

Retaining Flexibility to Maximize Quality: When the Scheduler Has the Right to Decide Activity Durations

Xiaofang Wang and Stephen F. Smith

The Robotics Institute, Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
xiaofanw, sfs@cs.cmu.edu

Abstract

In many resource constrained scheduling applications, the ability to vary task durations provides another degree of freedom for resolving resource conflicts. Furthermore, when each task is associated with a duration-dependent quality profile, activity durations must be determined to maximize overall quality while respecting process deadlines and resource capacity constraints. In this paper, we formulate this type of scheduling problem, which we refer to generally as quality maximization. We develop and empirically evaluate a new precedence constraint posting (PCP) algorithm, along with a number of search control heuristics for solving this class of problems. Our PCP algorithm incorporates linear optimization to set activity durations at each step, and search control heuristics direct the search toward resource feasibility. A central concept in the heuristics we define is use of a measure that combines an activity's *reducible duration* with its quality profile as a basis for determining how to resolve resource conflicts. This concept is found to yield heuristics that exhibit superior performance. Surprisingly, we also find that use of a simple local estimation of quality degradation at each posting step leads to better performance than use of an exact computation at dramatically reduced computation expense. Overall, the experimental analysis indicates that a good heuristic must strike the right balance between minimizing quality loss at each step and retaining flexibility for future duration reduction.

Introduction

Quality maximization scheduling problems are motivated by so-called knowledge-intensive production processes, where knowledge products are created and manipulated by resources (typically human) to meet demands within tight deadlines. Those processes exist within such settings as news reporting, intelligence gathering, and new product research and development. Consider the process of creating an issue of a weekly news magazine. Several different types of activities need to take place: information gathering, data analysis, writing, editing, confirming and duplicating sources, responding to late-breaking events, etc.

In contrast to most problems examined by prior AI scheduling research, these knowledge-intensive processes consider duration change as an additional degree of freedom for decision makers in attempting to satisfy tight resource constraints and meet deadlines. In essence, some activities can be executed in faster time cycles than would be desirable under non-constraining circumstances, with consequent degradation of process quality. The quality of many activities will be a function of the time that can be devoted to it: a story written in an hour by a news correspondent will be lower quality than it would be if a day was spent on it. Given the production deadline associated with publishing the next issue, decisions will frequently need to be made regarding which activities to spend more time on, and the best decisions are those that maximize the quality of the final content of the issue.

We model such a process as a project. In addition to resource constraints and precedence constraints, a linear quality profile is associated with each activity, which specifies the quality of the associated activity's output as an increasing function of time. Hence, an activity can be terminated at any time (subject to a required minimum duration, which represents the minimum quality requirement for each activity), with an output quality proportional to its duration. The goal is to establish resource-feasible activity start times and activity durations that maximize overall output quality given project release and deadline constraints. In the simplest case, overall output quality is defined as the sum of the output quality of each individual activity.

This new scheduling problem presents an interesting challenge for constraint-based scheduling research, where in addition to allocating activities to available resources, it is necessary to determine activity durations in a quality maximizing manner. Although constraint satisfaction problem solving (CSP) techniques have proven effective for several classes of scheduling problems (Baptiste, Le Pape, & Nuijten 2001; Beck 1999; Cesta, Oddi, & Smith 2000; Cheng & Smith 1996; Nuijten & Aarts 1994; Oddi & Cesta 1997; Sadeh 1991; Policella *et al.* 2004), they have not been applied to our duration-dependant quality maximization problem.

Our approach draws on two complementary results from previous research. In (Cesta, Oddi, & Smith 2002), a Precedence Constraint Posting (PCP) procedure for iteratively

transforming a time-feasible solution into a resource feasible solution by successive “levelling” of resource conflicts is shown to provide an effective basis for solving difficult instances of RCPSP/Max¹. Given this result and the related nature of our problem, we adopt this basic schedule generation framework. A second element of our approach is drawn from more classic project management research on the time/cost tradeoff problem (Kelley Jr. & Walker 1959), which addresses the question of whether activity durations should be reduced (by using more resources) at some increase in cost. It has been shown (Fulkerson 1961; Kelley Jr. 1961; Philips & Dessouky 1977) that under the assumption of linear and continuous cost function for each activity, the un-capacitated version of the time/cost tradeoff problem can be efficiently solved to optimality. Noting the similarity of this computation to the computation required to compute maximum quality durations in a project network, we utilize this procedure as the “constraint propagation” step of the solution procedure, to adjust activity durations when a new precedence constraint is posted.

We empirically evaluate and analyze various search control heuristics for solving resource conflicts. Finally, we find the behaviors of most heuristics can be understood as a tradeoff between *quality* and *flexibility*: the quality loss that results from each constraint posting step, and the flexibility that is retained for future duration reduction.

The remainder of the paper is organized as follows. We start with an overview of related work, followed by the formulation of the problem. Then we present our constraint posting algorithm and define a set of constraint-posting heuristics. Finally, we report the experimental results that indicate the relative performance of various heuristics and give an analysis and insights from the results.

Related Work

There are a few different areas of research that have considered issues relevant to factoring varying duration into a scheduler’s decisions.

Time-cost Tradeoff Problems (Project Crashing)

Within the operations research community, some work has attempted to bridge the gap between the time-cost tradeoff problem solution and scheduling under resource constraints. But this work has restricted attention to either the non-preemptive case with discrete cost functions (Talbot 1982; Icmeli & Erenguc 1996), or the preemptive scheduling case (Slowinski 1980; 1981). To our knowledge, there is no work dealing with the problem addressed in this paper, which can be seen as the resource-constrained, non-preemptive time-cost tradeoff problem with linear and continuous cost functions.

¹RCPSP/Max is an extended version of the resource constraint project scheduling problem(RCPSP) formulation considered here that additionally incorporates minimum and maximum time lags between pairs of activities.

Anytime Scheduling

In anytime scheduling research, the time of termination of a given activity is decided by the scheduler. So a tradeoff must be made between output quality and computation time allocated to algorithms whose results degrade in quality as computation time decreases (Horvitz 1987; Dean & Boddy 1988; Russel & Wefald 1991; Zilberstein 1993). But these problems have tended to involve the allocation of a single computational resource. The problems of scheduling anytime tasks with both precedence constraints and resource constraints have received less attention. Schwarzfischer (Schwarzfischer 2003b; 2003c; 2003a) has recently explored the idea of combining anytime scheduling and deadline scheduling, which he refers to as quality/utility scheduling. But his focus is on interruptible tasks in a stochastic domain(with probability distributions on task release times), and accordingly, he uses totally different methodologies with ours.

Hybrid Scheduling Techniques

In recent years there has been increasing interest in hybrid solutions to scheduling problems, which involve an integration of Constraint Programming (CP) and Linear Programming (LP) techniques (Hooker 2004; Beringer & De Backer 1995; El Sakkout, Wallace, & Richards 1998; El Sakkout & Wallace 2000; Ajili & El Sakkout 2003). Among them, the probe backtrack search procedure of (El Sakkout & Wallace 2000), which uses an LP optimization to continually update a tentative global solution during the search, is quite similar to the approach we adopt in this paper. However, this work has been principally concerned with the problem of minimum perturbation revision of schedules in response to environmental change (including duration change). We focus instead on the quality maximization problem, and our main interest is to gain insight into the types of search heuristics that might be integrated into such a search framework to obtain good scalable performance.

Temporal Preference Networks

In recent research with temporal preference networks (Khatib *et al.* 2001), the problem of interest is to optimize preference associated decisions such as how long a given activity should last, when it should start, or how it should be ordered with respect to other activities. The preference functions associated with different intervals in these networks are similar with our quality profiles. However, our problem is different from theirs in that we additionally consider resource constraints. The infinite capacity version of our problem can be viewed as one type of temporal reasoning problem with hard constraints (precedence relations) and preferences (each activity prefers to run as long as possible).

The Problem Formulation

Given a project composed of a set of non-preemptive activities $V = \{a_1, \dots, a_n\}$, a set of precedence constraints, a set of quality profiles and resources with limited capacity, the problem is to decide the start time and the end time of each activity so as to maximize the sum of qualities of all the ac-

tivities. In the remainder of the paper, we use notations and make assumptions as follows:

- r_i : release date for activity a_i ,
- D : a common deadline for all the activities, or the whole project's deadline,
- s_i : decision variable, the start time of activity i ,
- e_i : decision variable, the end time of activity i ,
- t : current time,
- q_i : output quality from activity a_i , which is a non-decreasing linear and continuous function of its duration with slope k_i , i.e., $q_i = k_i \cdot (e_i - s_i)$,
- d_i : minimum duration² for activity a_i ,
- E : the set of edges in the precedence graph, if $(i, j) \in E$, activity a_j should start after activity a_i is completed,
- C : constant resource capacity over the entire horizon, without loss of generality, each activity is assumed to require one unit of resource.³
- Q : objective, the total quality output from the project.

Given a schedule $S = (s_i, e_i)_{i \in V}$, let

$$A(S, t) := \{i \in V \mid s_i \leq t < e_i\} (t \geq 0)$$

be the set of activities in progress at time t , also called the *active set* at time t . Let

$$R(S, t) := |A(S, t)|$$

be the amount of resource used at time t . So the *resource constraint* is

$$R(S, t) \leq C$$

Then the problem can be formulated as follows:

maximize

$$Q = \sum_{i=1}^n q_i = \sum_{i=1}^n k_i \cdot (e_i - s_i) \quad (1)$$

subject to

$$e_i \leq s_j, (i, j) \in E, \quad (2)$$

$$R(S, t) \leq C, \quad (3)$$

$$d_i \leq e_i - s_i, i \in V, \quad (4)$$

$$r_i \leq s_i, i \in V, \quad (5)$$

$$e_i \leq D, i \in V, \quad (6)$$

(2), (3), (4), (5) and (6) are precedence, resource, minimum duration, release date and due date constraints respectively.

²This assumption makes sense in practice because we are required to invest at least some amount of time to each knowledge processing activity in order to guarantee basic quality.

³Our model can easily be extended to problems where each activity or task can require more than one unit of resource. In that case, we can divide an activity into several sub-activities, each of which requires one unit of resource and has a quality curve with a smaller slope, and add temporal constraints to synchronize the sub-activities' start and end times.

Complexity Analysis

For the single capacity ($C = 1$) problem, there exists a polynomial algorithm to solve it optimally. However, the multiple capacity ($C > 1$) problem can be proved to be *NP*-complete. Due to space constraints, we refer the reader to (Wang & Smith 2004) for details.

A Constraint Posting Algorithm

We adopt a precedence constraint posting (PCP) approach to solve this problem, motivated by the effective prior use of this type of algorithm in standard RCPSP contexts (Cesta, Oddi, & Smith 2002). Since our goal is to maximize quality, we adapt the traditional PCP framework to take advantage of an optimization procedure for solving the related time/cost tradeoff problem (Kelley Jr. & Walker 1959). We first define our basic PCP procedure, and then propose several constraint posting heuristics.

The Precedence Constraint Posting Framework

Following the tradition in previous PCP work, we begin by defining the notion of a contention peak.

Given a schedule, a **contention peak** (or simply a peak) is a set of activities, each of which simultaneously requires one unit of resource and whose total resource requirement exceeds the resource capacity in the maximal time window $[t_1, t_2]$, where t_1 is the start time of one of the activities in this peak, and t_2 is the end time of one of the activities in this peak. The **length** of a peak is $(t_2 - t_1)$.

As usually formulated, a *PCP* scheduling procedure acts to transform a time feasible solution into a resource feasible solution by eliminating all contention peaks. At each step of the search, a peak is selected for "levelling", and a precedence constraint is posted between some pair of activities in the peak to achieve this effect. Each time a constraint is posted, constraint propagation is performed, to compute new activity start and completion times and confirm that the solution is still feasible.

In our problem, however, the situation is somewhat different. If we ignore the resource constraint of the problem, we are left with essentially the time-cost tradeoff problem. The goal of this problem is to meet the project's due date, while minimizing total crashing costs. As previously noted, if the cost function for each activity is linear and continuous, the problem can be optimally solved in polynomial time by linear programming (Fulkerson 1961; Kelley Jr. 1961; Philips & Dessouky 1977).

Given this result, we can reformulate the *PCP* procedure to be one of transforming an initial "quality optimized" initial solution into a resource-feasible solution that optimizes quality through elimination of contention peaks. Essentially this corresponds to replacing the traditional constraint propagation step with a corresponding call to an LP Solver to solve the time/cost tradeoff problem for the current activity precedence graph. If the LP solver returns a solution, it will be the optimal quality solution for this graph. If it returns failure, then the search has reached an infeasible state. The PCP framework is illustrated in Fig. 1 and the algorithm procedure is described in Fig. 2.

We concentrate on solving problems without backtracking because our main interest is in evaluating the performance of different search heuristics. Hence, the algorithm is not complete. We actually also ran another series of experiments using a time-bounded backtracking version of our algorithm; in these experiments the number of solved problems by various heuristics did increase, but the relative comparative performance of the heuristics that we consider below remains the same.

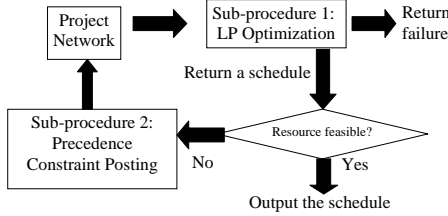


Figure 1: The Precedence Constraint Posting Framework

Constraint-posting Heuristics

In this section, we design three classes of heuristics for identifying which activities to sequence in order to resolve conflicts. Because a peak represents a bottleneck in the project network, we always choose activities in some peak to sequence.

Basic Choice Criteria In designing constraint-posting heuristics, our goal is to post constraints to effectively reduce the resource conflict at minimum expense of quality loss. Intuitively we prefer to post constraints between activities with the smallest quality **slopes** (k_i), because shrinking these activities would seem to lead to the smallest quality loss. But slopes only represent the static information of activities. A more informed criterion is one that considers the dynamically changing activity durations during search. To this end, we introduce a second criterion for choosing activities to sequence - choose the activities with the largest **ratios** of “reducible duration” to slope:

$$ratio = \frac{e_i - s_i - d_i}{k_i},$$

where the reducible duration of an activity is the amount greater than its minimum duration. From this choice, we actually bias on the choice toward activities with small slope and long reducible duration.

We define two basic heuristics that choose two activities to sequence based on these criteria, denoted as “Global-slope” and “Global-ratio” respectively in Table 1. In both cases, we assume that once two activities have been selected, they are sequenced according to the rule of *Earliest Start Time First*.

Focusing on the Longest Peak Observing that an activity’s length is closely related to the quality contribution from itself and the other activities running in parallel with it, we

Heuristic-based Constraint Posting Algorithm

Input: A set of activities with constraints.

Output: A solution or failure

1. **loop**
2. apply LP solver to find infinite capacity optimal solution
3. **if** there is no temporal feasible solution
4. **then return** failure
5. **else begin**
6. detect contention peaks
7. **if** there is no peak
8. **then return** solution
9. **else** constraint posting:
sequence a pair of activities in some peak
10. **end**
11. **end-loop**

Figure 2: Heuristic-based Constraint Posting Algorithm

can define Longest Peak First (denoted as LPF in the following tables) as an optional preprocessing step that focuses application of either of the basic choice criteria. By using LPF, the pair of activities to sequence is selected from the set of activities in the current longest peak based on ratio or slope as before.

Heuristics Incorporating Look-ahead An alternative way to choose a pair of activities is to choose the first activity according to either the criterion of slope or ratio, and to choose the second activity and sequence both activities according to another criterion—**quality loss**. Quality loss is the difference between the two optimal⁴ quality values obtained from problems with and without the candidate precedence constraint respectively. The optimal quality will remain unchanged or degrade after posting one constraint. We try to minimize the degradation—quality loss. To compute quality loss exactly, we can run the linear programming solver(LP) twice (before and after posting a given constraint), and this heuristic is denoted as “Ratio-Exact” in Table 1. But this is very costly in computation time. Hence, we design the following estimation methods.

Quality Loss Estimation Method 1 - Loss Only

Suppose we choose the partially overlapped activity pair $\langle a, b \rangle$. Without loss of generality, we assume they have slopes $k_a \geq k_b$, start times s_a and s_b , end times e_a and e_b , minimum durations d_a and d_b . Based on the relative position of a and b , we can either sequence a before b or b before a , and then shrink a or b or both to eliminate the overlapping. If we assume all other activities’ start times and end times are fixed in the solution with the newly posted constraint, then the minimum quality loss can be estimated from resolving the resource conflict locally. Given the different potential relative positions of a and b , we use a general form to compute the estimated minimum quality loss. Let A_{loss} be the

⁴Here, optimality means the solution for the problem assuming no limits on resource capacity. It is solvable by LP, as we mentioned before.

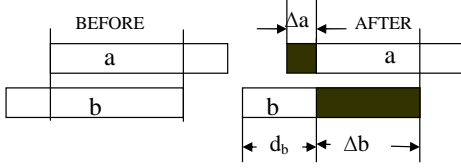


Figure 3: An Example for Quality Loss Estimation

quality loss due to a 's duration change, B_{Loss} be the quality loss due to b 's duration change.

$$QualityLoss = A_{loss} + B_{Loss} \quad (7)$$

If $s_b - s_a \geq e_a - e_b$, then sequence a before b , so

$$A_{loss} = k_a * [\max\{e_a + d_b, e_b\} - e_b] \quad (8)$$

$$B_{Loss} = k_b * [\min\{e_b - d_b, e_a\} - s_b] \quad (9)$$

If $s_b - s_a < e_a - e_b$, then sequence b before a , so

$$A_{loss} = k_a * [\max\{s_b + d_b, s_a\} - s_a] \quad (10)$$

$$B_{Loss} = k_b * [e_b - \max\{s_a, s_b + d_b\}] \quad (11)$$

We illustrate the estimation method with an example in Fig. 3. Because $s_b - s_a < e_a - e_b$, we use the formula in equations (10) and (11). As shown in Fig. 3, $s_b + d_b > s_a$, so the quality loss can be estimated as follows:

$$A_{loss} = k_a * [s_b + d_b - s_a]$$

$$B_{Loss} = k_b * [e_b - s_b - d_b]$$

$$QualityLoss = A_{loss} + B_{Loss}$$

From observation, we can see the best quality solution is to sequence b before a , shrink b 's duration to minimum duration, and shrink a 's duration a little bit to leave enough space for b . This corresponds to the above computation. Actually, the best quality solutions for all possible relative positions of a and b are generalized in equations(7-11).

If the durations of all other activities remain unchanged after re-optimization of the new problem (with the newly posted constraint), the above quality loss estimation will be the actual quality loss. However, in many cases, if we post a new constraint, the other activities will change, too. Thus this method only provides an estimate. This heuristic is denoted as "Ratio-Loss" in Table 1.

Quality Loss Estimation Method 2 - Loss and Gain

In the above quality loss function, we ignored some quality gain from a and b 's predecessors or successors. If a shrinks by Δa and b shrinks by Δb , a 's predecessors or successors can grow longer up to Δa , similarly, b 's predecessors or successors can grow longer up to Δb . Let the sets of activities which are likely to grow longer be Set_a and Set_b . For example, in Fig.3, Set_a is the set of activity a 's predecessors, and Set_b is the set of activity b 's successors. The quality gain can be estimated as follows:

$$QualityGain = \sum_{i \in Set_a} k_i * \Delta a + \sum_{i \in Set_b} k_i * \Delta b$$

Table 1: Constraint Posting Heuristics

Heuristics	Basic	LongestPeakFirst	Lookahead
LPF-slope	Slope	Yes	No
Global-slope	Slope	No	No
Slope-Exact	Slope	No	Exact
LPF-ratio	Ratio	Yes	No
Global-ratio	Ratio	No	No
Ratio-Exact	Ratio	No	Exact
Ratio-Loss	Ratio	No	Loss Only
Ratio-LossGain	Ratio	No	Loss and Gain
LPF-Ratio-Loss	Ratio	Yes	Loss
LPF-Slope-Loss	Slope	Yes	Loss

If we subtract the quality gain from the above quality loss function, the modified quality loss estimation function is as follows:

$$QualityLoss = k_a * \Delta a + k_b * \Delta b - \sum_{i \in Set_a} k_i * \Delta a - \sum_{i \in Set_b} k_i * \Delta b$$

This heuristic is denoted as "Ratio-LossGain" in Table 1. We notice that the computation of quality gain is overly optimistic because the activities can be constrained by their other successors or predecessors, and so that they may not be stretchable in all cases. However, quality gain also includes the slope information about this activity' predecessors or successors. It is thus actually an indicator of how "connected" this activity is with other activities.

Experimental Evaluation

To evaluate the performance of the set of heuristics defined above (summarized in Table 1), we generated data sets using precedence constraints defined in a single mode resource constraint project scheduling benchmark problem set.⁵ Three data sets of 400 problems each were generated with the resource capacities of 3, 5, 7 respectively. Each problem has 32 activities. And 32 uniformly distributed random integers in the range [1, 50] were generated to represent the slopes for each problem. We set a global due date = 30. Uniformly distributed random integers in the range [0, 5] were generated to represent the release dates for each problem. Uniformly distributed random integers in the range [1, 3] were generated to represent the minimum durations.⁶ The algorithms were implemented in Visual C++ 6.0, and LINGO 8.0 was used as the LP solver⁷. The CPU time presented in the following tables were obtained on a Pentium IV-2.40 Ghz processor under Windows XP. In the following tables, we compare the performance of heuristics based on following measures:

⁵<http://www.bwl.uni-kiel.de/Prod/psplib/data.html>. Filename: j30.sm

⁶We choose this range for release dates and minimum durations in order not to make the temporal constraints too tight, and hence guarantee that feasible temporal solutions exist for all the problems.

⁷<http://www.lindo.com>

1. *quality(%)*. The average percentage quality normalized to the infinite capacity solution. The infinite capacity solution gives us an upper bound for the capacitated problem.
2. *runtime(sec)*. The average CPU runtime to reach a solution for one problem.
3. *constraints*. The average number of posted constraints to reach a solution for one problem.
4. *solved(%)*. The percentage of problems solved, where a solved problem means the heuristic is able to find a resource feasible solution without backtracking.

The overall performance results are given in Table 2. Because of the fact that some heuristics cannot solve all 400 problems for a given capacity setting, *quality*, *runtime* and *constraints* are calculated based on the commonly solved problems by all heuristics in a given experiment. There are some blanks in Table 2, which is due to the fact that some heuristics sometimes solve much fewer problems than others, and such a small set of commonly solved problems does not provide the representative information we need for comparison in quality and time.

Ratio vs. Slope

First, we compare the relative leverage provided by the two basic choice criteria: ratio and slope. Fig.4 depicts the number of solved problems and the average quality of various heuristics. Ratio-based heuristics refer to “Global-Ratio”, “LPF-Ratio” and “Ratio-Exact”. Slope-based heuristics refer to “Global-Slope”, “LPF-Slope” and “Slope-Exact”. We observe the following:

Observation 1 *Slope-based heuristics yield slightly better quality performance than ratio-based heuristics on commonly solved problems, however, they solve fewer problems than ratio-based heuristics. Furthermore, on harder (smaller capacity) problems, the number of solved problems using Slope-based heuristics degrades significantly while ratio-based heuristics continue to solve most of the problems.*

This indicates that there is a tradeoff between solution quality and the percentage of solved problems. And this tradeoff implies the tradeoff between quality and flexibility. Let us go back to look at the definition of **ratio**. It is the reducible duration divided by quality slope. The larger the reducible duration, the more flexibility for the future shrinking. And for the denominator, **slope**, the smaller it is, the smaller possible quality loss if we shrink this activity. Overall, ratio represents a balance of two goals in our search: **maximizing quality and retaining flexibility in the schedule**. The ratio acts to retain flexibility for future shrinking and also achieves a good quality.

To understand the reason, it is useful to look at the inflexibility in a schedule resulting from slope-based heuristics. Consider a resulting schedule using “Global-Slope” when resource capacity is 5. Fig.5 illustrates the precedence constraints and the durations of all the activities in this schedule. The dark bar represents the minimum duration and the grey bar represents the actual duration. Observe that many small sloped activities are sequenced and congested on one

Table 2: Overall Performance Results

Capacity=7	Quality	Runtime	Constraints Solved	Solved
Ratio-Exact	96.9	13.8	4.63	392
Ratio-Loss	96.9	0.692	3.97	400
Ratio-LossGain	95.2	0.340	1.29	395
LPF-Ratio	95.2	0.355	1.63	400
Global-Ratio	95.3	0.334	1.62	400
LPF-Slope	97.1	0.378	1.78	396
Global-Slope	96.7	0.468	2.45	386
Slope-Exact	97.3	10.3	4.45	359
LPF-Ratio-Loss	96.0	0.404	1.99	400
LPF-Slope-Loss	95.3	0.416	1.67	393
Capacity=5	Quality	Runtime	Constraints Solved	Solved
Ratio-Exact	89.6	31.7	13.0	350
Ratio-Loss	89.5	1.64	11.4	395
Ratio-LossGain	84.0	0.664	3.63	367
LPF-Ratio	84.6	0.938	5.70	400
Global-Ratio	85.0	0.914	5.60	400
LPF-Slope	89.4	0.953	5.93	304
Global-Slope	87.7	1.08	7.05	201
Slope-Exact	89.9	21.5	12.3	217
LPF-Ratio-Loss	87.1	1.09	6.94	399
LPF-Slope-Loss	84.6	0.867	5.12	294
Capacity=3	Quality	Runtime	Constraints Solved	Solved
Ratio-Exact	49.5	174.6	65.3	112
Ratio-Loss	49.0	8.44	60.4	228
Ratio-LossGain				41
LPF-Ratio	47.1	3.46	24.6	303
Global-Ratio	47.0	3.49	25.2	291
LPF-Slope				5
Global-Slope				0
Slope-Exact				4
LPF-Ratio-Loss	48.7	4.6	33.1	290
LPF-Slope-Loss				5

path from the first activity to the last activity and nearly all of them are at minimum durations. Let’s take a close look at the path composed by “checked” activities in Fig.5. The set of activities on this path (ordered by the precedence sequence) is:

$$P = \{1, 4, 5, 13, 17, 18, 21, 23, 30, 32\}$$

Please refer to Table 3 for the data.

All the activities on this path are at their minimum durations. And the length of this path is the project duration. Actually, in schedules produced by slope-based heuristics there are multiple paths like this. They not only make it impossible to shorten the whole project duration, but also make the schedule inflexible for future shrinking. For example, due to capacity reduction, some activities need to be sequenced. Slope-based heuristics bias on choosing smallest sloped activities, which are most likely to be on those congested paths. Sequencing them leads to further shrinkage of their own durations or the durations of some other activities on those paths. This becomes impossible when all

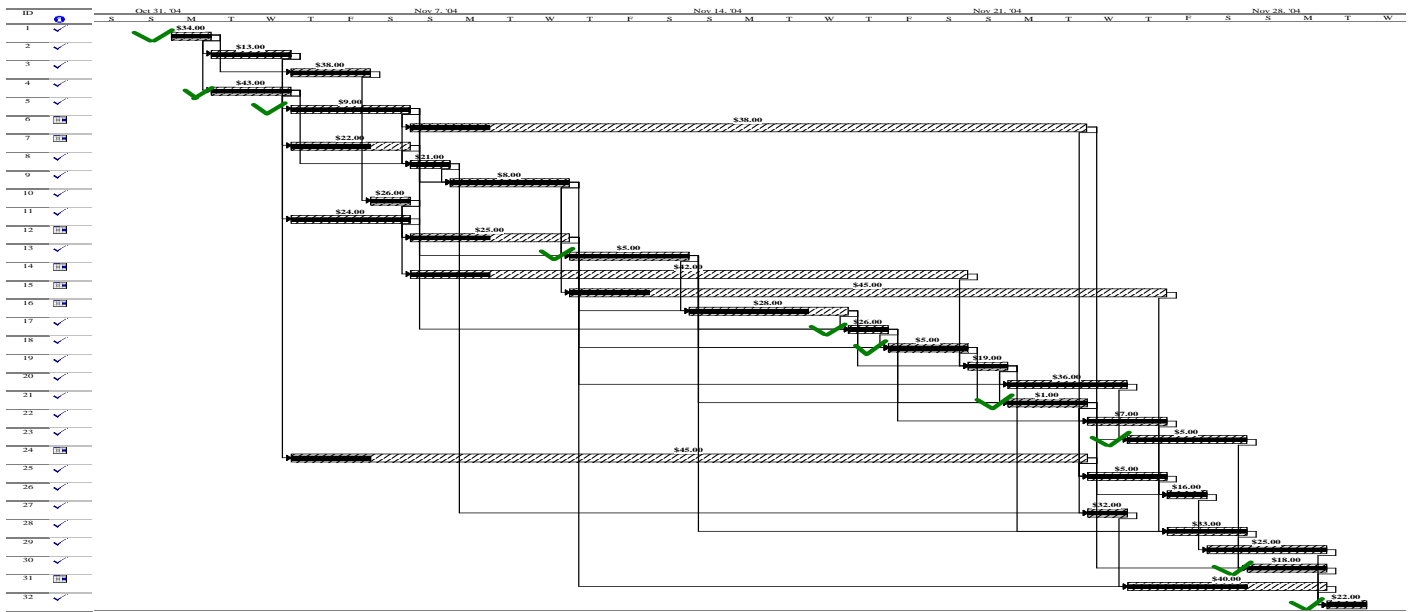


Figure 5: Resulting Schedule of One Problem Solved by Global Slope

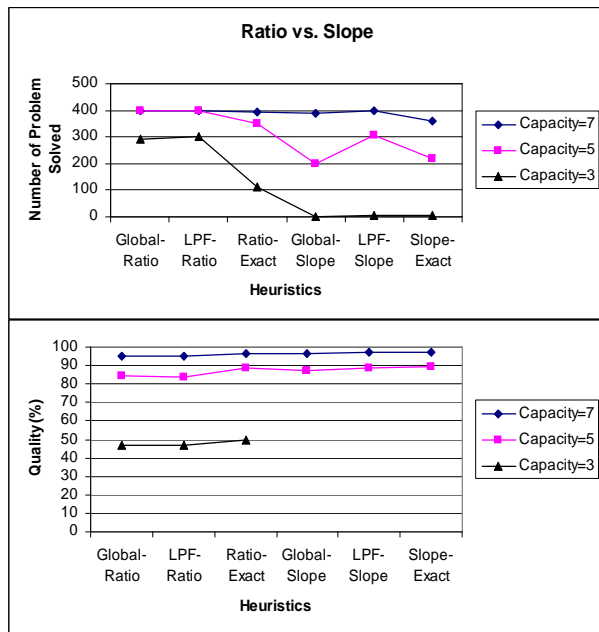


Figure 4: Comparison between Slope and Ratio

Table 3: Data for a Congested Path

Activity ID	1	4	5	13	17	18	21	23	30	32
Duaration	1	2	3	3	1	2	2	3	3	1
Min Duration	1	2	3	3	1	2	2	3	2	1
Slope	34	43	9	5	26	5	1	5	18	22

activities on the path reach their minimum durations, which leads to search failure. On the other hand, because the small sloped activities tend to be shrunk until they reach their minimum durations, large sloped activities can remain relatively longer in the final schedule. So the solution quality from slope-based heuristics is better than that from ratio-based heuristics when in fact the slope-based heuristics succeed.

In contrast to slope-based heuristics, ratio-based heuristics avoid the creation of such congested paths. Because an activity’s dynamic duration information is factored into activity selection and long activities are preferred, the search is guided away from a path if it is congested and instead favors activities on less congested paths. Therefore, the activities at minimum duration are balanced on different paths from the first activity to the last activity. Hence, ratio-based heuristics do quite well at avoiding infeasibility.

Exact Lookahead or Not?

We have seen the general advantage of ratio-based heuristics over slope-based heuristics. Now we consider how ratio-based heuristics perform when lookahead with different quality loss computation methods is incorporated. Fig.6 compares those heuristics on solution quality and number of solved problems.

From the results in Fig.6, we can make the following observation.

Observation 2 *Local estimation of quality loss— as implemented in “Ratio-loss” heuristics, actually achieves comparable quality to accurate computation (“Ratio-Exact”), while solving more problems and saving significant computational time. “Ratio-Loss” dominates “Ratio-LossGain” in solution quality and number of solved problems.*

We explain this observation using a problem instance. Table 4 compares the precedence relations in two resulting

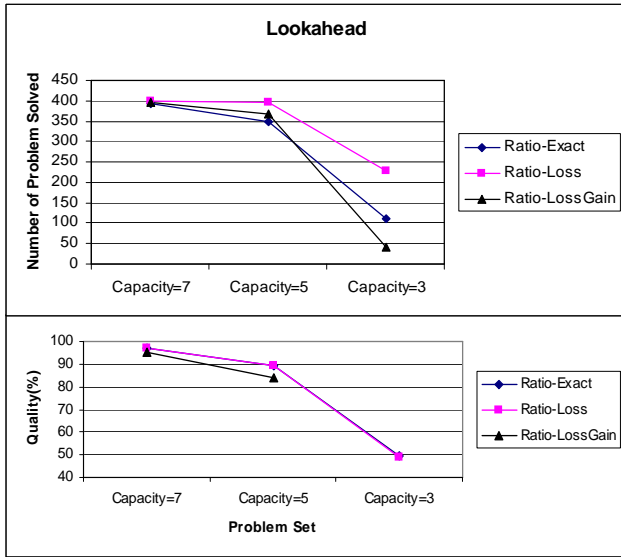


Figure 6: Different Quality Loss Computation Methods

schedules from “Ratio-Exact” and “Ratio-Loss” on the same problem instance, when resource capacity is 3. We ignore the activities which have the same set of successors in those two schedules and focus on the difference.

From Table 4, we can see generally that, in the schedule produced by “Ratio-Exact”, an activity has more successors than it does in the schedule produced by “Ratio-Loss”. Additionally, some activities have very large numbers of successors; for example, activity 11 has 15 successors. The reason is that “Ratio-Exact” tends to sequence two activities with minimum conflict (i.e., minimum overlapping) in order to achieve minimum quality loss. The first activity is chosen with largest ratio, and usually it is relatively long. It will shrink only a very short amount of time; otherwise, the posted constraint is not the choice achieving minimum quality loss. Thus, this first activity will likely remain the largest ratio activity and be selected again in the next cycle if it is still in some peak. Due to tight capacity, it is very often the case that an activity will remain in some peak after a posting step. So the search process has the following character: the first activity is always the same, and precedence constraints are posted between this activity and many other activities. We call it “large successor set effect”. The search process leads to a schedule which is not flexible, because in this “large successor set”, most of activities are at their minimum durations. For example, among the 15 successors of activity 11 in Table 4, 12 are at their minimum durations. Activity 11 starts at time 8 and its duration is 3, which is also its minimum duration. That means the 12 activities with no flexibility for shrinking must be packed between time 12(= 3 + 8 + 1) and time 30 (the project deadline) and satisfy resource capacity constraints. This is a situation very close to temporal infeasibility. And this tendency is why

Table 4: Precedence Relations in a Resulting Schedule

Activity ID	Successors in “Exact”	Successors in “Loss”
3	5 7 10 21	4 5 7 21
4	6 7 11 12 13 14 17 22 24 27	6 12 14 17 18 20 22
5	6 11 12 24 27 29	11 12 14 24 27 29
6	9 12 18 19 20 24	9 16
7	6 12 14 24 27	4 12 14 24 27
8	5 7 13 27	13 27
9	17 19 25	25
10	6 11 12 24 27	4 7 12 24 27
11	9 12 14 15 16 17 18 19 20 21 22 24 26 27 30	6 9 16 17 18 22 26 30
12	9 18 20	11 18 20
14	20 26 29	11 20 26 29
15	9 12 16 18 19 21 30	6 9 12 16 19 20 30
16	17 18 26 28 30	17 22
17	22 26 29 30 31	22 31
19	21 25	9 21
20	23 25	16 23
21	22 25 26 30	17 22 23
23	28 30	28
27	9 12 14 16 18 20 21 22 24 28	6 9 11 12 14 16 19 24 28 29

“Ratio-Exact” solves fewer problems than “Ratio-Loss”.

In Table 4, “Ratio-Loss” has a similar pattern of number of successors with “Ratio-Exact” and there are quite a few common constraints in two schedules. This is the intuition of why “Ratio-Loss” achieves comparable quality with “Ratio-Exact”.

Let us now consider why “Ratio-Loss” is better than “Ratio-LossGain”. As we said in the previous section, the computation of quality gain is overly optimistic because activities can be constrained by their other successors or predecessors, such that they may not be stretchable into the space created by posting a sequencing constraint. From the experiment results, we observe that in our computation of “LossGain”, “Gain” is a dominant factor. Without enough consideration on quality loss, therefore, the solution quality becomes worse. The overestimated “Gain” is actually an indicator of how “connected” this activity is with other activities: the slope information of its successors or predecessors. “Ratio-LossGain” tends to sequence activities with lots of successors or predecessors, which increase the severe level of “large successor set effect” as we said in “Ratio-Exact”. Therefore, “Ratio-LossGain” can’t solve many problems. We also observe that “Ratio-LossGain” tends to post fewer constraints overall than either “Ratio-Loss” or “Ratio-Exact”. We conjecture that since there is less emphasis on choosing activities that minimize loss, there is a tendency by “Ratio-LossGain” to post constraints that lead to greater reduction in peak length.

Impact of Longest Peak First(LPF)

Finally, to see the impact of restricting attention to the current longest peak, let us compare “Global-”(without LPF) and “LPF-” heuristics based on the results in Table 2. we observe the following:

Observation 3 “LPF” does not significantly improve quality, but does improve the solvability greatly. This improvement is particularly evident with respect to “Global-slope” and “LPF-slope”.

The intuition behind this observation is the quality/flexibility tradeoff. Slope-based heuristics statically select smallest sloped activities to sequence, leading to a schedule with relatively high quality but one that is also very fragile for future shrinking. “LPF”, which selects activities in the longest peak, is a mitigation of this static strategy by taking actual duration information into account, and thus leads to greater flexibility for future shrinking. In the case of ratio-based heuristics, “LPF” focusing adds additional bias to flexibility. Although ratio-based heuristics are already quite robust with respect to solving ability, this additional bias does increase solvability slightly in very tightly constrained problems.

Summary

Compared to slope-based heuristics, “Ratio-Loss” achieves comparable quality and solves much more problems. Compared to the other ratio-based heuristics, “Ratio-Loss” solves comparable numbers of problems and achieves the best quality. Compared to “Ratio-Exact”, “Ratio-Loss” achieves comparable solution quality, solves more problems and saves large amount of computation time. Therefore, “Ratio-Loss” is found to be the most effective heuristic tested.

Conclusion and Extensions

In this paper, we have investigated a new type of scheduling problem where the duration is not an inherent property of tasks, but instead a decision to be made by the scheduler. The longer a task is allowed to execute, the higher the quality of its result is, and the goal is to maximize the cumulative quality of all tasks. This additional degree of freedom for decision-making in precedence and resource constrained scheduling environments introduces a new challenge for constraint-based scheduling research.

This paper makes several contributions. First, we presented a new precedence constraint posting algorithm for solving this problem, along with several search control heuristics for resolving resource conflicts. Our solution procedure incorporates a linear program as a sub-procedure for optimally setting activity durations at each step of the search for a resource feasible schedule. The experimental results confirm the overall effectiveness of our approach.

A second contribution is in understanding the key considerations that underlie the design of effective search control heuristics for solving this class of problems. Our experimental analysis indicates two principal factors: the quality loss that results from each constraint posting step, and the flexibility that is retained for future duration reduction. This comes from the explanation of why many of the heuristics

tested fall short either in producing high quality schedules or in producing feasible solutions on a consistent basis. To successfully find a schedule with high quality, a good trade-off must be made between minimizing the quality loss as each new constraint is posted, and retaining duration flexibility to accommodate additional precedence constraints on subsequent cycles.

Third, the paper shows that use of a local estimate of quality loss actually outperforms use of the corresponding exact computation, at greatly reduced computational expense. This is due to the fact that use of the exact computation tends to lead to construction of activity networks with “large successor set” structure, which reduces overall flexibility.

There are several directions for future research.

- *More complex quality dependencies.* In our current model, the only dependencies among activities are precedence relationships. Quality dependencies, which imply that one activity’s output may influence it’s successors’ quality profiles, are more realistic. In that case, instead of a simple summation form, the objective will be a more complex function.
- *Activity Profiles.* We have assumed in this paper that the expected quality of activities can be expressed by linear profiles. Actually, with little change, our solution procedure can be applied to piece-wise linear profiles, although the development of good heuristics for the extended problem remains an important open question. Profile information can also be associated with resources. This latter extension allows for the possibility of differentiating the skill level of different resources.

Acknowledgements. This research was supported in part by the National Science Foundation under contract #9900298 and by the CMU Robotics Institute. The authors thank Anthony Gallagher, Larry Kramer, Christian Ohler, Nicola Policella and Terry Zimmerman for helpful comments on an earlier draft.

References

- Ajili, F., and El Sakkout, H. 2003. A probe-based algorithm for piecewise linear optimization in scheduling. *Annals of Operations Research* 118:35–48.
- Baptiste, P.; Le Pape, C.; and Nuijten, W. 2001. *Constraint-based Scheduling: Applying Constraint Programming to Scheduling Problems*. Norwell, MA: Kluwer Academic Publishers.
- Beck, J. 1999. *Texture Measurements as a Basis for Heuristic Commitment Techniques in Constraint-Directed Scheduling*. Ph.D. Dissertation, Dept. of Computer Science, University of Toronto.
- Beringer, H., and De Backer, B. 1995. *Logic Programming: Formal Methods and Practical Applications*. Amsterdam, Holland: Elsevier Science. chapter Combinatorial Problem Solving in Constraint Logic Programming with Cooperating Solvers.
- Cesta, A.; Oddi, A.; and Smith, S. 2000. Iterative flattening: A scalable method for solving multi-capacity schedul-

- ing problems. In *Proceedings of the Seventeenth National Conference in Artificial Intelligence (AAAI'00)*.
- Cesta, A.; Oddi, A.; and Smith, S. 2002. A constraint-based method for project scheduling with time windows. *Journal of Heuristics* 8(1):109–136.
- Cheng, C., and Smith, S. 1996. A constraint satisfaction approach to makespan scheduling. In *Proceedings 4th International Conference on Artificial Intelligence Planning Systems*.
- Dean, T. L., and Boddy, M. 1988. An analysis of time-dependent planning. In *Proceedings of 7th National Conference on Artificial Intelligence*, 49–54. St. Paul, MN: AAAI Press.
- El Sakkout, H., and Wallace, M. 2000. Backtrack search for minimal perturbation in dynamic scheduling. *Constraints, Special Issue on Industrial Constraint-Directed Scheduling* 5(4).
- El Sakkout, H.; Wallace, M.; and Richards, T. 1998. Minimal perturbation in dynamic scheduling. In *Proc. of the 13th European Conference on Artificial Intelligence (ECAI-98)*.
- Fulkerson, D. 1961. A network flow computation for project cost curves. *Management Science* 7(2):167–178.
- Hooker, J. N. 2004. A hybrid method for planning and scheduling. In *Tenth International Conference on Principles and Practice of Constraint Programming (CP 2004)*.
- Horvitz, E. 1987. Reasoning about beliefs and actions under computational resource constraints. In *Third Workshop on Uncertainty in Artificial Intelligence*.
- Icmeli, O., and Erenguc, S. 1996. The resource constrained time-cost tradeoff project scheduling problem with discounted cash flows. *Journal of Operations Management* 14:255–275.
- Kelley Jr., J., and Walker, M. 1959. *Critical Path Planning and Scheduling: An introduction*. Ambler, Pa: Mauchly Associates Inc.
- Kelley Jr., J. 1961. Critical path planning and scheduling: Mathematical basis. *Operations Research* 9(3):296–320.
- Khatib, L.; Morris, P. H.; Morris, R. A.; and Rossi, F. 2001. Temporal constraint reasoning with preference. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*.
- Nuijten, W., and Aarts, E. 1994. Constraint satisfaction for multiple capacitated job-shop scheduling. In *Proceedings of the 11th European Conference on Artificial Intelligence*.
- Oddi, A., and Cesta, A. 1997. A tabu search strategy to solve scheduling problems with deadlines and complex metric constraints. In *Proceedings of the 4th European Conference on Planning*.
- Philips, S., and Dessouky, M. 1977. Solving the project time-cost tradeoff problem using the minimal cut concept. *Management Science* 24(4):393–400.
- Policella, N.; Smith, S. F.; Cesta, A.; and Oddi, A. 2004. Generating robust schedules through temporal flexibility. In *Proceedings 14th International Conference on Automated Planning and Scheduling*.
- Russel, S., and Wefald, E. 1991. *Do the Right Thing: Studies in Limited Rationality*. Cambridge, MA: MIT Press.
- Sadeh, N. 1991. *Look-Ahead techniques for Micro-Opportunistic Job Shop Scheduling*. Ph.D. Dissertation, Dept. of Computer Science, Carnegie Mellon University.
- Schwarzfischer, T. 2003a. Application of simulated annealing to anytime scheduling problems with additional timing constraints. In *Proceedings of the Fifth Metaheuristics International Conference*.
- Schwarzfischer, T. 2003b. Quality and utility-towards a generalization of deadline and anytime scheduling. In *Proceedings of the 13th International Conference on Automated Planning and Scheduling*.
- Schwarzfischer, T. 2003c. Using value dependencies to schedule complex soft-real-time applications with precedence constraints. In *Proceedings of the 1st Multidisciplinary International Conference on Scheduling: Theory and Applications*.
- Slowinski, R. 1980. Two approaches to problems of resource allocation among project activities—a comparative study. *J.Operational Research Society* 31(8):711–723.
- Slowinski, R. 1981. Multiobjective network scheduling with efficient use of renewable and non-renewable resources. *European J.Operational Research* 7(3):711–723.
- Talbot, F. 1982. Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case. *Management Science* 28(10):1197–1210.
- Wang, X., and Smith, S. 2004. A precedence-constraint-posting procedure to generate schedules maximizing quality with anytime property. Technical Report TR-04-25, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.
- Zilberstein, S. 1993. *Operational rationality through compilation of anytime algorithms*. Ph.D. Dissertation, Computer Science Division, University of California at Berkeley.