May 2007

Modeling Collaboration in Civil Engineering

Mark Joseph Meko
Worcester Polytechnic Institute

Project Number: ACH-0701

Modeling Collaboration in Civil Engineering

A Major Qualifying Project Report:

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

by

_____

Mark Meko

Date:

Approved:

_____

Professor Arthur Heinricher, Advisor

Table of Contents

Abstract

Collaboration in a complex civil engineering project is modeled by two linear programs using leaks and node collapse. These linear programs are based on an event-oriented network and a task-oriented network. The critical path method is used to identify possible areas for collaboration. Node collapse is used to model collaboration in the task-oriented network. Leaks between nodes are used in the event-oriented network to model collaboration and assign a value to the collaboration.

Executive Summary

In civil engineering, project integration can improve quality and reduce costs by bringing together the knowledge management and workers through all levels of a complex project. Mathematical optimization using linear programming techniques, the critical path method, and graph theory involving network flows will be used in this report to identify opportunities for collaboration and measure the improvement achieved by collaboration.

A linear program based on a network flow diagram will be used to evaluate minimizing cost and time while maximizing flow through the tasks of a project in its dual program. A task-oriented network will be created allowing for a linear program to be modeled from the network working to minimize the total time of the project. The dual of this network, the event-oriented network, will also be created allowing for a linear program to be modeled working to maximize the flow through the activities. This gives the longest path through the network which by definition is the critical path of the project. The activities lying on the critical path have an effect on the total time of a project with change to their durations.

Areas for collaboration can then be assessed by project managers in order to cut down on time and cost of a project. These areas for collaboration are not limited to, but are primarily based on the notion of the critical path of a given system. The summation of the durations of all the tasks on the critical path is what corresponds to the total duration time of a project. These tasks must run on time without any delays for any prolonging in duration of the tasks will have a direct effect on the project as a whole.

One way of modeling collaboration in the linear program is a leak. The constraints of the network flow linear program can be thought of as written as $supply + x_{ij} = x_{jk} + demand$. A leak occurs when a task supplies resources in the form of workforce assistance or even helpful advice to another task which would accept it as demand. The constraints' right hand sides in the linear program are altered to display this forced supply and demand as negative *supply* and positive *demand*. As a result in the dual time/cost linear program is expanded from $\min t_{final} - t_{initial}$ to $\min t_{final} + leak(t_A - t_B) - t_{initial}$. The leak will be constant, thus for occurrences when task supplying resources early starting time, t_B, is later than the starting time of task accepting the resources as demand, t_A, there will be a decrease in total project duration. This can be thought of that as long as a task is waiting to go underway, there is a benefit to having that task supply additional resources to other tasks already started with the desired effect that it will shorten the task's duration with the overall effect of shortening the total project duration.

As other means of displaying collaboration in the model, the concept of node collapse is also utilized. This can occur between tasks displaying similar traits such as, but not limited to, work force, location of task, and precedence relationships. As a result of node collapse, two nodes would absorb each other, creating a new node with a duration based around an average of the two original nodes durations. The number of nodes and arcs in both linear programs are reduced. Due to this reduction in nodes and arcs, the number of constraints in the network flow and time/cost linear programs are condensed. Thus, the number of variables affecting the objective functions is cut. Node

collapse can be used best in areas where this drop in constraints and variables inevitably cause a shortening of the total project duration.

Linear programs can be greatly utilized by civil engineers for modeling collaboration. They have direct correspondence to the critical path method and can work to minimize the total duration of a project. Using leaks in the event-oriented linear program and node collapse in the task-oriented linear program, collaboration can be modeled and its effects on the project can be shown.

Introduction

In the field of Civil Engineering there are many subfields such as transportation, geotechnical, structural, environmental, water treatment, and construction. This project will focus primarily on the subfield of construction engineering and the ways that collaboration can improve overall performance in a complex construction project. Many factors affect the collaboration level. These factors include basic knowledge of the different areas of the project and the cooperation between parties involved, interaction between parties, budget, productivity, decisions, resolving issues, meetings, and quality.

Construction projects today are highly specialized consisting of various parties such as an owner, construction manager, architects, and subcontractors. Due to specialization, there is often vertical and horizontal fragmentation of a project. With many different groups involved in large, complex projects today, communication is often hindered because it is difficult to keep all individuals up-to-date on every aspect of the project. It is difficult to share knowledge in a timely fashion when there are many of groups from across the country working on the various parts of a project. Over time, lack of communication between parties can lead to harmful effects on the project. This miscommunication can lead to inefficient scheduling and production, which in effect will bring cost and project duration up. This can also lead to an unmotivated workforce displaying poor work quality.

For example, a university hires an architect to design a residence hall with a capacity of 300 beds at a specified maximum cost. The architect finds that it is not possible to reach the 300 bed capacity at the specified cost. She assumes that the cost constraint is the more important and completes a design that holds only 240 beds but

satisfies the cost constraint. When the presentation is made for the university, the university asks that the design be redone for the original 300 beds and says that the cost constraint is the less important of the two. This now has caused a delay in the project. If the client and the team had collaborated earlier in the design process, this delay could have been avoided and the design would have been prepared to the client's approval the first time.

In the world today, poor collaboration is often hard to notice until it begins to have harmful effects on a project. Many times the various parties will lead themselves to believe that the project is running smoothly and everyone involved is on the same page. This can lead to a false security since in reality some groups may have not been given the same knowledge as others.

The goal will be to establish a means to mathematically determine the effects of collaboration in a complex civil engineering projects. This system will be based on mathematical optimization using linear programming techniques, the critical path method, and graph theory involving network flow in order to create a way in which the basic factors that effect collaboration will be measured and combined to show their overall effect on the project as valued in time, cost, and effectiveness.

Background

Mathematical Programming

Mathematical programming, also known as operations research, is an essential tool in the field of management science. It studies the problem of distributing a given set of finite resources through activities performed in a particular situation, satisfying a set of constraints on the precedence relationships at all times. These precedence relationships involve those activities which must occur after or before other activities in the project. It uses a model building technique to assist management in finding the optimal solution to a given problem. In the field today there are four main types of models used. They include operational exercise, gaming, simulation, and analytical models. Each has an increasing degree of realism and cost in certain models and an increasing degree of abstraction and speed in others as shown in Figure 1. The type of model which is of most use to develop a mathematical model to measure collaboration is the analytical model. [2]

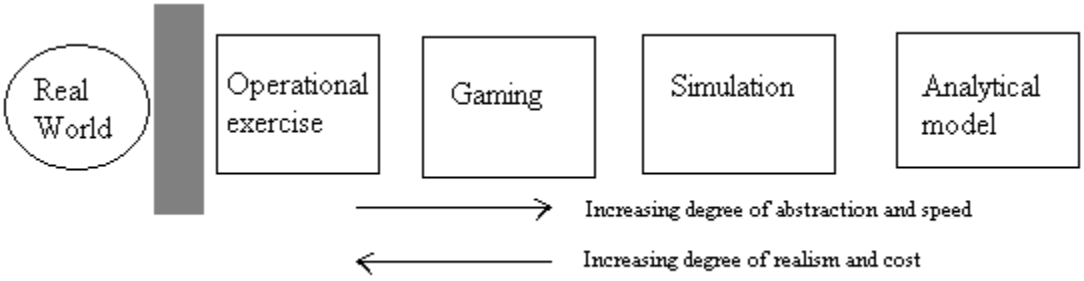


Figure 1: Types of Model Representations

Analytical modeling is completely mathematical based. A technique called linear programming is the most widely used form. Linear programming is part of the much broader field of mathematical optimization.

Linear Programming

The general form of a mathematical optimization problem is shown in Equation 1.

Equation 1: Mathematical Optimization Problem

$$\min/\max f(\underline{x}) \quad s.t. \quad \{g_i(\underline{x}) = 0, i \in \mathrm{E};\ g_j(\underline{x}) \le 0, j \in \mathrm{I}\}$$
$$x \varepsilon R^n$$

where :

$\underline{x}$: vector of variables (unknowns or decision variables)

f : objective function $R^n \rightarrow R$

g : vector of constraints g(x)

E : equality constraints

I : inequality constraints

Typically an objective function represents some part of the problem which is being maximized or minimized. This could be anything from the total cost of a project to the total duration of a project. The function depends on a number of decision variables.

For example, in the case of minimizing the total duration of a project, the starting time of each task is a decision variable. In a basic problem of this kind, the objective function is the difference between the starting time of the final task plus its duration and the starting time of the first task. In other examples, the objective function could be profit obtained from an allocation of labor and raw materials or the cost of the foods included in a nutritious diet. [2]

The decision variables must usually satisfy some set of constraints. In some examples, the constraints come from the amount of labor or raw materials available. In the problems considered in this project, the constraints come from precedence relationships between the tasks.

A standard linear programming problem is setup as shown in Equation 2.

Equation 2: Linear Program

$$\begin{aligned} \max \quad & \sum c_i x_i = \underline{c}^T \underline{x} \\ s.t. \quad & A\underline{x} \le \underline{b} \\ & x_i \ge 0 \text{ for all } i \end{aligned}$$

$\underline{c}$: $n \times 1$ vector of objective function coefficients
$\underline{x}$: $n \times 1$ vector of decision variables
A : $m \times n$ matrix of constraint coefficients
$\underline{b}$: $m \times 1$ vector of right hand side of constraint equations
m : number of constraints
n : number of decision variables

With the problem set up as shown, the set of constraints define a feasible set (Ω) consisting of all the possible solution to the problem. The solution vector $\underline{x} \in R^n$ is called feasible if it satisfies all the constraints, i.e. $\underline{x} \in \Omega$. A feasible $\underline{x} \in R^n$ is called optimal if it attains the desired maximum or minimum.

There are two special cases where linear programming may not have an optimal solution. In one case, the feasible set is empty so there are no feasible solutions. This type of problem is called *infeasible*. An example is shown in Equation 3.

Equation 3: Infeasible Solution Case 1

$$\begin{aligned} \max \quad & 7x_1 + 9x_2 \\ s.t. \quad & x_1 + x_2 \le 5 \qquad (1) \\ & -5x_1 - 5x_2 \le -30 \qquad (2) \\ & x_1, x_2 \ge 0 \end{aligned}$$

The two constraint equations (1) and (2) contradict each other and therefore causing $\Omega = \emptyset$.

A linear program can also fail to have a solution because the feasible set is unbounded and the objective function can be made arbitrarily large (positive or negative). An example of this is shown in Equation 4.

Equation 4: Infeasible Solution Case 2

$$\begin{aligned} \max \quad & x_1 - 4x_2 \\ s.t. \quad & -2x_1 + x_2 \le -1 \qquad (1) \\ & -x_1 - 2x_2 \le -2 \qquad (2) \\ & x_1, x_2 \ge 0 \end{aligned}$$

Notice that if $x_2 = 0$, then $-2x_1 \le -1$ and $-x_1 \le -2$ is the only constraint on x_1 allowing it to be arbitrarily large. Since the objective function is to maximize x_1, the problem has no optimal solution. [6]

To find the optimal solution to a linear programming problem, several algorithms can be used. The simplex method, developed by George Dantzig in 1947, is one of many methods used to solve linear programming problems. For this project, the linprog function in Matlab® was used in the trial runs of our linear programs. This function has the choice of using many different methods depending on the type of problem it is trying to solve.

Duality in Linear Programming

For any given linear programming problem, there is paired problem called its *dual*. The dual of the problem can be obtained by rearranging the coefficients, variables, and constraints of a linear programming problem and creating a new problem. The original problem is referred to as the *primal*. Both the primal and dual problems give much insight into the other's solutions, including its feasibility and optimality.

There are several ways to generate the dual for a linear programming problem, each of which depends on the structure of the primal. For example, take a linear programming problem in canonical or standard form as in Equation 5:

Equation 5: Primal Linear Program in Canonical Form

$$\begin{aligned} &\text{minimize} \sum_i b_i y_i \\ s.t \quad &\sum_i y_i a_{ij} \geq c_j \quad \text{for } j = 1,2,\ldots,n \\ &y_i \geq 0 \quad \text{for } i = 1,2,\ldots,m \end{aligned}$$

The dual linear program associated with this is shown in Equation 6:

Equation 6: Dual Linear Program in Canonical Form

$$\begin{aligned} &\text{maximize} \quad \sum c_j x_j \\ s.t. \quad &\sum_j a_{ij} x_j \leq b_i \quad \text{for i} = 1,2,\ldots,\text{m} \\ &x_j \geq 0 \quad \text{for j} = 1,2,\ldots,\text{n} \end{aligned}$$

For this case one can see that to move from the primal to the dual of any linear program, exchange the constraints with the coefficients of the objective, and the objective coefficients to form from the constraints. In this way one can analyze not just a maximization or minimization of a given set of variables for over a set of constraints, but to analyze an almost inverse of that problem with taking the constraints inequalities and seeing how if they are introduced to the objective, how they hold over constraints formed from the primal objective. [6]

Not every linear program comes in canonical form. If the constraints consist of both equalities and inequalities, they can be changed to canonical form using elementary mathematics. A set of simple rules convert the primal to the dual and vice versa. The rules are described in Table 1:

Primal	Dual
Constraint *i* is ≤	$y_i \geq 0$
Constraint *i* is =	y_i is unrestricted
Constraint *i* is ≥	$y_i \leq 0$
$x_j \geq 0$	Constraint *j* is ≥
x_j is unrestricted	Constraint *j* is =
$x_j \leq 0$	Constraint *j* is ≤

Table 1: Primal Dual Correspondence

Finding a feasible solution for either the primal or dual gives a bound on the optimal objective function value for the other. This is proven in the weak duality theorem shown here [6]:

Let $\underline{x}$ be a feasible solution for the primal linear programming problem in the form:

$$\begin{aligned} \min \quad & z = \underline{c}^T \underline{x} \\ s.t. \quad & A\underline{x} = \underline{b} \\ & \underline{x} \geq 0 \end{aligned}$$

And let $\underline{y}$ be a feasible solution for the dual linear programming problem:

$$\begin{aligned} \max \quad & w = \underline{b}^T \underline{y} \\ s.t. \quad & A^T \underline{y} \leq \underline{c} \\ & \underline{y} \text{ unrestricted} \end{aligned}$$

Then $z = \underline{c}^T \underline{x} \geq \underline{b}^T \underline{y} = w$.

Proof:
Since the constraints for the dual are $A^T \underline{y} \leq \underline{c}$, taking the transpose of both sides would give:

$$\underline{y}^T A \leq \underline{c}^T .$$

Since $x \geq 0$, $z = \underline{c}^T \underline{x} \geq (\underline{y}^T A)\underline{x}$. Therefore, $z = \underline{c}^T \underline{x} \geq (\underline{y}^T A)\underline{x} = \underline{y}^T (A\underline{x})$. Since it is given by the primal problem that A$\underline{x}$=$\underline{b}$, by substitution

$z = \underline{c}^T \underline{x} \geq (\underline{y}^T A)\underline{x} = \underline{y}^T (A\underline{x}) = \underline{y}^T \underline{b}$

Finally, $\underline{b}$ is an $n \times 1$ vector and $\underline{y}$ is an $n \times 1$ vector, the product of $\underline{y}^T \underline{b}$ would be $1xn \cdot n \times 1 = 1x1$ which is a scalar and the product $\underline{b}^T \underline{y}$ would be $1 \times n \cdot n \times 1 = 1 \times 1$ which is also a scalar and would be equal to $\underline{y}^T \underline{b}$. Thus, $z = \underline{c}^T \underline{x} \geq \underline{b}^T \underline{y} = w$.

Critical Path Method

The purpose of a project schedule is to order the tasks based on their precedence relationships. The precedence relationships play a major role in civil engineering projects. They describe what tasks must be complete in order for another task to begin. In this way, tasks can be scheduled so that they do not occur before they are supposed to and the project can run as smoothly as possible. Before the Critical Path Method was developed, bar charts were the most economical, simple, and common form of project scheduling. For small projects they are simple and easy to follow, but they do not show the relationships between the activities. Without this it is almost impossible to analyze the importance of each event to the project as a whole. For example, which tasks could possibly delay the finish time of the project if their durations were altered? [7]

In the early 1940's the use of preplanned and written schedules using milestones and phases had become common practice on capital projects. As the need for a more accurate and professional way to monitor projects, the Critical Path Method (CPM) was developed. In the late 1950's E.I. DuPont deNemour Company wanted to seek out a system by which they could monitor and manage how their in-plant processes changed throughout construction. DuPont's operation team of mathematicians and engineers, along with the Rand Corporation, developed what is known today as the Critical Path Method. Around the same time, the United States Navy hired Booz-Allen & Hamilton,

as a consulting firm, to develop a system to monitor the progress of the Polaris Missile Project, in which they developed Program Evaluation Review Technique, also known as PERT. PERT is based on the same concepts as CPM but makes a best guess estimate on each activity's duration using tools such as past experience. PERT does not offer the same scheduling benefits as CPM because its time estimates are generalized and hence do not provide exact time durations. [7]

CPM was first used in the aerospace and petrochemical industries, and it was not until the 1960's that it was first successfully applied to a project in the construction industry. CPM is now so widely used that on most public works, industrial, commercial, and multi-residential construction it is a contractual requirement that it be used for project scheduling. The main reasons that CPM is widely used is that it allows management to monitor progress in the project, identify possible production problems, and decide what changes they need to make at a certain time in order to keep the project on track. It is because CPM allows for in-depth analysis of a project's components and provides a variety of solutions that it is the most accurate and efficient system of network scheduling today. [7]

The Critical Path Method can be divided into three distinct phases. During each phase data is organized and then analyzed to show relationships between the different steps in a project. The key step in this process is to identity predecessor sets for each activity. Each step's duration, prior activity completion requirements, cost data, and other factors are taken into account throughout the three phases in order to obtain the greatest understanding of a project as a whole.

The first phase of the CPM is the most basic. During this phase a "network model" is created to show the types of activities needed to complete a given project. Many factors are taken into consideration when organizing this model. The following shows how the activities are created. [1]

First the types of work and labor classifications are considered. For example, electrical work is separated from plumbing work. Next the structural elements are separated. For example, column and frame construction is separated from the activities needed to construct walls or floors. Activities are then broken down further by the parties responsible for their completion. An example of this would be the separation of activities performed by the general contractor from those performed by the subcontractors. Further division can be made by organizing the activities by the location where the work needs to be performed. For example, construction done for the walls and frame must be completed prior to work done on the roof. Finally, the activities are categorized by cost consideration. Certain areas may need to be completed during certain times to keep cost down. An example of this would be weather considerations. If a task that requires digging into the earth can be performed any time during the spring months, it would be more efficient to wait until the end of the spring to start the task. This is due to the fact that the ground would be softer and therefore much easier to dig into, bringing the overall time and cost down.

With all the activities needed to complete a project partitioned as above, a model can now be created to show the flow of the activities from the start of the project to its completion. Activities are arranged by the order in which they need to be performed. That is, certain activities naturally must occur prior to the start of another activity, while other

activities may have no prior work to be done and can be performed at any time throughout the construction process. [1]

The final model created during this phase will give a visual representation of the flow of activities to complete the given project. The network model is constructed of the activities represented in squares or circles with arrows moving from one activity to the next ensuring all the factors mentioned above are achieved when having one activity precede another. [1]

The second phase of the CPM focuses on scheduling the activities. During this phase activity durations, early start and finish times, and late start and finish times are calculated. Using durations and order of the activities, a schedule can be proposed to show when each of the activities need to be started and finished. Usually this information is represented in bar chart form, also known as a Gantt chart. Activities that have prerequisites are placed one after another in chronological order as they need to be performed. Using this form or visualization, activities which are critical can be easily identified to show priority. Critical activities include those which would postpone the completion of the project as a whole if it is not started at a specified time and completed in the given duration. In contrast to this, non-critical activities include those which have floating time in which they need to be started or completed. Usually these activities do not need to be completed to proceed with other activities or they occur simultaneously with another activity with a much longer duration period. The activity then can be delayed as long as it is started enough in advance that it will finish with the completion of the activity with longer duration time. All this information is important for control purposes. It helps contractors create an optimum shift schedule, which is used to

determine when each group working on a project will begin and end their task, for all parties involved in the construction of the project. [1]

The third phase of the CPM brings the information gathered during the second phase of the CPM and shows its significance on a cost reduction point of view. With the early start and finish and late start and finish times known, for all the activities which are non-critical, beginning and ending times must be established. The method used for establishing these times is centered on a time-cost relationship. Start and finish times are created to minimize the overall cost of a project. The minimal overall cost of the project is calculated by having each of the activities analyzed and a minimal cost for the activity is calculated. All of the activities' minimal costs are then summed resulting in the least direct cost calculation. During this phase normal duration time of activities are analyzed to see if any "speeding up" can be done to reduce the overall time of the project. However this "speeding up" can only be done so that it does not have a negative effect on the cost side of the project. At a certain point this process can lead to overtime work, multiple-shift scheduling, larger but less efficient crews, materials being purchased at higher prices to accommodate for rush deliveries, and overuse and wear of equipment and personnel. This point in which reducing normal duration times of activities becomes harmful to the overall flow and cost of the project is known as the "all-crash" point. This point is usually strived to not be met, with the unusual exception in which the "all-crash" point is necessary due to forces beyond the contractor's control, e.g. the buyer of the project wanting it completed now and not later or they will look to other contractors to perform the job. In the end the third phase of the CPM is focused on reducing the overall cost of the project. [1]

An example of a critical path is shown in Figure 2. Tasks on the critical path will be those whose late finish is equal to its late start and early finish is equal to its early start. Thus these tasks have no slack time and must be started and completed on time or it will affect the total outcome of the project's duration.

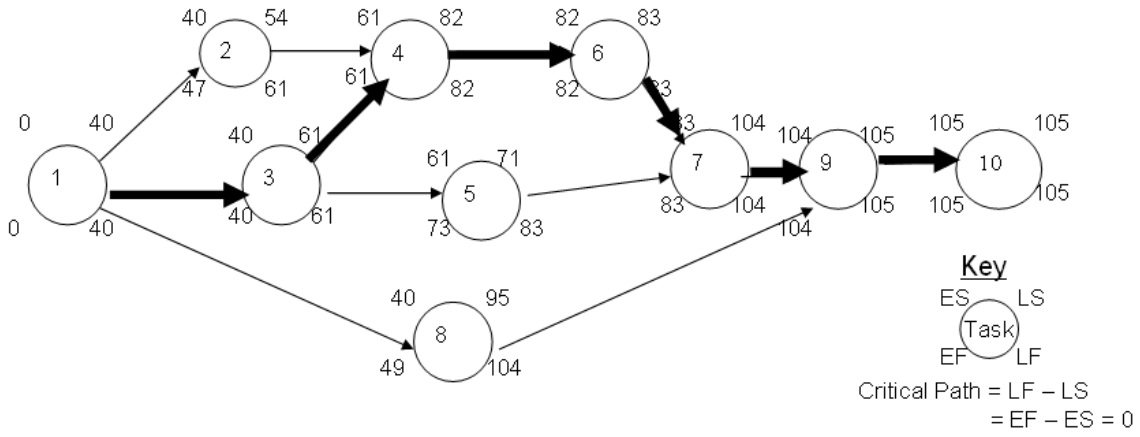


Figure 2: Critical Path Example

To obtain the critical path, start with the first activity and place its early start time in the top left corner. Then the early finish time is computed by adding the duration of the first task to its early start time and placing this value in the top right corner. This early finish time then becomes the early start time of the next activities. Their early finish time is then computed the same way and this continues through the network.

Working your way back the late finish time of the last task is set equal to its early finish time. The duration of the task is subtracted from this and this value becomes the late start time of that task. This late start time then becomes the late finish time of all preceding tasks. This method is following back through the network until all late and early start and finish times are calculated. The critical path through the tasks then becomes those whose early start and early finish times are equal as well as its late start and late finish times.

The critical path will be utilized in making the choice for task collaboration. Those tasks lying on the critical path should be looked at first for possible means of collaboration since their starting times have a direct affect on the total project duration. The tasks on the critical path should have a clear understanding of what is being supplied to them, what is being demanded by them, and the status of all other tasks in the network. This way a complete knowledge of each task will be known and possible means of collaboration can be established by the parties involved.

Network Flow

Graphs have great importance in the analysis of activities and their precedence relationships in a project. A graph is a collection of vertices and edges that connect pairs of vertices. In graph theory a network flow diagram is a graph where flow is distributed through the directed edges of the graph, and each edge of the graph has a certain weight which cannot be exceeded. The first vertex of the network is called the source node and the final vertex of the network is called the sink node. One unit is sent into the network through the source node and one unit is sent out of the network through the sink node as shown in Figure 3. Each node represents a precedence relationship in the project and each edge represents an activity.

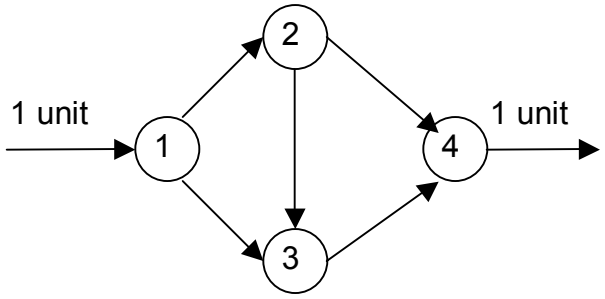


Figure 3: Basic Network Flow Diagram

There is a *balance equation* that must hold at each node: the total amount of flow into a node must equal the total amount of flow out of a node. This flow can be sent through any path in the network or even split between multiple paths. In the example in Figure 3, there is one unit of supply coming into node 1, so the total flow out, some into node 2 and some into node 3, must equal 1 unit. Similarly, the total flow into node 2 (from node 1) must equal the total flow out of node 2 (into nodes 3 and 4). An example of a feasible solution for the flow problem in Figure 3 is shown in Figure 4 below.

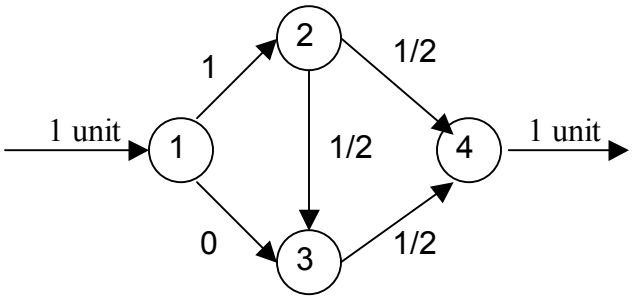


Figure 4: Flow Distribution Example

A network flow diagram can be used for traffic studies, pipes, electrical work, or as in this project for project scheduling. Below is a proof of how sending 1 unit of flow along the critical path will result in a payoff at least equal to any payoff calculated when splitting the flow through multiple paths. This result could then calculate the longest path in the system, and thus determine the critical path.

Suppose a network diagram exists with a structure as shown in Figure 5:

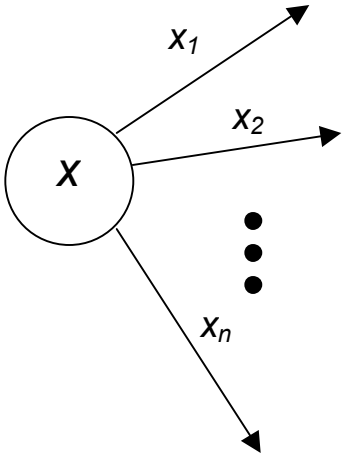


Figure 5: Single Node with Arcs

Let the nodes of the network represent certain activities along the path of the network. Each activity has a certain payoff, τ_x with x corresponding to the activity. Suppose now that 1 unit of flow is sent through the network. When this unit reaches a node with multiple arcs leaving the node it may move 1 whole unit through one of the arrows, or may be split into multiple arcs. If it is split into multiple arcs certain percentages, x_i where *i* is a subscript of the arc taken, are sent through each arc where $\sum x_i = 1$. The final payment of each activity is calculated by summing the products of the percentage of unit of flow sent through each arc by the payoff for that activity $\sum x_i * \tau_x$ where *i* are the subscripts of arrows taken and x is the leaving node.

Let 1 unit of flow enter node x . Node x has n arcs pointing to other nodes. Node x has one option of sending 1 unit of flow through 1 of the arcs. This final payment would be $1 * \tau_x$. Else the unit may send certain percentages through multiple arcs. In this case the total payment would be $\sum x_i * \tau_x$. Using summation properties,

(1) $\sum x_i * \tau_x = \tau_x * \sum x_i$. Since $\sum x_i = 1$, (1) becomes $\tau_x * 1$. Thus, the total payment for either choice is equal.

Now let's say there exist two nodes x_i and x_j with multiple paths connecting them. Let $P(y_1)$ denote the payoff of the path resulting in at least the greatest sum of payoffs across the nodes with flow α. Let $P(y_2)$ denote a different path which can be chosen with flow $1-\alpha$. Thus, $P(y_1) \geq P(y_2)$. Then the total payment between these two nodes would be $P_T = \alpha * P(y_1) + (1-\alpha) * P(y_2)$. There are 3 possibilities then for flow between these two nodes. If $\alpha = 1$, then $P_T = P(y_1)$. If $\alpha = 0$, then $P_T = P(y_2)$. If $0 < \alpha < 1$, then $P_T = \alpha * P(y_1) + (1-\alpha) * P(y_2)$. Since $P(y_1) \geq P(y_2)$, $\alpha * P(y_1) + (1-\alpha) * P(y_2) \leq P(y_1)$ and $P(y_2) \leq P(y_1)$. Thus, using $\alpha = 1$ and sending 1 whole unit through a single path along the greatest payoff path will result in the largest payoff.

Linear Programming for Task-Oriented and Event-Oriented Networks

CPM Linear Programming

Moder and Phillips [4] were perhaps the first to determine the activities on a critical path and formulate this as a linear programming problem. The starting point for the formulation is the set of tasks and the precedence relations between the tasks. An example is presented in Table 2.

Table 2: Tasks and Predecessors

Activity	Predecessors	Duration	Earliest Start Time
0	ϕ	–	–
1	0	τ_1	t_1
2	1	τ_2	t_2
3	1	τ_3	t_2
4	2,3	τ_4	t_3
5	3	τ_5	t_4
6	4	τ_6	t_5
7	5,6	τ_7	t_6
8	1	τ_8	t_2
9	7,8	τ_9	t_7
10	9	τ_{10}	t_8

The relations listed in Table 2 define the task-oriented graph in Figure 6 below. Each node refers to a task and there is an arc from node *i* to node *j* if task *i* must be completed before task *j* can begin.

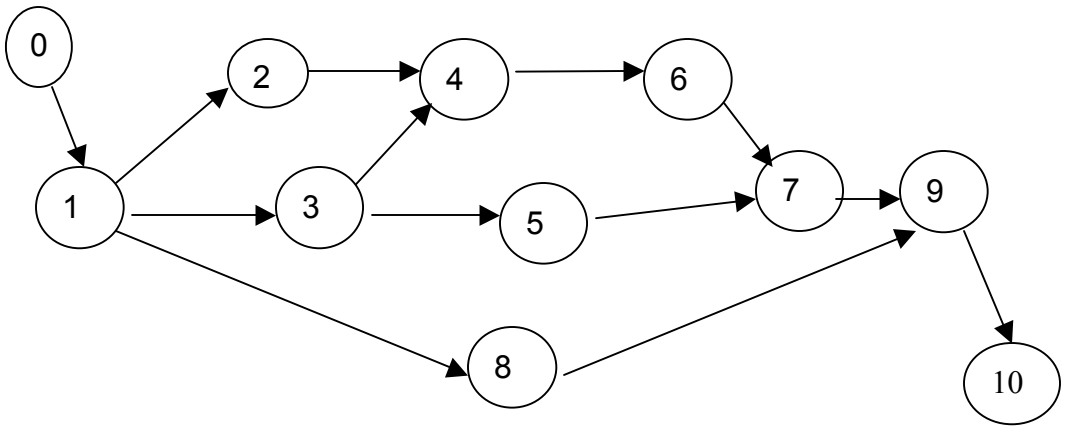


Figure 6: Task-oriented network

The decision variables for the critical path problem are the starting times for the tasks grouped by predecessor sets (the final column in Table 2). For example, tasks 2, 3 and 8 have the same starting time t_2 because they have the same precedence set, namely task 1. Since tasks 2 cannot begin until task 1 is complete, the following constraint must be satisfied:

$$t_2 \geq t_1 + \tau_1$$

where τ_1 is the duration for task 1. A similar constraint holds for tasks 2 and 8 and then for every other precedence set in the table.

The objective is then to find the minimum duration. In this example, the duration of the entire project is $\tau_{10} + t_8 - t_1$. Because τ_{10} is simply the duration of the last task and is not affected by the starting times, the linear program seeks to minimize $z = t_8 - t_1$.

Equation 7: LP for the Task-oriented Network

$$\text{Minimize } z = t_8 - t_1$$

subject to

$$\begin{aligned} t_1 &\geq 0 \\ t_2 - t_1 &\geq \tau_1 \\ t_2 - t_1 &\geq \tau_3 \\ t_3 - t_2 &\geq \tau_4 \\ &\vdots \\ t_8 - t_7 &\geq \tau_{10} \end{aligned}$$

Notice that when a task has two or more predecessors, as in the second and third constraint above, only the longer duration is actually needed n the final formulation.

Assigning durations for the tasks, the final linear program for the task-oriented example is shown in Equation 8:

Equation 8: Task-oriented LP

$$\begin{aligned} \min z &= t_8 - t_1 \\ t_2 - t_1 &\geq 40 \\ t_3 - t_2 &\geq 21 \\ t_4 - t_2 &\geq 21 \\ t_5 - t_3 &\geq 21 \\ t_6 - t_4 &\geq 10 \\ t_6 - t_5 &\geq 1 \\ t_7 - t_6 &\geq 21 \\ t_7 - t_2 &\geq 55 \\ t_8 - t_7 &\geq 1 \\ t_i &\text{ unrestricted} \end{aligned}$$

When this LP is solved, the minimum value gives the duration for the entire project (once you add the duration of the final task) and the start times that appear in the

optimal solution will identify the activities that constrain the duration of the project. This determines the critical path in the task-oriented network shown in Figure 6. (See also the example in Figure 2.) An activity that is not "critical" is still necessary for the completion of the project but its duration could change slightly without delaying the entire project. [2]

Duality in CPM Linear Programming

The dual problem for the task-oriented network LP in Equation 7 will be the linear program for a new type of network optimization problem. The dual problem is obtained "mechanically" using the rules introduced in Table 1 above. The new problem can be interpreted as a network flow problem. Its solution identifies the critical path and the minimum duration for the project, but it also gives new information on how collaboration can reduce the time to complete a project.

To build the dual problem, start with matrix associated with these constraints in Equation 8. This matrix is shown in Figure 7, where each row comes from one constraint and each column is associated with one starting time. Notice also that each row contains a single +1 and single -1.

t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
-1	1	0	0	0	0	0	0
0	-1	1	0	0	0	0	0
0	-1	0	1	0	0	0	0
0	0	-1	0	1	0	0	0
0	0	0	-1	0	1	0	0
0	0	0	0	-1	1	0	0
0	0	0	0	0	-1	1	0
0	-1	0	0	0	0	1	0
0	0	0	0	0	0	-1	1

Figure 7: Matrix of Task-Oriented Constraints

The transpose for this matrix defines the constraints for the dual problem. More important, this new matrix is the adjacency matrix for a new network. The columns are labeled with the new (dual) variables for a network flow problem. For example, x_{46} is the variable associated with the fifth column where a -1 in the 4th row and a +1 in the 6th row indicate that the flow is from node 4 to node 6.

x_{12}	x_{23}	x_{24}	x_{35}	x_{46}	x_{56}	x_{67}	x_{27}	x_{78}
-1	0	0	0	0	0	0	0	0
1	-1	-1	0	0	0	0	-1	0
0	1	0	-1	0	0	0	0	0
0	0	1	0	-1	0	0	0	0
0	0	0	1	0	-1	0	0	0
0	0	0	0	1	1	-1	1	-1
0	0	0	0	0	0	1	0	1

Figure 8: Matrix of Event-Oriented Constraints

The event-oriented network defined by this new matrix is shown in Figure 9:

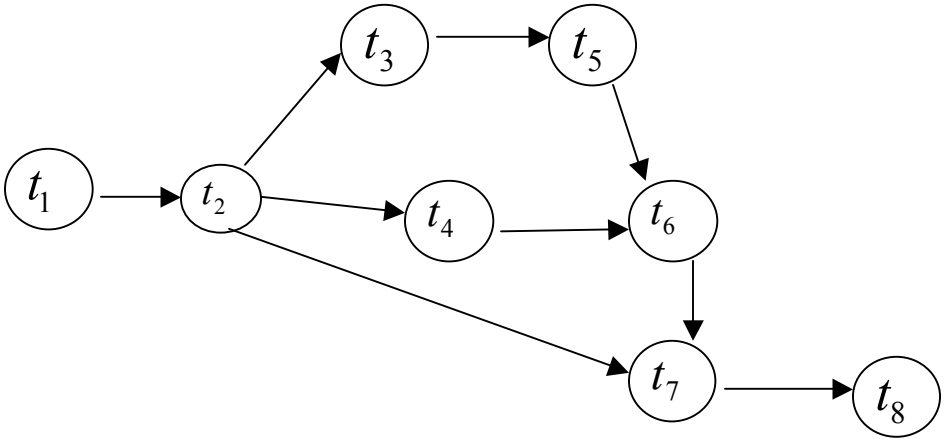


Figure 9: Event-oriented network

The linear program associated with this new network is a network flow problem. Assign one unit of supply at the initial node t_1 and one unit of demand at the final node

t_8. There is a constraint at each node which comes from a *balance equation: the flow into a node must equal the flow out of a node.* The decision variables are the amount of flow to assign to each arc in the network. At node t_1, the flow in is 1 and the flow out is x_{12} so the balance equation is just $x_{12} = 1$, or rewriting this in a standard form (as flow in minus flow out equals demand minus supply at the node): $0 - x_{12} = -1$. There is a similar constraint at node t_2: $x_{12} - (x_{23} + x_{24} + x_{27}) = 0$ because there is no net supply or demand at this node.

The objective function for this dual problem comes from the right hand side of the constraints for the task-oriented problem. The coefficients in the objective are just the durations for the tasks associated with each arc in the network. The final linear program for the even-oriented network is given in Equation 9:

Equation 9: Event-Oriented Linear Program

$$\text{Maximize } w = 40x_{12} + 21x_{23} + 21x_{24} + 55x_{27} + 21x_{35} + 10x_{46} + 1x_{56} + 21x_{67} + 1x_{78}$$

subject to

$$\begin{aligned} -x_{12} &= -1 \\ x_{12} - x_{23} - x_{24} - x_{27} &= 0 \\ x_{23} - x_{35} &= 0 \\ x_{24} - x_{46} &= 0 \\ x_{35} - x_{56} &= 0 \\ x_{46} + x_{56} - x_{67} &= 0 \\ x_{27} + x_{67} - x_{78} &= 0 \\ x_{78} &= 1 \\ x_{ij} &\geq 0 \end{aligned}$$

The last constraint needs to be included because no event can accept or distribute a negative supply or demand. [8]

In this use of linear programming, time and flow have a close relationship. In the problem it shows how to maximize the path along the network. In doing so, the problem

calculates the maximum amount of length to provide for the minimal amount of time. The resulting value of the objective function, using the solution found to maximize the length, would be the time required to move completely through the network, i.e. minimum time to complete the project in its entirety. [8]

Duality in linear programming becomes a useful tool in critical path method linear programming problems. Using the linear program above, the optimal solution would give the activities which occur on the critical path of the project. This is fine to solve for analysis about the flow of the network, but what happens if you want to find information about the other parameters such as the early start times of the activities?

This is where duality comes into play. Duality allows for a problem to be analyzed from a different perspective. For the dual problem, a task-oriented network is created where the arcs represent the precedence relationships and the nodes represent the tasks. Figure 6 shows the task-oriented network formed from the event-oriented network mentioned previously.

In the primal problem given previously, the objective function is written to find the maximum length through the network using the time durations for each activity as parameters. Finding the dual of this problem would result in a linear program representing the task-oriented network. Each task is assigned a variable t_i, representing its early start time based off of similar predecessors where the subscript *i* corresponds to the precedence relationship. Tasks having the same precedence relationship will have the same early starting time. The linear program is set up by having the objective function minimize the difference between the early starting time of the last task, which is the completion time of the project, from the early starting time of the first task.

The constraints are made based on the fact that the early start time of any task must be greater than or equal to the early start time of its predecessor plus the duration of that predecessor, $t_{task} \geq t_{predecessor} + \tau_{predecessor}$. So the first constraint starts with task two since task one's predecessor is task zero, which has no early start time. So task two's early start time is denoted t_2. Its predecessor, task one, has an early start time of t_1 and duration 40. So the constraint is set up as $t_2 \geq t_1 + 40$ or moving the variables to the left hand side, $t_2 - t_1 \geq 40$. The rest of the constraints are set up in a similar way. For the occurrence when a task have two predecessors with the same early start time as with task 4, the longer duration is used since it creates a tighter constraint.

Models for Collaboration

The ultimate goal of this project is to develop a mathematical model upon which project integration, mainly through the existence of collaboration, can be analyzed and quantified. Critical Path Management and the associated linear programming problems (primal and dual) compute the minimum time to completion for a complex project. The linear programs also provide a great deal of additional information. Perhaps the most important information is the set of tasks in the critical path, the tasks that actually determine the duration of the project. An effort to control the time to completion must focus on these tasks first. On the other hand, effort to speed up any task which is not on the critical path will be wasted because it will have no effect on the final duration of the project.

A mathematical analysis of collaboration must focus on how different tasks in the project interact, or can be made to interact. Looking at the project as a whole makes developing the model too complicated. In civil engineering projects (primarily construction) there are stages that the project goes throughout the duration of the project. Within each stage of a project there are certain activities that occur and with each activity there are certain cost, scheduling, and management constraints that are considered. In developing the model for this problem, these different factors will be looked at by each stage in a project and then each stage will be broken down according to its activities and then each factor analyzed.

With almost any action involving reorganizing activities in a project a cost is associated with that action. It has already been shown how manpower and overtime can help reduce a project's total duration but increases the cost of the project. Most often

activities get behind schedule due to miscommunication, improper forecasting of activity durations, and other factors. Miscommunication comes from the management of the project, subcontractors, owner, designer, etc. not communicating efficiently. If the management of the project can cooperate effectively and collaborate with one another it is possible that a project can stay on schedule better, have fewer delays, and less cost. Since time and cost are the two major concerns of a project looking at how collaboration affects these two factors is essential. Ideally the payoff associated with collaboration method will be less than that associated with the time-cost tradeoff, so a linear program modeling collaboration should be analyzed prior to that of the time-cost tradeoff method. Therefore by looking at activities within the project and how the cooperation of management affects a certain activity and the project as a whole a linear program can be developed which models the effectiveness of the management in relation to collaboration with one another. This can be done through the use of leaks in the event-oriented network and node collapsing in the task-oriented network.

As previous said the more integrated a project is the less expensive, better quality, and more collaborative it is. For every task in a project there is a certain amount of collaboration that is associated with that task. If the collaboration throughout an activity is efficient it is less likely to lead to delays, higher cost constraints, or rescheduling than if there were inefficient collaboration.

Leaks

The dual problem to the original critical path management problem can be visualized as a network flow problem in which 1 unit of "project" must be "shipped" from a source node (the start of the project) to a final node (the end of the project). Each

edge in the network (the edges are the directed arrows connecting two nodes) is associated with one of the tasks in the original description of the project. There is a cost associated with each path in the network and this cost is exactly the time associated with the task. The optimal solution to the network flow problem includes the tasks on the critical path and the total cost along the entire path is the time to complete the project.

In the standard network formulation there is only one node with supply (the start node) and one node with demand (the final node). Adding artificial supply and demand at intermediate nodes changes the network flow problem. If the supply and demand is created at appropriate nodes, there is an associated reduction in the total time for the entire project. In this way, these artificial leaks between nodes can be viewed as identifying opportunities for collaboration in a complex project.

There is a leak of flow that is associated with each node in the event-oriented network that expresses how efficient or inefficient the collaboration of management was for that milestone. This occurs when a portion of demand out of a node is reallocated to another node as supply. The leak that is associated with that node relates to the efficiency of collaboration for the event preceding that milestone.

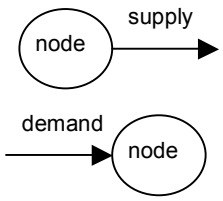


Figure 10: Supply and Demand

The leak of flow in the network is based on the efficiency of collaboration amongst management. How the leak of flow is quantified may possibly be based on a variation of methods such as Expected Monetary Value, Probability of Success, Expected

Regret, Scientific Judgments (i.e. injury to project, threat to project, etc.) and weights, and any other methods that may measure how efficient management is. The value of these leaks would be determined by the civil engineer, denoted in the linear program as l_{ij}, where *ij* = task with precedence relation *i* affecting task with precedence relation *j*.

When leaks are introduced, the right hand side of the flow linear program is perturbed as shown in Equation 10:

Equation 10: Event-Oriented Linear Program Constraints with Leaks

$$\begin{aligned} -x_{12} &= -1 \\ &\vdots \\ x_{ij} - x_{jk} &= -l_{xy} \\ x_{rs} - x_{st} &= l_{xy} \\ &\vdots \\ x_{45} &= 1 \\ x_{ij} &\geq 0 \end{aligned}$$

In the dual for this new problem, the leaks appear as a change in the objective function. This is now the linear program for a task-oriented network similar to the original critical path management problem, with objective function shown in equation 11:

Equation 11: Task-Oriented Linear Program Objective Function with Leaks

$$\min z = t_8 + l_{xy}(t_A - t_B) - t_1$$

As a result of the network flow alteration, the objective function of the task-oriented linear program is altered by adding/subtracting percentage of the early start time of affected tasks to the total project duration. So leaks will have a positive effect for those tasks that can decrease the demand of other tasks which have a smaller early starting time. This can be thought of as having a task waiting to begin supply input and/or assistance to other tasks that are underway. An example of the effects of leaks on the

linear program is shown in Figure 11. Here 0.1units of flow is forced from node 3 into node 5 causing a reduction of total duration of the project from 105 days to 102.9 days. The code for running this analysis is shown in Appendix B.

```
>> primallinprog(0,0)
Optimization terminated.
x =
    1.0000
    1.0000
    0.0000
    0.0000
    1.0000
    0.0000
    1.0000
    1.0000
    1.0000
fval = 105.0000
```

```
>> primallinprog(-.10,.10)
Optimization terminated.
x =
    1.0000
    1.0000
    0.0000
    0.0000
    0.9000
    0.0000
    1.0000
    1.0000
    1.0000
fval = 102.9000
```

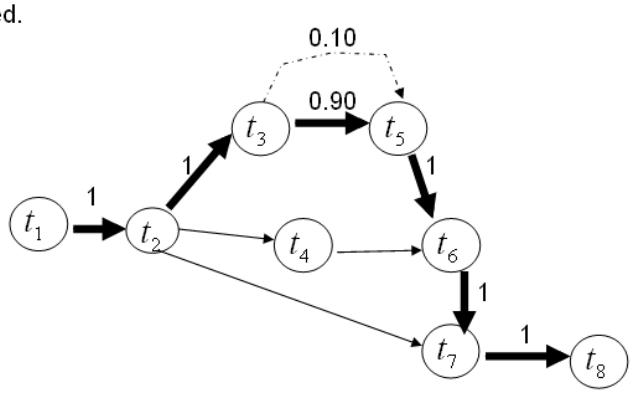


Figure 11: Leak Example

Node Collapse

Another means of modeling collaboration is through node collapse in the original task-oriented network. In this formulation, two separate tasks are combined into a single task for the purposes of critical path analysis. In practice, this would probably occur for tasks with similar traits (for example work force, location) or identical precedence sets. The ability to collapse two nodes into one another is a result of the highest amount of collaboration possible between two tasks. This can only occur when the factors between two distinct tasks are so similar that the tasks can be joined together to create a new task with a duration between the durations of the two tasks collapsing into one another. Node collapse is illustrated in Figure 12.

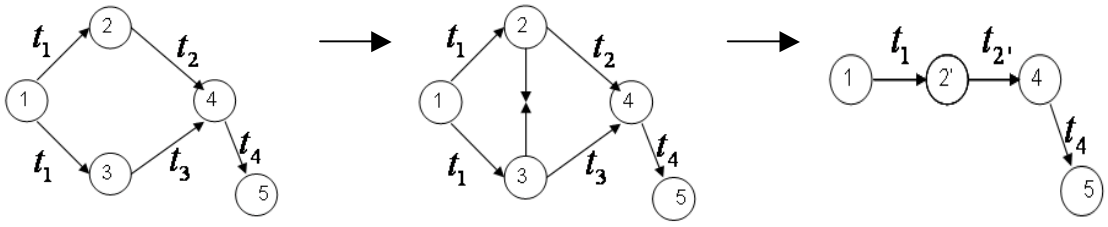


Figure 12: Example of Node Collapsing

As a result of node collapse, two nodes would absorb each other, creating a new task whose duration is a weighted average of the original tasks' durations. The number of nodes and arcs in both linear programs are reduced. Due to this reduction in nodes and arcs, the number of constraints in the network flow and time/cost linear programs are condensed and thus the number of variables affecting the objective functions is cut. Node collapse can be used best in areas where this drop in constraints and variables inevitably cause a shortening of the total project duration. Figure 13 shows how the collapsing shown above affects the task-oriented linear programs.

Figure 13: Task-Oriented Linear Program with Node Collapsing

Summary

The methods described in this section provide a mathematical model for simple forms of collaboration in a complex construction project. The impact of simple forms of collaboration between pairs of tasks can be modeled by node collapse in the task-oriented network for the critical path problem. Leaks between nodes can be used to identify the best pairs of nodes for collaboration.

There are two important features of the linear programs that require further study. The first is range analysis for the optimal solution in the network flow problem. The original optimal basis will not remain optimal for an arbitrarily large magnitude of leak. While the dual problem does assign a value to the collaboration modeled by the leak, an additional step is needed to determine the maximum possible impact. The second problem for further study is the detailed change in the optimal solution for the node collapse approach. Once again, the original optimal solution (pre-collapse) provides an initial guess for the new optimal solution, but it will not remain optimal after collapse.

Conclusion

In conclusion, collaboration can be modeled in two linear programs. These linear programs are duals of one another. They are formed from the event-oriented and task-oriented networks designed from the precedence relationships of the activities. The critical path of the project will assist in identifying possible areas for collaboration.

Collaboration will affect the linear programs through the use of leaks and node collapse. Leaks occur in the event-oriented network. They are the result of an activity collaborating with another activity with an effect that the activity's duration will be reduced. This is shown in the event-oriented linear program as a forced allocation of flow around the activity being helped, causing a percentage of the activities duration to be removed from the total duration of the project. Node collapse occurs in the task-oriented network. They are the result of an activity collaborating with another activity with an effect that both activities are absorbed into each other. The result is a single activity with duration time between the two original activity's durations. This is shown in the task-oriented linear program as a reduction of constraints and decision variables.

With the growing complexity in Civil Engineering projects today, collaboration must be enforced amongst the activities. Through the use of leaks and node collapse, beginning steps can now be taken to derive mathematical means for calculating the effects of collaboration in a project.

Bibliography

1. Benson, Ben., Critical Path Methods in Building Construction. Prentice-Hall, Inc. 1970.
2. Bradley, Stephen P., Hax, Arnoldo C., Magnanti, Thomas L., Applied Mathematical Programming. Addison-Wesley Publishing Company, Inc. 1977.
3. Fondahl, John W., A Non-Computer Approach to the Critical Path Method. Dept. of Civil Engineering, Stanford University. 1962.
4. Moder, Joseph J., Project Management with CPM, PERT, and Precedence Diagramming. Van Nostrand Reinhold Company, Inc. 1983.
5. Phillips, Don T., Fundamentals of Network Analysis. Prentice-Hall, Inc. 1981.
6. Vanderbei, Robert J., Linear Programming: Foundations and Extensions. 2nd Edition. Kluwer Academic Publishers Group. 2001.
7. Priluck, Herbert M. Hourihan, Peter M., Practical CPM for Construction. R.S. Means Co., Inc. 1968.
8. Stark, Robert M. Nicholls, Robert L., Mathematical Foundations for Design: Civil Engineering Systems. McGraw-Hill, Inc. 1972.
9. Mitropoulos, Panagiotis, Tatum, C. B., "Management-Driven Integration." Journal of Management in Engineering. January/February (2000). 48 – 58.
10. Guimera, Roger, Uzzi, Brian, Spiro, Jarrett, Nunes-Amaral, Luis A. *Team assembly mechanism determine collaboration network structure and team performance*, Science, 308 (9) April, 2005.

Table of Figures

Table of Tables

Table of Equations

Appendix A: Project Presentation

Optimization Methods for Project Integration

Heather LaHart
Mark Meko
April 17th, 2007

Advisor: Prof. Arthur Heinricher

Introduction

- In Civil Engineering Projects the use of the Critical Path Method as a means to analyze a project is highly useful and quite common.
- CPM allows for in-depth analysis of a project and provides a variety of solutions that it is the most accurate and efficient system of network scheduling today.

A variety of aspects like **TIME, COST**, and **MANAGEMENT** affect the **EFFIENCY, QUALITY, & VALUE** of a project.

When all these aspects of a project interact and collaborate with one another, it is called ***INTEGRATION***. *So…*

…IS THERE A WAY TO REPRESENT COLLABORATION?

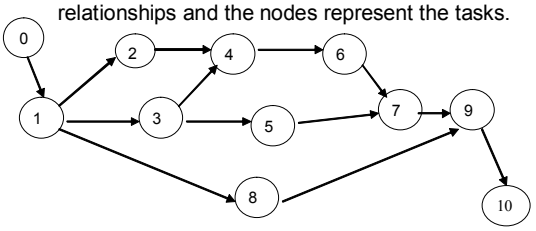
Presentation Overview

- Network Flows
  - Critical Path Method
- Linear Program Formulation
- Task Classification
- Modeling Collaboration
  - Node Collapse
  - Leaky Nodes
- Collaboration Effect on LP
- Analysis and Results
- Conclusion and Recommendations

Task-Oriented Network

- Using precedent relations between tasks in a project, a task-oriented network can be designed to represent the order of the tasks throughout the process.
- This network has arcs represent the precedence relationships and the nodes represent the tasks.

Activities	Predecessors	Duration
0	-	0
1	0	40
2	1	14
3	1	21
4	2,3	21
5	3	10
6	4	1
7	5,6	21
8	1	55
9	7,8	1
10	9	0



Task No.	Name
1	Project Definition
2	Hire Architect
3	Hire CM
4	Schematic Design
5	Project Budget

Task No.	Name
6	SD Berkey Approval
7	Conceptual Design
8	Acquire Land
9	City Approval
10	Project Phase Completion

Task-Oriented LP Formulation

- Each task is assigned a variable t_i representing its early start time based off of similar predecessors where the subscript i corresponds to the precedence relationship.

Activities	Predecessors	Duration	Early Start Times
0	-	0	-
1	0	40	t_1
2	1	14	t_2
3	1	21	t_2
4	2,3	21	t_3
5	3	10	t_4
6	4	1	t_5
7	5,6	21	t_6
8	1	55	t_2
9	7,8	1	t_7
10	9	0	t_8

$$\min z = t_8 - t_1$$
$$t_2 - t_1 \geq 40$$
$$t_3 - t_2 \geq 21$$
$$t_4 - t_2 \geq 21$$
$$t_5 - t_3 \geq 21$$
$$t_6 - t_4 \geq 10$$
$$t_6 - t_5 \geq 1$$
$$t_7 - t_6 \geq 21$$
$$t_7 - t_2 \geq 55$$
$$t_8 - t_7 \geq 1$$
$$t_i \text{ unrestricted}$$

$$\min \underline{b}^T \underline{t}$$
$$s.t. A^T \underline{t} \geq \underline{\tau}$$
$$\underline{t} \text{ unrestricted}$$

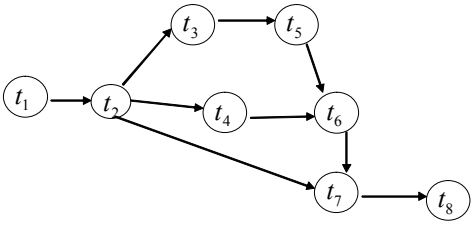
$$t_{task} \geq t_{predecessor} + \tau_{predecessor}$$

- Note: subscript of τ corresponds to task number, unlike t whose subscripts correspond to precedence relationships.

Event-Oriented Network

- Also using precedent relations between task, an event-oriented network can be designed to represent the flow of supply/demand throughout the project.
- This network has the arcs represent the tasks and the nodes represent the precedence relationships.

Activities	Predecessors	Early Start Times
0	-	-
1	0	t_1
2	1	t_2
3	1	t_2
4	2,3	t_3
5	3	t_4
6	4	t_5
7	5,6	t_6
8	1	t_2
9	7,8	t_7
10	9	t_8



Task No.	Name
1	Project Definition
2	Hire Architect
3	Hire CM
4	Schematic Design
5	Project Budget

Task No.	Name
6	SD Berkey Approval
7	Conceptual Design
8	Acquire Land
9	City Approval
10	Project Phase Completion

Event-Oriented Formulation

- The linear program constructed from the event-oriented network allow for the analysis of maximizing the percentage of the task's durations which contribute to the total project duration.
- In doing this one can find what tasks of the project are contributing to the total time or cost and therefore considered critical tasks.
- Since in the task-oriented network, the arcs and nodes are opposite to those in this network, the linear program formed will be the dual of the previous one.

$$\max \tau^T \underline{x}$$
$$s.t. A^T \underline{x} = \underline{b}$$
$$\underline{x} \geq \underline{0}$$

$$\max w = 40x_{12} + 21x_{23} + 21x_{24} + 55x_{27} + 21x_{35} + 10x_{46} + 1x_{56} + 21x_{67} + 1x_{78}$$
$$-x_{12} = -1$$
$$x_{12} - x_{23} - x_{24} - x_{27} = 0$$
$$x_{23} - x_{35} = 0$$
$$x_{24} - x_{46} = 0$$
$$x_{35} - x_{56} = 0$$
$$x_{46} + x_{56} - x_{67} = 0$$
$$x_{27} + x_{67} - x_{78} = 0$$
$$x_{78} = 1$$
$$x_{ij} \geq 0$$

$$x_{ij} = x_{jk}$$

or

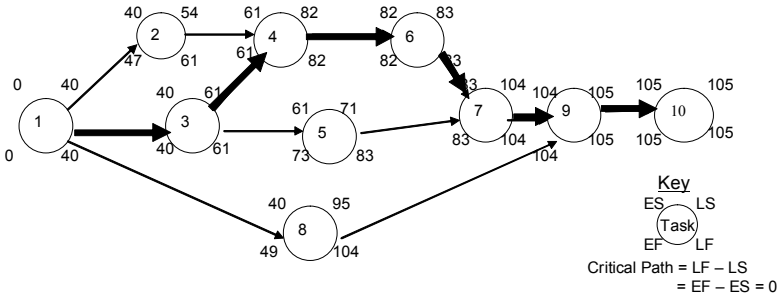
$$x_{ij} - x_{jk} = 0$$

Task Classification for Collaboration

- Tasks that would be considered for collaboration would be determined by the Civil Engineer.
- The Civil Engineer would determine these tasks by classifying them according to
  - Workforce Similarity
  - Task Dependency
  - High Budgeted Tasks
  - Tasks on the Critical Path
- Non-critical tasks are likely candidates to assist other tasks as best they could

Critical Path Method

- Used for the scheduling of tasks in civil engineering projects
- Information required for the method include:
  - listing of all tasks in project
  - projected duration of each task
  - all precedence relationships between tasks
- Method evaluates those tasks whom have no slack time in their scheduling.



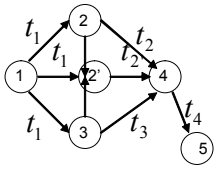
Modeling Collaboration

- Collaboration amongst nodes within the network results in
  - Collapsed Nodes
  - Leaky nodes

  denoted collaborated nodes.
- Leaks are portion of the demand of a node in the event-oriented network to be to reallocated into an accepting node as supply.
- Collapsing is caused by the collaboration of independent tasks resulting in the absorption of one task into the other causing the duration of both tasks to be affected.

Collapsed Nodes

- The Civil Engineer would be able to indicate whether task collaboration would result in the altered nodes collapsing due to common factors of integration.
- Occur when presence of collaboration causes nodes in the task-oriented network to collapse into each other.
- As a result of the collapsed nodes, the number of nodes and arcs in both LPs are reduced. For a single collapse, the number of nodes is reduced by one and the number of arcs is reduced by two.
- Due to this reduction in nodes and arcs, the number of constraints in the task-oriented network is condensed and thus the number of variables affecting the objective functions are cut.



4 variables

$$\min\ t_4 - t_1$$

4 constraints

$$t_2 - t_1 \geq \tau_1$$
$$t_3 - t_1 \geq \tau_1$$
$$t_4 - t_2 \geq \tau_2$$
$$t_4 - t_3 \geq \tau_3$$

3 variables

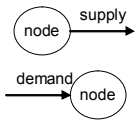
$$\max\ t_4 - t_1$$

2 constraints

$$t_{2'} - t_1 = \tau_1$$
$$t_4 - t_{2'} = \tau_{2'}$$

Leaks

- Occur when presence of collaboration causes a portion of the demand of a node in the event-oriented network to be reallocated into an accepting node as supply.
- Due to this forced additional allocation of resources, the rhs of the constraints of event-oriented LP is altered. A leak would be added to the affecting task's supply variable and the affected task's demand variable.
- The value of these leaks would be determined by the Civil Engineer, denoted in the LP as l_{xy}, xy = task x affected task y.
- As a result of the network flow alteration, the objective function of the task-oriented LP is altered by adding/subtracting percentages of the earliest start time of affected tasks to the total project duration.



$$supply + x_{ij} = x_{jk} + demand$$

$$-x_{12} = -1$$
$$\cdot$$
$$\cdot$$
$$x_{ij} - x_{jk} = l_{xy}$$
$$x_{rs} - x_{st} = l_{xy}$$

$$\min t_n + l_{xy}(t_y - t_x) - t_1$$

Collaboration Modeling Analysis

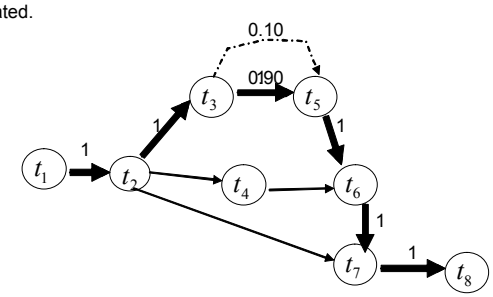
- When there exists a collaboration of tasks in the network structure that creates a change in the original linear program and results in a new LP with correlating payoffs in time and financial costs.
- A comparison of the original LP to the newly altered one and there corresponding time and financial costs will provide a way in which the project's integration can be measured.
- Any drop in total project duration would show a positive effect of the collaboration.

Results

To show the effects of leaks, the event-oriented network is used.

A leak of 0.10 is created between nodes 3 and 5. This is essentially saying that task 4 demand is being reduced by one tenth and reallocated to the proceeding supply of node 5. This in turn causes a reduction in the total project duration.

```
>> primallinprog(0,0)          >> primallinprog(-.10,.10)
Optimization terminated.       Optimization terminated.
x =                            x =
    1.0000                         1.0000
    1.0000                         1.0000
    0.0000                         0.0000
    0.0000                         0.0000
    1.0000                         0.9000
    0.0000                         0.0000
    1.0000                         1.0000
    1.0000                         1.0000
    1.0000                         1.0000
fval = 105.0000                fval = 102.9000
```



Conclusions

- Effects of collaboration can be expressed through a linear program by means of leaks and node collapsing.
- Node collapsing is performed on the task-oriented network. It causes a break down in the number of constraints and variables in the associated linear program, while also altering the durations of the affected tasks.
- Leaks are performed on the event-oriented network. They occur when presence of collaboration causes a portion of the demand of a node in the event-oriented network to be reallocated into an accepting node as supply.
- Leaks will have a positive effect for those tasks who can decrease the demand of other tasks which have a smaller early starting time.
- This can be thought of as having a task waiting to begin supply input and/or assistance to other tasks who are underway.

Recommendations

- Using multi-layer linear programming techniques, formulate a global model which would encompass both minimizing cost and time simultaneously.
- The drop in total project duration should be weighed against the cost associated with the collaboration .

Appendix B: Matlab® Code

```
function primallinprog(l1,l2)
f = [40;21;21;55;21;10;1;21;1];
Aeq = [-1 0 0 0 0 0 0 0 0 ;1 -1 -1 -1 0 0 0 0 0;0 1 0 0 -1 0 0 0 0;0 0 1 0 0 -1 0 0 0;0 0 0 0 1
0 -1 0 0;0 0 0 0 0 1 1 -1 0;0 0 0 1 0 0 0 1 -1;0 0 0 0 0 0 0 0 1];
beq = [-1;0;0;0;0;l1;l2;1];
lb = zeros(9,1);
[x,fval,exitflag,output,lambda] = linprog(-f,[],[],Aeq,beq,lb,[]);
x
fval
lambda.eqlin
end



function duallinprog(l1,l2)
f = [-1;0;l2;0;0;l1;0;1];
A = [-1 1 0 0 0 0 0 0;0 -1 1 0 0 0 0 0;0 -1 0 1 0 0 0 0;0 -1 0 0 0 0 1 0;0 0 -1 0 1 0 0 0;0 0 0
-1 0 1 0 0;0 0 0 0 -1 1 0 0;0 0 0 0 0 -1 1 0;0 0 0 0 0 0 -1 1];
b = [40;21;21;55;21;10;1;21;1];
lb = zeros(8,1);
[x,fval,exitflag,output,lambda] = linprog(f,-A,-b,[],[],lb,[]);
x
fval
lambda.ineqlin
end
```