E	15	F, G	7	8	22	23	1	0	2
F	17	H, I, K	23	23	40	40	0	0	4
G	16	H, I, K	22	24	38	40	2	2	3
H	6	M	40	53	46	59	13	0	5
I	11	L	40	40	51	51	0	0	5
J	19	L	26	32	45	51	6	6	4
K	15	T/O	40	54	55	69	14	14	5
L	18	T/O	51	51	69	69	0	0	6
M	10	T/O	46	59	56	69	13	13	6
T/O	3	N/A	69	69	72	72	0	0	7

Note: Boldface activities are on the critical path.

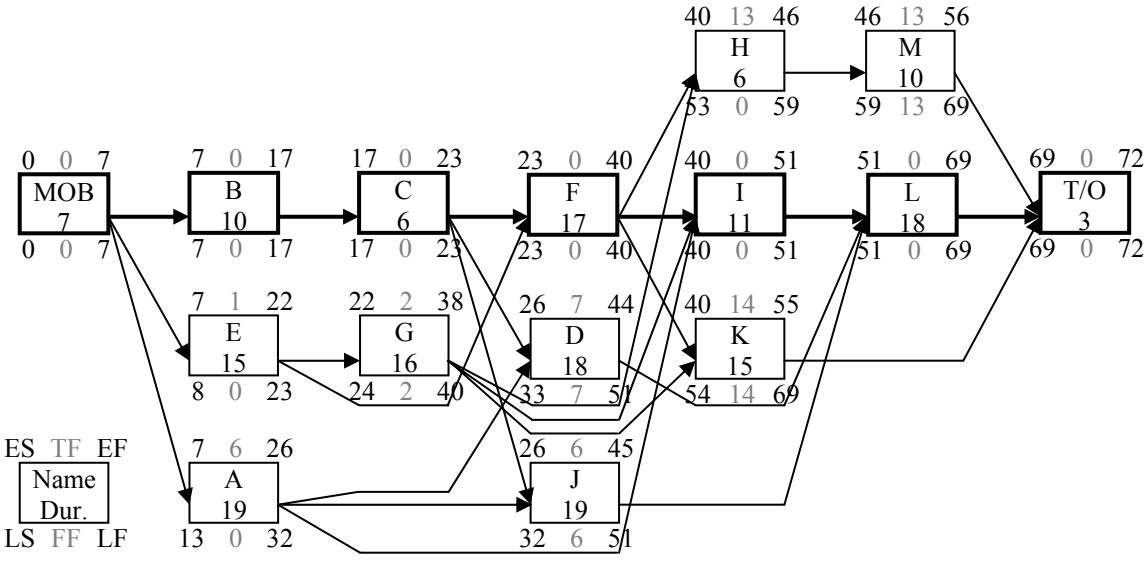

Figure 1: Initial Schedule Network Diagram

In Step 9 an additional refinement is introduced by examining the balance in the number of activities on both sides. In Figure 1 there are two and six activities, respectively, indicating that examining moving activities may be beneficial in reducing the number of crossings. Note that this indirect approach is based on the assumption that activities on average have similar numbers of predecessors and successors. Table 2 contains the measures of schedule balance for two iterations of Step 9 that marked in boldface indicate that activities H and M should be examined for moving, which indeed reduces the total number of crossings.

Table 2: Schedule Balance

	Step 9 – Move H					Step 9 – Move M				
	DPDS	DSDS	DPSS	DSSS	B	DPDS	DSDS	DPSS	DSSS	B
A	0	0	0	2	-2	0	0	0	2	-2
D	0	0	1	0	-1	0	0	1	0	-1
E	0	0	0	1	-1	0	0	0	1	-1
G	0	1	1	1	-1	0	1	1	1	-1
H	1	0	0	1	**0**	0	1	1	0	0
J	0	0	1	0	-1	0	0	1	0	-1
K	0	0	1	0	-1	0	0	1	0	-1
M	0	0	1	0	-1	1	0	0	0	**1**

Note: Do no count critical path activities for calculating the balance.

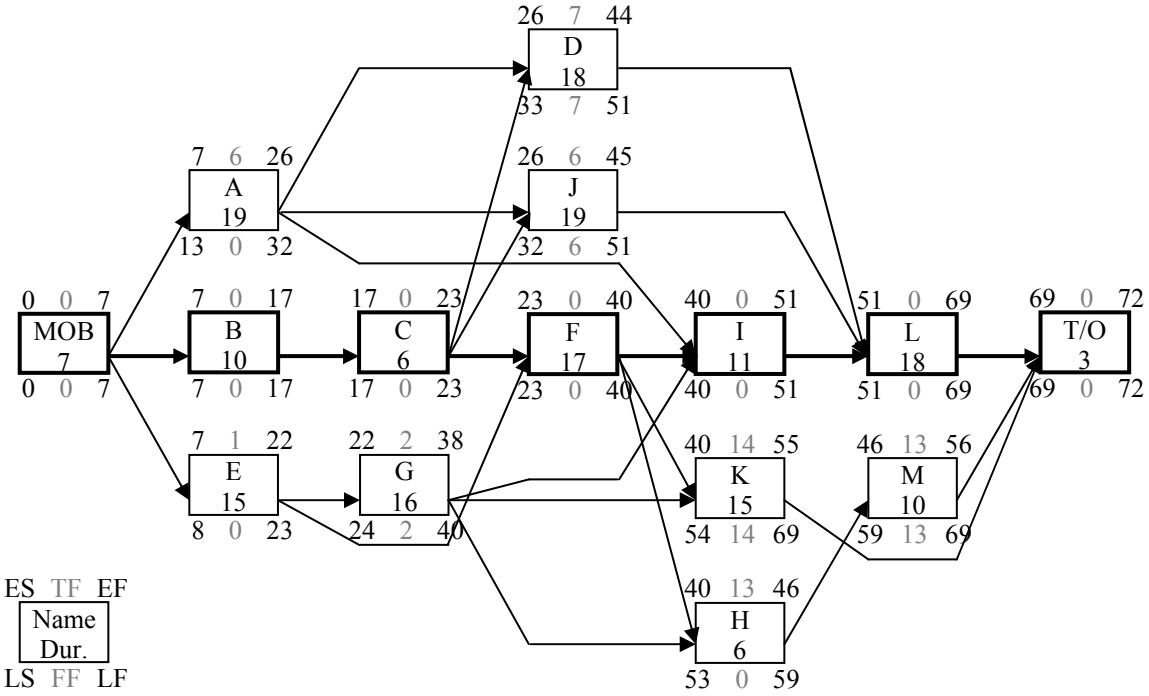

Figure 2: Improved Schedule Network Diagram

Overall the procedure outlined here reduces the total number of crossings from 24 to nine crossings that, upon closer examination of Figure 2, could be reduced to six if the link from activity E to F would be routed differently and if a permutation between the location of activity K and H would be applied as a final refinement, which is planned to be included in a subsequent more powerful version of the arranging algorithm that will be developed under the ongoing research.

FUTURE RESEARCH AND CONCLUSIONS

This working paper has presented an arranging algorithm for AON diagrams that assigns activities into specific columns in a grid in the schedule network diagram under consideration of the dependency structure and thereafter attempts to reduce the number of crossings among the logic links by strategically examining selected groups or individual activities for a potential move that would yield an improvement. To accomplish this, the new concepts of grid column allocation, partial successor path, and schedule diagram balance were introduced and used. At the same time, the paper has emphasized a number of graphics design issues that, if considered appropriately, will render a schedule diagram that is much clearer and readable than the average software-generated default printout.

In attempting to create a workable initial version of the algorithm that can still be executed and checked manually it is possible that in some cases it will yield results that deviate slightly from the overall theoretical optimum in case of a complex dependency structure or large schedule networks with their exponentially growing number of possible permutations. Further research is necessary to develop a more powerful version of this algorithm that for such complex systems will capture the nature of the schedule network, generate the optimum column and row arrangement, route logic links, and address the number of crossings, bends, and length of arrows simultaneously. Such improved algorithms would fully disentangle schedule diagrams with adequate computational effort when implemented as an add-on module with a commercial scheduling software application.

REFERENCES

Basu, A. (1990). A mechanical model for CPM scheduling calculations. Transactions of the Association for the Advancement of Cost Engineering Annual Meeting, H.5.1-H.5.6.

Callahan, M. T., Quackenbush, D. G., Rowings, J. E. (1992). *Construction Project Scheduling*. Irwin/McGraw-Hill, Boston, Massachusetts.

Lucko, G. (2005). Reviving a mechanistic view of CPM scheduling in the age of information technology. To appear in *Proceedings of the 2005 Winter Simulation Conference*, Institute of Electrical and Electronics Engineers (IEEE), Piscataway, New Jersey.

Melin, J. W., Whiteaker, B. (1981). Fencing a bar chart. *Journal of the Construction Division, Proceedings of the American Society of Civil Engineers* 107 (CO3) 497-507.

Purchase, H. C. (2002). Metrics for graph drawing aesthetics. *Journal of Visual Languages and Computing* 13 (5) 501-516.

Tufte, E. R. (1989). *Visual design of the user interface: Information resolution, interaction of design elements, color for the user interface, typography and icons, design quality.* Prepared by Edward Tufte for the IBM Design Program, IBM Corporation, Armonk, New York.

Tufte, E. R. (2001). *The visual display of quantitative information*. 2nd edition, 2nd printing 2002, Graphics Press, Cheshire, Connecticut.