Exploiting Temporal Flexibility to Obtain High Quality Schedules

Nicola Policella
ISTC-CNR
Rome, Italy
nicola.policella@istc.cnr.it

Xiaofang Wang
The Robotics Institute, CMU
Pittsburgh, PA, USA
xiaofanw@cs.cmu.edu

Stephen F. Smith
The Robotics Institute, CMU
Pittsburgh, PA, USA
sfs@cs.cmu.edu

Angelo Oddi
ISTC-CNR
Rome, Italy
angelo.oddi@istc.cnr.it

Abstract

We consider a schedule optimization problem where each activity to be scheduled has a duration-dependent quality profile, and activity durations must be determined that maximize overall quality within given deadline and resource constraints. To solve this quality maximization problem, prior work has proposed a hybrid search scheme, where a linear programming solver for optimally setting the durations of temporally related activities is embedded within a larger search procedure that incrementally posts sequencing constraints to resolve resource conflicts. Under this approach, dual concerns of establishing feasibility and optimizing quality are addressed in an integrated fashion. In this paper, we propose an alternative approach, where feasibility and optimization concerns are treated separately: first, we establish a resource-feasible partial order schedule, assuming minimum durations for all activities; second, these fixed duration constraints are relaxed and quality optimal durations are determined. Experimental results indicate a tradeoff: when resource capacity constraints are loose, the integrated hybrid approach performs comparably to the separated scheme. However, in problems with tighter capacity constraints we find that separation of concerns enables both better solving capability and higher quality results. Following from these results, we discuss potential synergy between problem objectives of maintaining temporal flexibility and maximizing quality.

Introduction

(Wang & Smith 2005) brings attention to a class of quality maximization scheduling problems that is of significant importance in the real world, but has received little attention in the research community. In many domains, the quality obtained by performing a given activity will depend on how long it is executed, and the overall performance of the enterprise depends on how effectively time and resources are apportioned to various activities. News reporting, intelligence gathering, and new product research and development are representative examples, where resources are finite and results must be obtained within specified time frames. When viewed abstractly, such quality maximization scheduling problems are distinguished by the fact that activity duration is an additional free variable. In attempting to satisfy tight resource constraints and meet deadlines, an activity can be scheduled for a shorter period than might be most desirable, with a corresponding degradation in quality. Quality maximization scheduling problems are closely related to a number of previously studied problems, including the time/cost tradeoff problem (Kelley Jr. & Walker 1959), anytime scheduling (Dean & Boddy 1988; Zilberstein 1993), extended deadline scheduling (Schwarzfischer 2003) and temporal preference networks (Khatib *et al.* 2001). But quality maximization problems are different due to the additional presence of cumulative resource capacity constraints.

In (Wang & Smith 2005), a resource-constrained project scheduling version of the quality maximization problem is considered. A linear quality profile is associated with each activity in the project network, which specifies the quality of the associated activity's output as an increasing function of time. Taking overall quality to be the sum of the output quality of each individual activity (according to their scheduled durations), the objective is to construct a schedule maximizes overall quality within given deadline and resource constraints. They developed a hybrid solution procedure which combines two components: (1) a linear programming (LP) solver for optimally setting the activity durations of a set of temporally related activities, and (2) a precedence constraint posting (PCP) search procedure for resolving resource conflicts and establishing resource feasibility. Within this approach, the dual concerns of resource feasibility and quality optimization are considered in a tightly integrated fashion. The LP solver is embedded within the PCP procedure and the impact of various constraint posting decisions on solution quality is used to direct the search process. Wang and Smith's analysis centered on the design of search control heuristics, and they found it difficult to design a heuristic that effectively balances the quality loss incurred in each constraint posting step against the need to retain flexibility for achieving resource feasibility.

In this paper, we propose an alternative approach to solving this problem, motivated by recent work in the development of procedures for generating temporally flexible schedules (Cesta, Oddi, & Smith 1998; 2000; Policella *et al.* 2004b). Specifically, we use these results to first generate a resource-feasible, temporally flexible schedule, assuming minimum durations for all project activities. Then, in a second step, we relax these duration constraints and use an LP solver as before to optimally "stretch" project activities. We empirically contrast the performance of this new *separated* algorithm with the previously developed *integrated* algorithm. The new search algorithm is found to dominate on problems with tighter resource capacity constraints.

We begin by specifying the quality maximization problem of interest more precisely and describing the two alternative solution approaches.

Problem Formulation

Given a project composed of a set of non-preemptive activities $V = \{a_1, \ldots, a_n\}$, a set of precedence constraints, a set of quality profiles and resources with limited capacity, the problem is to determine the start time s_i and the end time e_i of each activity a_i so as to maximize the sum of qualities of all the activities. More precisely, assume the following definitions:

- r_i: release date for activity a_i,
- D: a common deadline for all the activities, i.e., the overall project deadline,
- q_i: output quality from activity a_i; q_i is a non-decreasing linear and continuous function of a_i's duration with slope k_i, i.e., $q_i = k_i \cdot (e_i - s_i)$,
- d_i: minimum duration[1] for activity a_i,
- E: the set of edges in the precedence graph, if $(i, j) \in E$, activity a_j must start after activity a_i is completed,
- C: resource capacity available over the entire horizon; without loss of generality, each activity a_i is assumed to require one unit of resource.

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. Then the quality maximization problem can be formulated as follows:

maximize

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

subject to

$$e_i \leq s_j, (i,j) \in E, \tag{2}$$

$$R(S,t) \leq C, \tag{3}$$

$$d_i \leq e_i - s_i, i \in V, \tag{4}$$

$$r_i \leq s_i, i \in V, \tag{5}$$

$$e_i \leq D, i \in V, \tag{6}$$

(2), (3), (4), (5) and (6) are precedence, resource, minimum duration, release date and due date constraints respectively. The multiple capacity ($C > 1$) version of this problem has been shown to be *NP*-complete (Wang & Smith 2004).

[1]This 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.

Integrated Approach

The integrated approach to the quality maximization problem previously developed in (Wang & Smith 2005) was motivated by the observation that the un-capacitated version of the problem in a project network is essentially the time/cost tradeoff problem (Kelley Jr. & Walker 1959), and hence the linear programming techniques used widely to solve this problem could be exploited for use in solving the resource constrained variant. In the subsections below, we summarize their solution procedure and the search heuristics found to be most effective.

The Precedence Constraint Posting Framework

Generally speaking, a Precedence Constraint Posting (PCP) scheduling procedure iteratively transforms a time feasible solution into a resource feasible solution by eliminating all contention peaks. A *contention peak* is a set of activities which simultaneously requires resources in excess of the resource capacity C over a 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. A contention peak is eliminated (or levelled) by posting one or more sequencing constraints between pairs of competing activities in the peak. Each time a new constraint is posted, constraint propagation is performed to update activity start and end times, and to confirm that the solution is still feasible.

In the case of Wang and Smith's procedure, the above constraint propagation step was replaced by a call to the LP solver. Thus, new "quality optimal" start times and durations are recomputed each time a new sequencing constraint is posted, and, once all contention peaks are eliminated, the final solution is returned. If the LP solver fails after a constraint has been posted, then the project network has become over-constrained and is no longer temporally consistent. The basic schedule generation framework is shown in Fig. 1. The algorithm is described in Fig. 2.

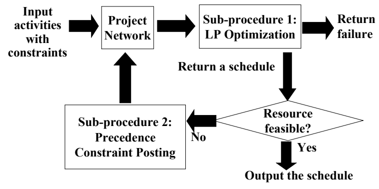

Input activities with constraints
Project Network
Sub-procedure 1: LP Optimization
Return failure
Return a schedule
Resource feasible?
No
Yes
Sub-procedure 2: Precedence Constraint Posting
Output the schedule

Figure 1: The Precedence Constraint Posting Framework

```
Quality-Maximizing 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: Quality-Maximizing Constraint Posting Algorithm

Constraint-posting Heuristics

Given the greedy nature of the above algorithm, the heuristics used to select which pair of activities to sequence after each call to the LP solver play an important role in the algorithm's performance. In (Wang & Smith 2005) several heuristics were tested and one named "Ratio-Loss" proved to be most effective. Hence we adopt this heuristic in the experiments reported below.

The "Ratio-Loss" heuristic uses different criteria to choose the two conflicting activities to sequence at each step. Given the set of activities currently in some contention peak, the first activity chosen is the activity in this set with the largest **ratio** of "reducible duration" to slope. The second activity chosen is the activity in the same peak that leads to the minimum estimated **quality loss**, and the pair is sequenced in the order that achieves this minimum quality loss.

"Ratio" is defined as:

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

where the reducible duration of an activity is the amount greater than its minimum duration. The larger the reducible duration, the greater the flexibility for future shrinking. The smaller the slope, the smaller the possible quality loss if we shrink this activity. As concluded in (Wang & Smith 2005), bias toward the choice of activities with small slope and long reducible duration strikes a good balance between the two goals in the search: maximizing quality and retaining flexibility.

"Quality loss" is a local estimation of the difference between the quality optimal values that would be obtained by running the LP solver before and after posting the candidate precedence constraint. Thus, the heuristic can be considered as a one step lookahead greedy search for a high quality solution. Estimation is used instead of an exact LP solver computation due to computational cost.

Quality loss is estimated by locally calculating the duration changes of the pair of selected activities, say a and b, with 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. Let A_{loss} be the quality loss due to a's duration change, and B_{Loss} be the quality loss due to b's duration change. Then

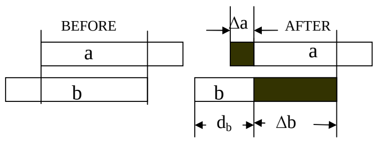


Figure 3: An Example for Quality Loss Estimation

$$QualityLoss = A_{loss} + B_{Loss} \tag{7}$$

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

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

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

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

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

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

An example used in (Wang & Smith 2005) to illustrate this method is reproduced in Fig. 3. Because $s_b - s_a < e_a - e_b$, according to (10) and (11), the quality loss can be estimated as follows:

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

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

The minimum estimated quality loss solution from observation 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.

Solve-and-Maximize

A different way to approach the resource constrained quality maximization problem is to separately address the two principal difficulties it presents: first find a feasible solution and then achieve good quality. To maximize chances of obtaining a resource feasible solution in the first phase, we simplify the problem. First we reduce the degrees of freedom in the problem and assume that all activities will be executed for their minimum duration. Further, to provide greater leverage to the PCP process in leveling conflicts, we restrict attention to the earliest start time solution and concentrate on generating a fixed-times schedule.

Once a resource feasible solution is obtained, we must adjust our assumptions to achieve a good quality solution in the second phase. By relaxing the constraint that all activities must execute for their minimum duration only, we can provide the opportunities to expand activity durations. However, since all analysis during schedule generation was focused on the early start time solution, there is no guarantee that we will not reintroduce conflicts as we begin to increase activity durations. We are still hindered by the inflexibility of the fixed-times schedule we have generated.

To cope with this problem, the fully instantiated solution is transformed into a Partial Order Schedule, or $\mathcal{POS}$ (Policella *et al.* 2004b). A $\mathcal{POS}$ is a graph $G(V, E)$ where the

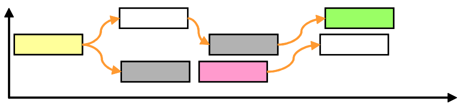
```
**Chaining**$(P, S)$
**Input**: A problem $P$ and one of its fixed-times schedules $S$
**Output**: A partial order schedule $\mathcal{POS}$
1. $\mathcal{POS} \leftarrow P$
2. Sort activities according to their start times in $S$
3. Initialize all chains empty
4. **for each** activity $a_i$
5.         $k \leftarrow SelectChain(a_i)$
6.         $a_k \leftarrow last(k)$
7.         **if** $\neg\exists(a_k \prec a_i)$
8.             $AddConstraint(\mathcal{POS}, a_k \prec a_i)$
9.         $last(k) \leftarrow a_i$
10. **return** $POS^{ch}$
```

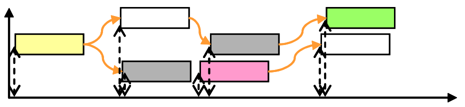
Figure 4: Basic Chaining procedure.

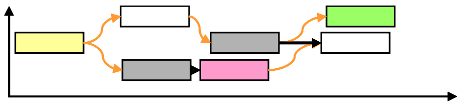
nodes V are activities and the edges E are temporal constraints between pairs of activities such that any temporal solution to this graph is also a resource feasible solution. We obtain a Partial Order Schedule by applying a post-process procedure called *chaining*. This procedure feeds temporal flexibility back into the fixed-times solution. In a chaining-form representation of the schedule, activities which require the same resource unit are linked via precedence constraints into precedence chains. Given this structure, each constraint becomes more than just a simple precedence. It also represents a producer-consumer relation, allowing each activity to know which other activity will supply the unit of resource it requires for execution. It is clear that this representation provides temporal flexibility, allowing chained activities to float "back and forth" and/or to execute longer than their minimum duration.

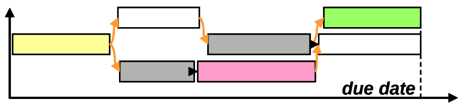
The chaining procedure is summarized by the algorithm in Fig. 4. The first step sorts all activities according to their start times in the schedule S. Then activities are incrementally allocated to the different chains. The allocation of an activity a_i to a chain k amounts to adding a precedence constraint (if not already present) between the current last element of the chain, $a_k \leftarrow last(k)$, and a_i, as well as, in updating the last element of the chain $last(k) \leftarrow a_i$. The function $SelectChain(a_i)$ is the core of the procedure; it can admit different definitions giving different results. A basic implementation chooses, for each activity, the first available chain of r_j. Given an activity a_i, a chain k is *available* if the end time of the last activity allocated on it, a_k, is not greater than the start time of a_i (in the algorithm $a_k \prec a_i$ means that a_k preceeds a_i, that is, the end-time of a_k is not greater than the start-time of a_i). Note that since the input to a chaining procedure is a consistent solution it will always be possible to find the chains that a given activity a_i needs.

Once a Partial Order Schedule is obtained, it is possible to expand activity durations without danger of introducing resource conflict. In fact, the structural properties of a $\mathcal{POS}$ prevent any two activities using same resource unit from overlapping. Of course, duration decisions must still respect the temporal constraints defined by the partial order schedule (i.e., precedence constraints, release date, bounds over the activity durations, common deadline). But this can be accomplished in the same manner as before, by applying the same LP optimization to obtain durations of activities that optimize the overall quality of the $\mathcal{POS}$.


(a) Project Network (with minimum duration).


(b) Fixed-times solution


(c) Partial Order Schedule



(d) Solution (maximum reward)

Figure 5: The Solve-and-Maximize steps.

```
**Solve-and-Maximize**
**Input**: A problem $P$
**Output**: A solution $s$
1. $P' \leftarrow P \cup_{i=1}^{n} \{e_i - s_i = d_i\}$
2. $s' \leftarrow$find-a-schedule$(P')$
3. $\mathcal{POS} \leftarrow$ Chaining$(P, s')$
4. $s \leftarrow$LPsolver $(\mathcal{POS})$
5. **return** $s$
```

Figure 6: Solve-and-Maximize.

Fig. 5 illustrates the different stages of the Solve-and-Maximize solution procedure[2]. First an initial fixed-times solution to the problem variant with activity durations equal to their lower bound (step 1 in the algorithm) is computed[3] (Fig. 5(b) and step 2). This solution is used as the seed schedule for the chaining procedure (step 3). In this step, the chaining procedure considers the original problem P (instead of P') in which activity durations are not fixed (s' is a solution for both P and P'). Eventually, exploiting the temporal flexibility of a $\mathcal{POS}$, the activity durations are stretched out to increase the final solution reward by a LP solver (step 4). The Solve-and Maximize algorithm is given in Fig. 6.

In the following empirical evaluation section we present results obtained by one possible implementation of the Solve-and-Maximize scheme. Specifically, we use the more sophisticated chaining procedure introduced in (Policella *et al.* 2004a). In this work the authors have used an iterative sampling method to increase the flexibility of the final $\mathcal{POS}$. In fact, since many $\mathcal{POS}$s are obtainable from the same fixed-times schedule, a more sophisticated approach can produce more flexible $\mathcal{POS}$s. In practice the method is obtained by iterating the steps 3-4 of Fig. 6 and returning the

[2]In Fig. 5, the gray and black arrows represent the constraints defined in the problem and those posted during the solving process respectively.

[3]In the implemented method we have used the ESTA algorithm (Cesta, Oddi, & Smith 1998) as solver.

	due date = 25			due date = 30			due date = 35		
	solved	**npc**	**quality**	**solved**	**npc**	**quality**	**solved**	**npc**	**quality**
Integrated Approach									
C=3	0.8%	-	-	57.0%	62.03	48.51%	95%	64.44	48.67%
C=5	88.8%	23.66	80.75%	98.8%	24.67	79.91%	100%	24.91	79.68%
C=7	99.2%	6.11	95.45%	100%	6.07	95.33%	100%	6.09	95.37%
Solve-and-Maximize									
C=3	31.2%	14.30	47.21%	100%	13.94	50.87%	100%	13.93	52.01%
C=5	100%	6.33	81.60%	100%	6.27	81.37%	100%	6.32	81.19%
C=7	100%	2.51	95.37%	100%	2.52	95.28%	100%	2.54	95.19%

Table 1: Experimental results.

highest quality solution found. More precisely, we use an iterative sampling variant guided by a heuristic which aims at minimizing the interdependencies between pairs of chains[4]. This heuristic further enhances the temporal flexibility of the final chaining-form solution.

Experimental Evaluation

In this section we evaluate the relative performance of the two algorithms just described. Same as in prior work (Wang & Smith 2005), we generated data sets using precedence constraints defined in a single mode resource constrained project scheduling benchmark problem set[5]. We compare the performance of the two algorithms using following measures:

1. *solved (%)*. The percentage of problems solved.
2. *npc*. The average number of posted constraints in the generated solution.
3. *quality (%)*. The average percentage quality normalized to the infinite capacity solution. The infinite capacity solution gives us an upper bound for the capacitated problem.

Regarding the Solve-and-Maximize procedure, the iterated phase is repeated for 100 times. CPU times are not presented here for the sake of space, but we note that the average solving time on the problems tested for both approaches is on the order of a few seconds.

The Hardness of Different Problem Sets

For the same temporal network (i.e. the same set of precedence constraints), a problem can be hardened along two different dimensions: by moving the common deadline earlier or by lowering the resource capacity. To give a comprehensive comparison of our algorithms, we have tested the approaches on the problems at different levels of hardness. In particular, nine data sets of 400 problems each were generated with the combination of three resource capacity values (3, 5, and 7) and three different due dates (25, 30, and 35) respectively. Each problem has 32 activities, and 32 uniformly distributed random integers in the range $[1, 50]$ represent the slopes for each problem. The release date of each activity is an integer uniformly distributed in the range $[0, 5]$. Last, a uniformly distributed random integer in the range $[1, 3]$ represents the minimum duration of an activity.

The overall performance results[6] are given in Table 1. There are three points worth mentioning here. First, for the $duedate = 25$ problems, 25 of the 400 problems are known to be infeasible. To provide a fair comparison, *solved* is the percentage of problems solved out of the feasible problems. Second, in cases where one algorithm cannot solve all problems for a given capacity and deadline setting, *quality* and *constraints* are calculated based on the commonly solved problems. Third, there are some blanks in Table 1; these are due to the fact that the integrated approach solves much fewer problems than the other approach, and such a small set of commonly solved problems does not provide the representative information we need for comparison.

Integrated vs. Separated

From an analysis of Table 1 several observations can be made. A first observation regards the number of *solved* problems: when resource capacity constraints are loose, the integrated approach performs comparably to the separated scheme under different deadline settings. However, in problems with tighter capacity constraints we find that the separated approach performs much better.

With regard to *quality*, the results are more surprising. Even though the integrated scheme takes into account the quality goal at each step of the solving process, it turns out to produce lower-quality schedules on most of the data sets, and, in particular, on the hardest ones. This behavior is confirmed by *npc*, where we can see that the separated scheme posts fewer constraints than the integrated approach. These results indicate that there is a synergy between the objective of maintaining temporal flexibility and that of maximizing quality. The separated scheme builds up a more flexible schedule at the first stage. And this flexibility helps to produce better high-quality schedules in the second stage. By attempting to consider quality maximization along the way to generating a resource feasible solution, the integrated approach lessens its opportunities to retain temporal flexibility.

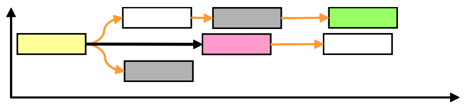
A Further Analysis on Temporal Flexibility

To further confirm our hypothesis about the utility of maintaining temporal flexibility from the standpoint of maximizing quality, we replace the iterative chaining procedure used

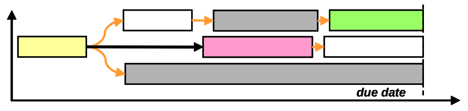
[4]In (Policella *et al.* 2004a) it is named H^2.

[5]http://www.bwl.uni-kiel.de/Prod/psplib/datasm.html. Filename: $j30.sm.tgz$.

[6]Regarding the implementation, two different libraries have been used as LP solver: LINGO 8.0 in case of the Integrated Search and Ilog CPLEX 8.0 in the Separated approach.



(a) Alternative $\mathcal{POS}$ w.r.t. Fig. 5(c)



(b) Better solution w.r.t. Fig. 5(d)

Figure 7: The effects of temporal flexibility on the quality solution.

in our previous experiments with a simpler version presented in Fig. 4. In this simpler chaining procedure a $\mathcal{POS}$ is generated using a trivial heuristic, without consideration of the flexibility of the result. Table 2 shows the results obtained using this chaining version. We can see that the solution quality degrades greatly compared to the original algorithm. In particular in the case of $C = 7$, the quality values are reduced from approximately 95% to approximately 70%.

We illustrate the importance of temporal flexibility in the separated scheme approach by a simple example shown in Fig. 7. Fig 7(a) presents a different $\mathcal{POS}$ for the same fixed times schedule in Fig. 5(b). It has greater flexibility than the $\mathcal{POS}$ in Fig. 5(c) (as there are fewer constraints between pairs of chains) and thus is capable of producing a solution of better quality (compare Fig. 5(d) and Fig. 7(b)).

quality	due date = 25	due date = 30	due date = 35
C=3	46.39%	49.03%	49.88%
C=5	67.53%	67.67%	68.26%
C=7	70.03%	70.57%	71.55%

Table 2: Quality values using the simple chaining method.

Conclusions and Future Work

In this paper we described a new strategy for solving a scheduling problem where activities have a duration-dependent linear quality profiles, and activity durations must be determined that maximize overall quality within given deadline and resource constraints. Our main contribution is an optimization procedure where resource feasibility and quality maximization concerns are treated separately. At the core of our solving procedure is the simple and effective idea that temporally flexible solutions provide better *grist* for generating higher quality schedules. After generating a resource feasible schedule, we transform it into a temporally flexible partial order schedule, which is then provided as input to an optimizing LP solver.

We compare the proposed optimization procedure with a previously developed procedure that takes an integrated approach to solving the problem, attempting to minimize quality loss in the course of generating a resource feasible solution. Experimental results confirm that while both approaches perform comparably on easier problem sets, the separated approach leads to better solving ability and higher quality final solutions on problem sets with tighter resource capacity constraints. It is clear from these results that an ability to retain greater temporal flexibility in the schedule promotes better quality final solutions.

We are pursuing future work in several directions. First, we are working on a new tighter upper-bound of the objective function which takes resource capacity into account. Next, we would like to extend our optimization procedure to more complex variations of this scheduling problem (e.g., piece-wise linear profiles, multiple resource skill levels). Finally, we are interested in the possibility of some sort of middle ground solution technique, drawing on the qualities of both algorithms considered in this paper.

Acknowledgements. Wang and Smith's research has been funded in part by the National Science Foundation under contract # 9900298 and the CMU Robotics Institute. Policella and Oddi's work has been sponsored in part by ASI (Italian Space Agency) under the project SACSO and by the Italian Ministry for Education, University and Research under the project RoboCare.

References

Cesta, A.; Oddi, A.; and Smith, S. F. 1998. Profile Based Algorithms to Solve Multiple Capacitated Metric Scheduling Problems. In *Proceedings of the* 4th *International Conference on Artificial Intelligence Planning Systems, AIPS-98*, 214–223.

Cesta, A.; Oddi, A.; and Smith, S. F. 2000. Iterative Flattening: A Scalable Method for Solving Multi-Capacity Scheduling Problems. In *Proceedings of the* 17th *National Conference in Artificial Intelligence (AAAI'00)*.

Dean, T. L., and Boddy, M. 1988. An analysis of time-dependent planning. In *Proceedings of the 7th National Conference on Artificial Intelligence*, 49–54. St. Paul, MN: AAAI Press.

Kelley Jr., J., and Walker, M. 1959. *Critical Path Planning and Scheduling: An introduction*. Mauchly Associates Inc.

Khatib, L.; Morris, P. H.; Morris, R. A.; and Rossi, F. 2001. Temporal Constraint Reasoning With Preference. In *Proceedings of the* 17th *International Joint Conference on Artificial Intelligence*.

Policella, N.; Oddi, A.; Smith, S. F.; and Cesta, A. 2004a. Generating Robust Partial Order Schedules. In Wallace, M., ed., *Principles and Practice of Constraint Programming,* 10th *International Conference, CP 2004*, volume 3258 of *Lecture Notes in Computer Science*, 496–511. Springer.

Policella, N.; Smith, S. F.; Cesta, A.; and Oddi, A. 2004b. Generating Robust Schedules through Temporal Flexibility. In *Proceedings of the* 14th *International Conference on Automated Planning & Scheduling, ICAPS'04*, 209–218. AAAI.

Schwarzfischer, T. 2003. Quality and Utility - Towards a Generalization of Deadline and Anytime Scheduling. In *Proceedings of the* 13th *International Conference on Automated Planning & Scheduling, ICAPS'03*.

Wang, X., and Smith, S. F. 2004. Generating Schedules to Maximize Quality. Technical Report TR-04-25, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Wang, X., and Smith, S. F. 2005. Retaining Flexibility to Maximize Quality: When the Scheduler Has the Right to Decide Activity Durations. In *Proceedings of the* 15th *International Conference on Automated Planning & Scheduling, ICAPS'05*.

Zilberstein, S. 1993. *Operational rationality through compilation of anytime algorithms*. Ph.D. Dissertation, Computer Science Division, University of California at Berkeley.