Automation in Construction 39 (2014) 145–151

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon

Optimized acceleration of repetitive construction projects

Ibrahim Bakry *, Osama Moselhi, Tarek Zayed

Department of Building, Civil and Environmental Engineering, Concordia University, Montreal, Canada

ARTICLE INFO

Article history:
Accepted 5 July 2013
Available online 27 July 2013

Keywords:
Repetitive projects
Acceleration
Linear scheduling

ABSTRACT

Contractors and/or owners frequently need to accelerate the delivery of construction projects. Contractors may have to accelerate in order to benefit from contractual bonus, avoid penalties, recover from delays and/or avoid undesirable weather and site conditions. Owners, on the other hand, may order acceleration to meat business and operational opportunities. This paper presents an algorithm for schedule updating, dynamic rescheduling and optimized acceleration of repetitive construction projects. Schedule updating captures the exact progress on site. Dynamic rescheduling aims at capitalizing on the repetitive nature of the project to fine-tune the remaining portion of the project. Optimized acceleration presents an optimized time–cost trade-off that is tailored for repetitive projects. Through a set of iterative steps, the optimized acceleration procedure divides each activity into segments and identifies the segments that would shorten project duration if accelerated. For those identified segments, the ones with the least cost slope are selected and queued for acceleration. Through the proposed segmentation of activities this algorithm provides optimum allocation of additional acceleration resources, thus is rendered capable of identifying least cost acceleration plans. The algorithm allows users to select among different acceleration strategies such as working overtime, working double shifts, working weekends, and employing more productive crews. The presented algorithm maintains work continuity and accounts for typical and non-typical activities. The algorithm is implemented in a spreadsheet application, which automates calculations, yet allows users to fine tune the algorithm to fit the project at hand. The developed algorithm is applied to a case study drawn from literature in order to illustrate its basic features and demonstrate its accuracy.

1. Introduction

Repetitive construction projects are identified as construction projects formed of recurring units, each unit consisting of the same group of sequential activities. This unique characteristic paves the way for making considerable savings on time and cost by maintaining the continuity of crews and different resources involved in repetitive projects. Maintaining work continuity in repetitive projects helps achieve those time and cost savings through maintaining a constant workforce by reducing firing and hiring of labor, retaining skilled labor, maximizing use of learning curve effect and minimizing equipment idle time [12]. However, maintaining work continuity forms an additional constraint when planning and managing repetitive projects. Consequently, using traditional scheduling and planning tools and techniques to manage repetitive projects has been widely criticized [24], which highlights the need for developing special tools and techniques specifically designed to suit repetitive projects.

Shortly after the launch of critical path method (CPM) as a scheduling technique for non-repetitive projects in the mid-1950s, researchers started investigating enhancements and additions to the CPM technique. Possibly the two most famous and most practical outcomes of researchers' efforts were resource leveling and schedule compression techniques [21]. This paper is concerned with schedule compression, which is sometimes referred to as project time reduction, least-cost expediting, project compression, least-cost scheduling, project cost optimization, optimized scheduling, scheduling with constraints, project acceleration and project crashing. Schedule compression is about finding the delicate balance between the increase in direct cost, due to assigning additional resources and the decrease in indirect cost, due to shorter project duration. Schedule compression techniques can be divided into two main groups, heuristic and optimization techniques. Heuristic techniques are simpler and require less computational effort but present good solutions that are not necessarily the optimum solutions. Examples of heuristic methods are genetic algorithms [17,27] and harmony search [8]. On the other hand, optimization techniques find the optimal solution but are more difficult to create, need considerable computation effort, and are not efficient when handling large scale projects [25]. Optimization techniques include integer programming [22], linear programming [18] and dynamic programming [4]. Recent but not so common additions to the common scope of the schedule compression problem include utilizing discounted cash flows and considering maximizing profit as a criterion instead of reducing total project cost [1,3,25]. The above mentioned techniques address different aspects of scheduling compression, but all of them only address traditional non-repetitive project schedules,

* Corresponding author.
E-mail address: bakryibrahim@yahoo.com (I. Bakry).

0926-5805/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.autcon.2013.07.003

thus rendered not suitable for repetitive projects as will be detailed later.

In comparison to traditional projects, repetitive projects usually have a smaller number of activities which makes exact optimization techniques a feasible optimization option. For repetitive projects, techniques are available for performing different optimization procedures at the initial scheduling stage but they perform schedule acceleration. Linear programming was used for single and multi-objective optimization [16,23] and dynamic programming was used as well [5]. These optimization techniques were designed to find crew formations - and optimum interruption vectors (El-Rayes1997) - that would yield a schedule with optimum duration and/or cost. Genetic algorithms (GA) were used as a heuristic technique [14], and in some efforts GA were used in conjunction with dynamic programming [7]. These techniques are capable of considering different alternatives and identifiying an optimum solution, all having varying yet acceptable degrees of practicality at the initial scheduling stage. However, when it comes to accelerating a project, each activity is divided into segments and each segment can be accelerated using different strategies, and this acceleration is performed using incremental assignment of acceleration resources, which results in an almost infinite number of possibilities. Therefore investigating the whole schedule for acceleration would lead to a lot of redundant calculations. That being said, the need is established to adopt an approach that is capable of identifying critical segments of activities and nominates these segments for acceleration, which is bound to save a lot of computational time and effort. Moreover, this problem calls for an algorithm that is capable of incremental assignment of additional resources until an optimum solution is reached, instead of investigating every single positive solution.

This paper presents an algorithm for schedule updating, dynamic rescheduling and optimized acceleration algorithm for repetitive construction projects. Two types of updating are incorporated in the algorithm, a traditional updating procedure where site updates are utilized to bring the executed part of the project up to date, and a dynamic rescheduling procedure that allows for utilizing the exact durations of the executed portion of each activity to re-evaluate and reschedule the duration and the time buffers of the remaining portion of the schedule. After updating the project's schedule, the algorithm performs a heuristic time–cost tradeoff that is custom-made for repetitive projects in an effort to locate the optimum plan for accelerating the project.

2. Updating and dynamic rescheduling

More often than not, construction projects do not proceed exactly as planned. This makes the process of schedule monitoring and updating a basic component of any project management plan. The presented algorithm begins by schedule updating, followed by dynamic rescheduling and ends by optimized acceleration of repetitive projects. Updating aims at incorporating the actual data of the executed portion of the project into the schedule. This is performed through replacing planned start and end dates with the actual dates of completed activities. In addition, the user is given the opportunity to revise quantities of activities and productivity rates of crews for the uncompleted works of the project. This revision is deemed useful because after the project starts the user has a better knowledge of the project environment, which enables him to refine the estimates for the rest of the schedule. Based on the actual dates entered and the estimates revised, the schedule is recalculated and the project end date is changed accordingly.

The second task before starting the optimized acceleration procedure is dynamic rescheduling. Dynamic rescheduling aims at capitalizing on the repetitive nature of the project. It utilizes the experience and knowledge of project managers and their assessment of the project performance up to reporting date. The dynamic rescheduling aims at re-sizing the time buffers inserted between successive activities. Time buffers are usually built according to the expected uncertainty level affecting the activities involved. Activities in a highly uncertain environment are more likely to suffer from interruptions and/or delays and hence are followed by bigger buffers to provide adequate protection to work continuity, and vice versa. It is common practice to rely on experience when estimating the size of buffers. This subjective approach highlights the need for monitoring and fine-tuning time buffers once the project starts and delays occur. The proposed technique works under the assumption that buffers are calculated for each unit and then aggregated to form a single buffer for each activity. The algorithm at hand utilizes relative weights to add delays that happened and those estimated for the remaining activities. The relative weights are calculated based on the number of units completed to the total number of units, and the number of units to be completed to the total number of units, respectively. The equation below shows how the buffer size is fine-tuned using weighted average.

$$B_n = \left(\frac{U_c}{U_t} \times D_a\right) + \left(\frac{U_r}{U_t} \times B_o\right) \quad (1)$$

where

- B_n is the new buffer per unit
- U_c is the number of units completed
- D_a is the average delay per unit for the completed units
- U_r is the number of units remaining
- B_o is the originally estimated buffer per unit
- U_t is the total number of units

As such, for an activity of 10 day duration scheduled to be repeated throughout 40 units with an estimated delay (buffer) of 1 day per unit, the total activity duration will be 400 days followed by a 40 days buffer. If the schedule is updated after 10 units have been completed and it was noticed that the average delay per unit is 1.2 days instead of 1 day. The new buffer per unit can be expressed as:

$$B_n = \frac{(10 \times 1.2) + (30 \times 1)}{40} = 1.05 \text{ days of delay per unit}$$

This example shows how the estimated buffer for each unit is refined based on the performance experienced on site up to reporting date. When the 1.05 days of anticipated delays for each unit is aggregated, a buffer of 31.5 days will be needed to cover the remaining 30 units. The described updating and dynamic rescheduling flowchart are illustrated in Fig. 1.

3. Identifying activities to accelerate

Selecting the right activities to accelerate in a repetitive project is a key step toward successful project acceleration. Accelerating the wrong activity will lead to spending more money without any effect on a projects duration, or spending more money than needed. In traditional projects such a decision is made easier by the existence of a critical path. In CPM every project schedule includes a critical path or paths, which is a group of sequential activities with a total duration longer than other paths, hence, determining the project's total duration. Crashing any activity on this path would shorten the project's duration. This remains valid until that critical path is no longer the longest path in the network [21]. Things are different in repetitive projects, as many alternatives exist in literature for identifying which activities control a repetitive project's duration. Two well-known methods to identify the critical activities controlling a repetitive projects' total duration are "Controlling Activity Path" for schedules built using Linear Scheduling Model (LSM) [9,10], and "Controlling Sequence" for schedules built using Repetitive Scheduling Method (RSM) [11]. Many comparisons have been made between these two methods highlighting their advantages and disadvantages. Although both successfully identify critical activities, both techniques only account for sequential activities with