

Solving the Engineering Vice-President's Problem

**Michael M. Murray
Construction Engineering and Management
Civil and Environmental Engineering Department
School of Engineering
Stanford University**

October 4, 2004

Acknowledgement of Support and Disclaimer

This material is based upon work supported by the National Science Foundation under Grants No. 9980109 and 9907403. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation

1	Executive Summary	1
2	Introduction.....	2
3	Introduction to the EVPP	2
3.1	Definition	2
3.2	A Real-World Example.....	2
3.2.1	Project Portfolio and Project Completions Drive Revenue	3
3.2.2	Human Resource Investment Drives Cost.....	4
3.2.3	Design Work Process Drives Feasibility	5
3.3	Interactions Make the EVPP Interesting.....	6
3.4	Distinguishing Features of the EVPP	7
4	Points of Departure and Research Questions.....	10
4.1	Models.....	10
4.2	Algorithms	12
4.3	Research Questions.....	13
5	A Simplified EVPP	14
5.1	Formulation.....	14
5.2	Example Problem.....	15
6	Solving the Simplified Formulation	17
6.1	Linear Programming Approximation.....	17
6.2	Embedding the LP Approximation in Discrete Search.....	18
6.3	Heuristics for Discrete Search	19
6.3.1	The Problem.....	19
6.3.2	Value Ordering Heuristic.....	20
6.3.3	Variable Ordering Heuristic.....	23
6.4	Testing the Optimizer	23
6.4.1	Test Case.....	23
6.4.2	Results.....	24
7	Next Steps	26
7.1	Extending the EVPP Formulation.....	26
7.1.1	Portfolio Selection	26
7.1.2	Task Quality.....	26
7.1.3	Fast-Tracked Tasks with Coordination Penalty	26
7.1.4	Preemption	27
7.2	Testing.....	27
7.3	Performance Tuning.....	28
7.4	Future Research	28
8	Conclusion	29
9	References.....	30

1 Executive Summary

The Engineering Vice-President's Problem is the problem of simultaneously managing an engineering organization's

- project portfolio and due dates,
- human resource investment, and
- engineering design work process

to maximize that organization's contribution to corporate profitability. This working paper describes my research on solving the EVPP.

I believe that my research has made four contributions to engineering management science and to optimization science. I have

1. recognized that real world engineering management problems have too much flexibility to be represented and solved using a constraint-based scheduling approach. Real managers get paid to circumvent constraints, not be bound by them.
2. developed a mathematical formulation that allows a natural expression of the interactions, options and constraints in engineering management. This formulation can be solved with a combination of linear programming (LP) and discrete search. The LP models the continuous tradeoffs in the formulation, but it cannot easily represent the discontinuities associated with temporally reordering tasks. The alternatives for temporal reordering are explored by discrete search.
3. shown how to improve the effectiveness of discrete search in this formulation. The binding of a single variable should be made within the context of projected bindings for all of the other variables.
4. written a solver that has solved an academic EVPP test problem efficiently. My solver can produce a good answer to this problem in seconds and better answers given more time.

My future research will be to test the solver on additional academic and industrial test cases and to improve its run time performance.. Other researchers may want to

- extend the formulation to model additional options, constraints, or phenomena in engineering management.
- investigate whether using projected bindings in discrete search is helpful for other search problems.
- intergrate the EVPP solver with Bijan Khosraviani's GP based optimizer for organizations.
- use the EVPP as a tool for developing organization theory.

This working paper describes the above contributions in more detail.

2 Introduction

The Engineering Vice-President's Problem (EVPP) is an interesting and unexplored problem in engineering management. This working paper describes the motivation and formulation of the EVPP and my research towards solving it. I believe that the EVPP can be solved through a novel combination of linear programming and discrete search techniques. Solving the EVPP will contribute primarily to project management science and practice. This research may also make a secondary contribution to optimization science.

Section 3 of this paper motivates the EVPP and describes its distinguishing features. Section 4 reviews past work in project management models and algorithms and presents my research questions. Section 5 presents a mathematical formulation for a simplified version of the EVPP and a simple example. Section 6 describes how the EVPP can be solved and gives results from solving a test case. Section 7 presents the next steps for research involving the EVPP. Section 8 concludes the paper.

3 Introduction to the EVPP

3.1 Definition

For many high technology companies, the technical superiority of their products or services is a cornerstone of corporate strategy. Engineering design is therefore a vital function in these companies. Engineering's importance is recognized organizationally by the titles given to the top engineering executives, titles such as Chief Technology Officer or Vice-President of Engineering. These executives are accountable to the CEO for the performance of the Engineering organization. The Engineering Vice-President's problem is my name for the managerial challenge these executives face.

I define the Engineering Vice-President's Problem as the problem of simultaneously managing an engineering organization's

- project portfolio and project completion dates,
- human resource (HR) investment, and
- engineering design work process

to maximize that organization's contribution to corporate profitability.

I will elaborate this definition in subsequent sections.

3.2 A Real-World Example

Before I entered the Ph.D. program, I was a second level electronic engineering manager at Acuson in Mountain View, CA. Acuson designed, manufactured, and serviced premium quality ultrasound imaging equipment for medical applications. Acuson's flagship system, the Sequoia, is shown in Figure 1 below. Sequoia required approximately \$70 million and several man-centuries of engineering effort to develop. The EVPP is inspired by my observation of engineering design and management during nine years at Acuson.

The following sections describe how Acuson addressed the three elements of the Engineering Vice-President's Problem.

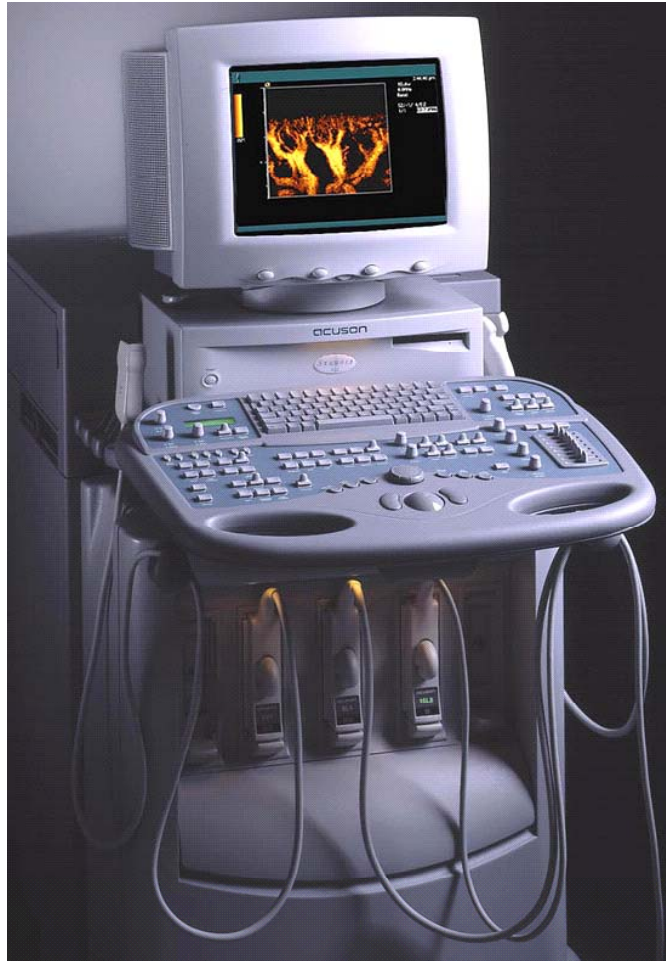


Figure 1. Acuson Sequoia Ultrasound System

3.2.1 Project Portfolio and Project Completions Drive Revenue

Engineering makes a positive contribution to corporate profitability by completing design projects, because completed design projects drive corporate revenue. Typically, completing a design sooner, with higher quality and more features, results in higher cumulative corporate profits generated by the corresponding product or service over its life cycle.

Engineering's positive contribution for a product can be defined as the net present value (NPV) of the product's cumulative revenue minus the NPV of the product's cumulative non-engineering expenses, such as production costs and sales costs. For brevity, Engineering's positive contribution will be referred to as revenue throughout this paper.

Figure 2 shows a plan for development and upgrade releases for each of the ultrasound products in (Pyxis, PACS, etc.) in Acuson's portfolio. Acuson Engineering was responsible for developing these releases. Each diamond symbol indicates a planned release date; the features of the release are listed under the diamond. New releases generated revenue by stimulating sales of new ultrasound systems, upgrade kits, and service contracts. Acuson's engineering managers considered revenue implications as they made project portfolio and release date decisions.

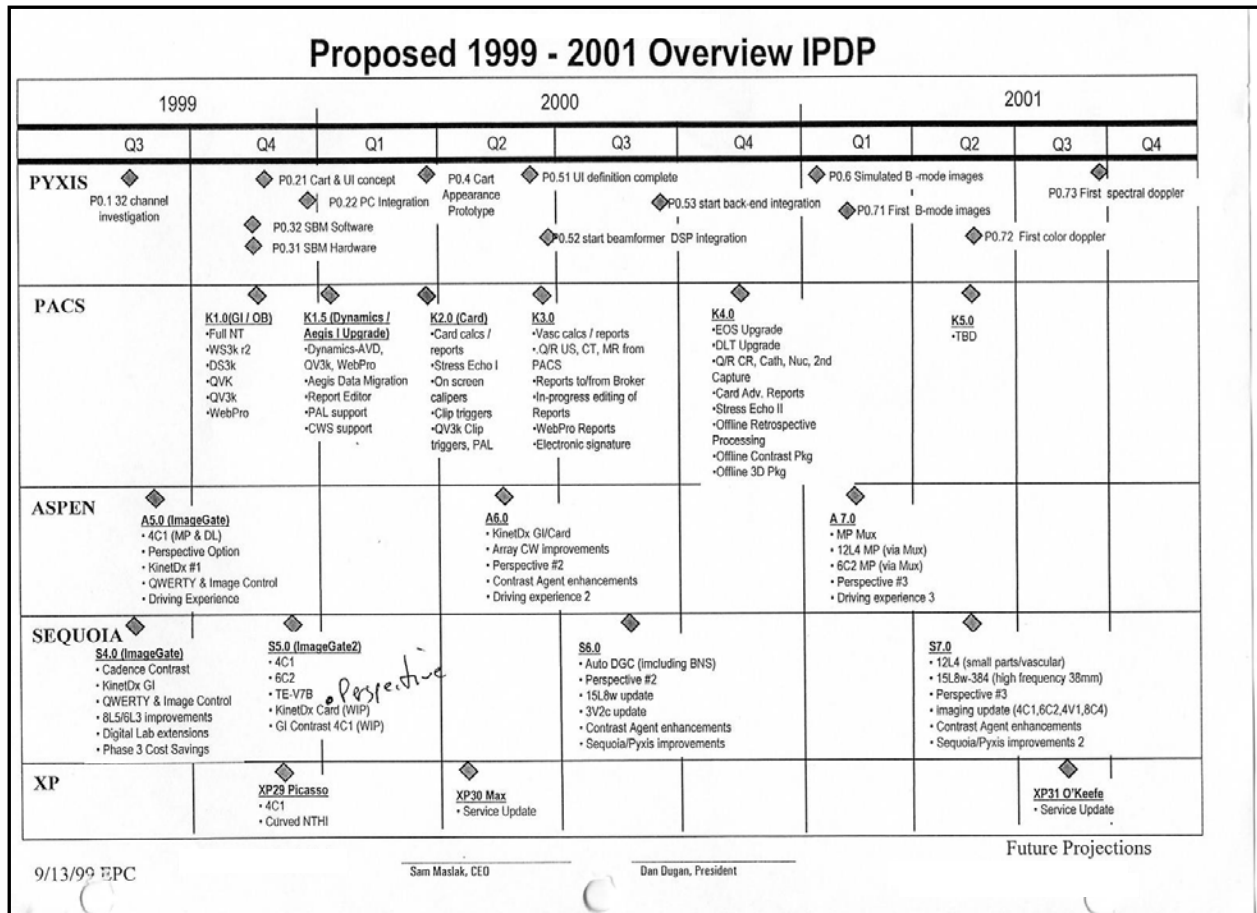


Figure 2. Acuson Release Plan

3.2.2 Human Resource Investment Drives Cost

Engineering costs make a negative contribution to corporate profitability. Engineering costs are tied to headcount, i.e. the number of employees in Engineering. Salary constitutes the majority of engineering costs. Related capital expenses, such as computers, lab equipment, and design automation software, are roughly proportional to headcount. For brevity, all human resource (HR) investment costs will be referred to as salary throughout this paper.

An ultrasound system is a complex signal processing instrument. Sequoia had a list price of approximately \$200,000 and contained 20 square feet of printed circuit boards, more than a dozen microprocessors, and a million lines of embedded software. New engineers at Acuson typically had a several month learning curve before they knew enough about Acuson's products to become productive. Also, the accumulated knowledge of the veteran engineers was vital to developing new product releases quickly. These engineers were compensated well to encourage them to remain at Acuson. Consequently, hiring an engineer was an investment for the Engineering organization, and layoffs were avoided.

Designing and maintaining Acuson's products required engineers experienced in a variety of disciplines. Figure 3 shows the eight specialties needed just to perform electrical and mechanical engineering. Software engineers, image analysis engineers, industrial designers, and

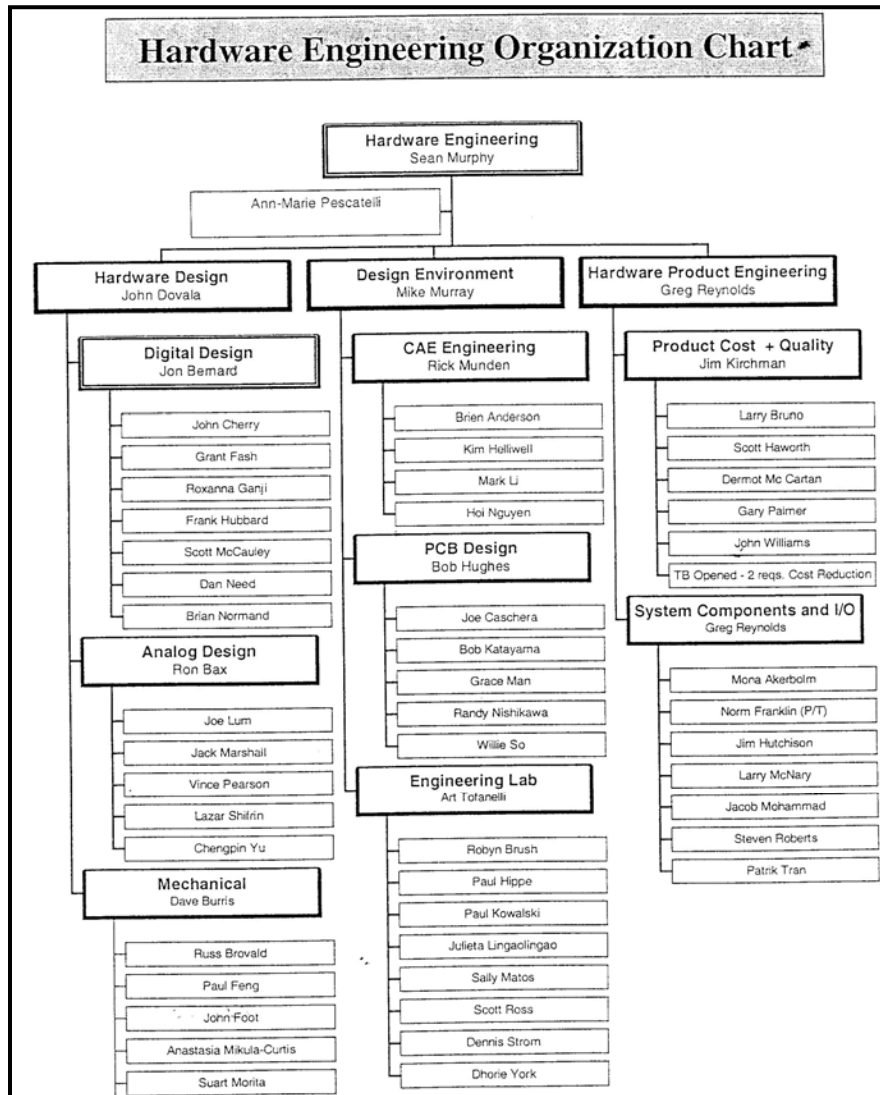


Figure 3. Acuson Hardware Engineering Specialties

sonographers were also needed to develop an ultrasound system. Engineering managers had to decide on the degree of investment in each engineering specialty.

3.2.3 Design Work Process Drives Feasibility

Profit would be maximized by completing design projects in zero time using no employees, but this is clearly infeasible. Feasibility is enforced through design work process constraints, which specify the number of engineers required to accomplish a task in a given time with a given quality. Completing a task in less time requires assigning more engineers to work on it or compromising quality. Feasibility also constrains the extent to which tasks can be fast-tracked. Fast-tracking is the starting of a successor task before one or more of its predecessor tasks are completely finished.

Figure 4 shows some design work process decisions made by Acuson managers. The left hand column lists projects, and the other two columns list the number of engineers assigned to

Developer Resource & Task Profile through End of Year 2000		
Projects	1 st Half – Year 2000	2 nd Half – Year 2000
Pyxis Product		
- Electrical	14.5	14.5
- Mechanical	3.5	5.0
Aspen Product		
- Array CW (A6.5)	1.0	0.1
ΑΑΟ - Mux PSU	J.H.✓	0
ΑΑΟ - Printer Qualification	0.25	0
Sequoia Product		
- BF.4 (S6.0)	0.25	0
- Az Filter (S6.0)	✓	0
Product Cost & Quality		
Cost Reduction		
- Aspen } *	2.25	2.25
- Sequoia }		
- Shared }		
- Mechanical	1.0	1.0
Product Quality Reliability & Sustaining	4.0	4.25
Perspective	0.25	1.0
MISC		
- Ecton	0	0
- JCC-XC	0.3	0
Total	27.3	28.1

Figure 4. Acuson Engineering Allocations

those projects. The labor demands for various projects were summed and compared with the organization chart to verify feasibility of a project portfolio plan.

3.3 Interactions Make the EVPP Interesting

Figure 5 shows a schematic of the EVPP. The dotted rectangle shows the decisions that are within the scope of the EVPP. The solid rectangles outside the dotted rectangle show environmental factors that are treated as givens. Work physics refers to determining the additional coordination load required by a larger task team or by fast-tracked tasks. Work physics also refers to the determining the effort vs. duration vs. quality time tradeoff for tasks and projects.

The essence of the EVPP is that its three elements interact in interesting ways. For example, accelerating a project's completion date by assigning more engineers to a critical path task would increase revenue, but would also increase cost if more engineers had to be hired. However, these newly-hired engineers would be subsequently available to accelerate future projects and increase their revenue.

Conversely, an inability to hire an electrical engineer might cause a task's completion date to slip. The slip might disrupt the temporal dovetailing of a successor task assigned to the software engineering group. This disruption might, in turn, cause other project completion dates to require adjustment.

Each element of the EVPP introduces options and constraints. For example, at Acuson, some project completions could be deferred in order to free engineers to work on higher value projects, but other projects had hard deadlines that demanded staffing. Depending on the Silicon Valley economy, some kinds of specialist engineers might be readily available for hiring, while other

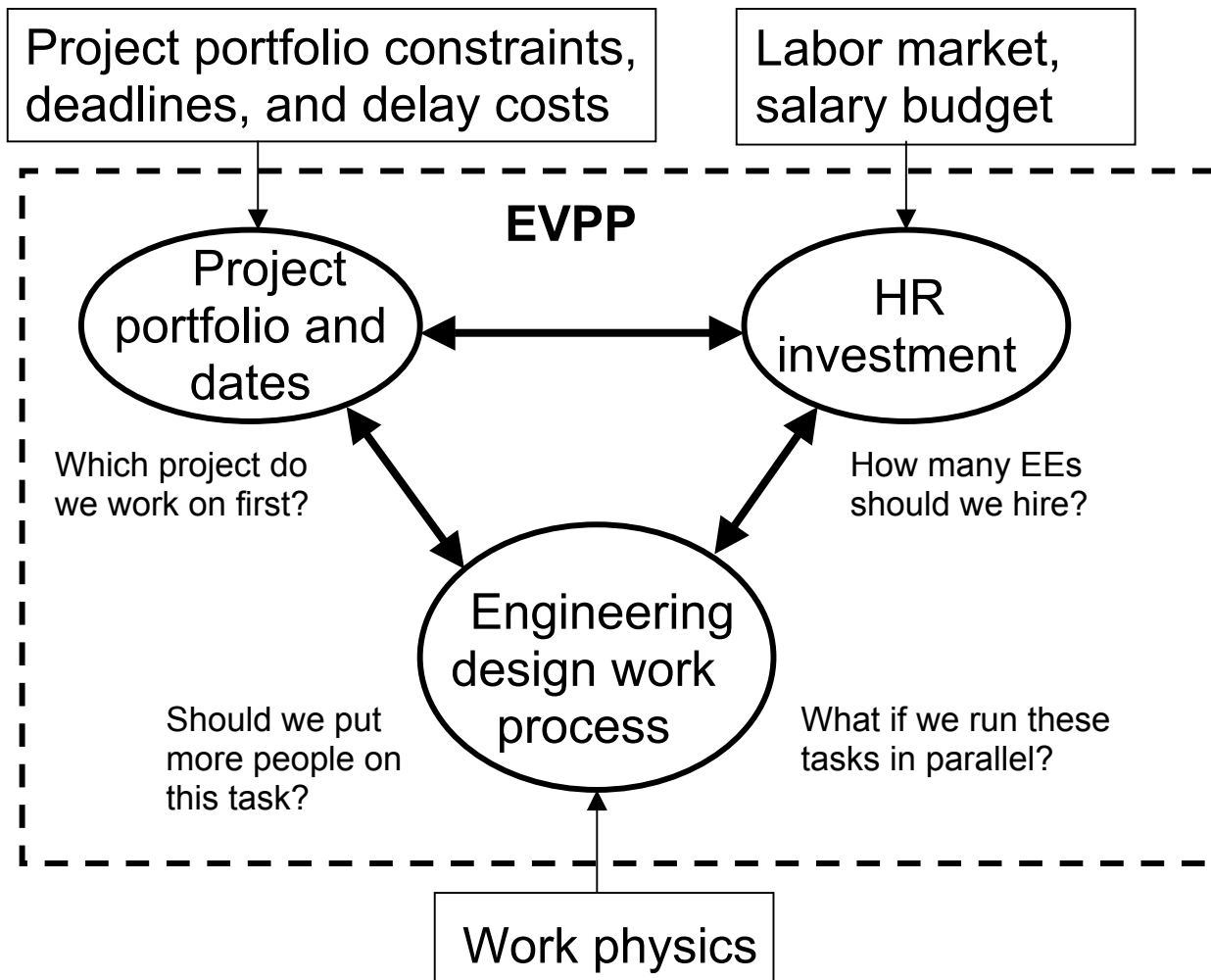


Figure 5. EVPP Schematic

specialists were scarce. A divisible task can be accelerated by assigning a larger team to work on it, but at some point, the coordination load becomes prohibitive.

Optimal engineering management decisions must consider these interactions, options, and constraints.

3.4 Distinguishing Features of the EVPP

What separates the EVPP from the many existing approaches to project scheduling and organizational design?

Under the framework of the EVPP's three elements, I have identified eight distinguishing features of the EVPP. Based on my experience at Acuson, I believe these eight features are required to represent the strategic options and constraints facing engineering managers. These features are like the pieces of a jigsaw puzzle; all are required to see the complete picture of engineering management. This subsection describes these features qualitatively. Section 4

compares the EVPP to prior management science problems with respect to these features. Sections 5.1 and 7.1 describe the mathematical formulations of these features.

The first two distinguishing features relate to the project portfolio and project completion date element of the EVPP.

1. Selectable portfolio: The EVPP allows projects to be added (e.g. to generate revenue from slack resources) or deleted (e.g. to resolve a scheduling bottleneck).
2. Revenue vs. time tradeoff for individual projects in a portfolio. (Figure 6) Engineering organizations typically work on a portfolio of projects simultaneously. The EVPP assumes projects are defined in the portfolio such that they have independent revenue payoffs in the marketplace. The EVPP models the revenue tradeoffs of selecting different project priorities and completion dates.

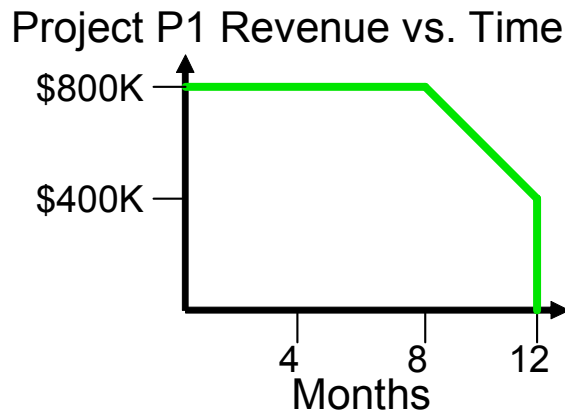


Figure 6. Revenue vs Completion Time Tradeoff for Projects

The next three features relate to human resource investment.

3. Skilled employees. The EVPP models employees with different skill capabilities and tasks with different skill requirements.
4. Limited employee availability. The number of engineers with a particular skill in an organization is limited. The EVPP model allows a cap to be set on the number of engineers with each skill, and then enforces HR consistency in schedules. A schedule is HR consistent when the number of engineers assigned to tasks requiring a particular skill never exceeds the number of engineers employed who possess that skill.
5. Selectable number of salaried employees. Conversely, if there is little need for engineers with a particular skill, some of those engineers can be laid off to reduce HR costs. However, layoffs should be minimized in order to retain corporate knowledge. The EVPP model balances layoffs vs. retention by allowing headcount levels to be chosen at the beginning of a planning period (subject to labor market availability), but then requiring that headcount to be maintained throughout the planning period. The EVPP assumes engineers are paid salary whether they are working on a project or not, allowing modeling of vacation time, sick leave, training, etc.

The final three features relate to engineering work process.

Task T1 Effort vs. Duration vs. Quality Tradeoff

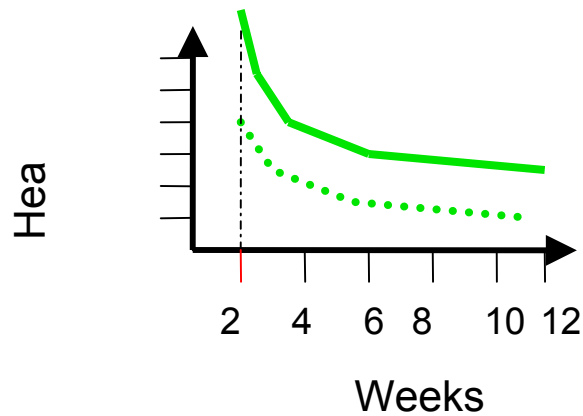


Figure 7. Effort vs Duration vs. Quality Tradeoff for Tasks

6. Effort vs. duration vs. quality tradeoff for tasks. (Figure 7) Assigning more engineers to a task will reduce the task's duration, as shown by the solid green line. The EVPP models this tradeoff, including elasticity. Inelasticity is the extra work of coordinating a larger team. A minimum duration for a task may be specified; think of this as reflecting a practical limit on the maximum team size. The EVPP models the effort vs. duration tradeoff as a continuous function because an engineering team can achieve a particular task duration by adjusting the percentage of its time that it spends on the task.

The effort and/or or duration required to complete a task can be reduced by reducing the quality of the work on that task, as shown by the dotted green line. Alternatively, one can view a downward shift of the effort vs. duration curve as taking a risk on the timely completion of a task.

7. Fast-tracked tasks with a coordination penalty. (Figure 8) Fast-tracking tasks refers to allowing start-to-start precedence relations in addition to finish-to-start relations. The EVPP allows pairs of tasks to require extra coordination work when they are executed in parallel, but not when they are executed serially. This coordination work is shown as extra headcount required throughout the execution of the tasks in order to hold the

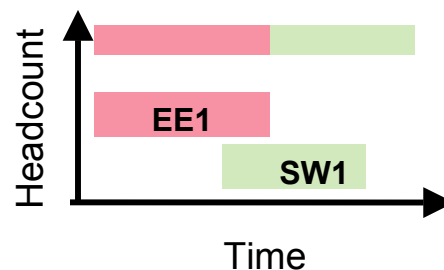


Figure 8. Coordination Penalty for Fast Tracked Tasks

tasks' durations constant. A coordination penalty may also apply when two tasks make exclusive but intermittent use of a single physical resource, such as a piece of equipment.

8. Preemption: The EVPP supports task preemption. The time that a task is preempted does not count towards its duration.

4 Points of Departure and Research Questions

4.1 Models

Demeulemeester and Herroelen (Dem02) provide an in-depth review of several hundred papers in the project scheduling literature, and they construct a taxonomy of features of project scheduling models. Table 1 adapts their taxonomy to show how the EVPP compares with representative previous work in project scheduling and organizational design. All of the EVPP's distinguishing features presented in section 3.4 have been explored previously in the project management literature. The EVPP is unique in its integration of these features.

Previous models can be divided into several families.

The classic CPM time-cost tradeoff models support revenue vs. time and effort vs. duration tradeoffs, but they have no concept of human resource investment. They also do not model reducing effort or duration by compromising quality. "GPR" stands for generalized precedence relations. Under a GPR model, precedence relations may be between task start and/or finish times, may be minimal or maximal, and may have a nonzero time lag. GPR supports fast tracking, but does not model the coordination penalty.

The Resource Constrained Project Scheduling Problem (RCPSp) models have excellent support for skilled employees and limited employee availability, but they typically do not allow employee headcount to be adjusted, nor do they support an effort vs. duration tradeoff. The Job Shop Scheduling formulation is a special case of the RCPSp problem in which the "employees" are single machines. The Resource Availability Cost model does allow headcount adjustments, but it models only a single project with a fixed deadline; no tradeoff of deadline vs. headcount is possible. "Makespan" indicates that the optimizer tries to minimize the completion date of a single project, and tradeoffs among completion dates of multiple projects in a portfolio are not supported. Net present value (NPV) formulations of the RCPSp are designed to consider these tradeoffs. Kaplan (Kap88) studied adding preemption to the RCPSp.

The multi mode RCPSp models add effort vs. duration tradeoff to the RCPSp optimizers. However, they typically support only discrete effort vs. duration task tradeoffs.

The Rough Cut Capacity Planning model comes very close to the EVPP with respect to project completions and HR investment. Hans, the Rough Cut Capacity Planning author, argues (Han01) for the need to integrate capacity planning and scheduling. His optimization algorithm appears to be trivially extendible to selecting salaried employee levels, yet he only considers overtime in his objective function. Also, his work process model is motivated by factory scheduling, and it supports neither inelasticity in the effort vs. duration tradeoff nor a coordination penalty.

The MMRCPSP Pareto formulation computes all the points of a makespan vs. resource Pareto curve for a project. It does not select a point subject to a user specified optimization criterion. I believe it would be computationally prohibitive on a project with many different resource types.

Table 1. Points of Departure: Models

Formulation	Project Portfolio and Completions		HR Investment			Engineering Design Work Process		
	Selectable Portfolio	Revenue / time trade off for individual projects	Skilled employees	Limited employee availability	Selectable number of salaried employees	Effort vs. duration vs. quality trade off for tasks	Fast track tasks with coordination penalty	Task preemption
CPM time-cost tradeoff (Mod83)		✓				effort vs. duration		
CPM time-cost + GPR (Elm92)		✓				effort vs. duration	no penalty	
Job Shop Scheduling (Smi93)		makespan	✓	✓				
RCPSP (Pri69)		makespan	✓	✓				
RCPSP + NPV + GPR (DeR98)		✓	✓	✓			no penalty	
Rsrc Avail Cost + GPR (Dem95)		fixed deadline	✓	✓	✓		no penalty	
RCPSP + Preemption (Kap88)		makespan	✓	✓				✓
Multi Mode RCPSP (Tal82)		makespan	✓	✓		discrete		
MMRCPSP + GPR (DeR99)		makespan	✓	✓		discrete	no penalty	
Rough Cut Capacity Planning (Han01)		✓	✓	✓	overtime	elastic	no penalty	
MMRCPSP Pareto (LeP94)		makespan	✓	✓	✓	discrete	no penalty	
Portfolio selection + scheduling (Gha99)	✓	✓	✓	✓		discrete		
Factory scheduling (Fox83)		✓	✓	✓	overtime	factory work process		
VDT / SimVision simulator (Lev94)	✓	✓	✓	✓	✓	✓	partial	✓

Ghasemadep et. al (Gha99) describe portfolio optimization as a multiobjective optimization problem. Projects are evaluated in terms of fit with corporate strategy, risk, and return on investment. (Gha99) appears to be unique in considering scheduling feasibility under resource constraints in choosing a portfolio. However, the Ghasemadep's scheduling model is very restricted compared with the RCPSP. Scheduling conflicts can only be resolved by time shifting entire projects, not individual tasks within a project.

The factory scheduling models capture manufacturing constraints and optimization criteria such as inventory levels, setup times, and labor work rules, in contrast to the knowledge-intensive work model of the EVPP.

The Virtual Design Team (VDT) model supports nearly all of the features of the EVPP. VDT also models many additional organization structure and policy features not modeled in the EVPP. The VDT model is implemented as the SimVision simulator. SimVision predicts organizational performance but does not improve it. An EVPP optimizer can be seen as the first step towards an optimizer for the full VDT model.

Finally, Goldratt (Gol97) presents the Critical Chain and Buffer Management (CC/BM) methodology for inserting buffers to make a schedule robust given uncertainty in task duration. Demeulemeester and Herroelen (Dem02) critique the method as being oversimplified. Buffer widths are set coarsely by rules of thumb, and the insertion points are not precisely defined. CC/BM is not shown in Table 1 because CC/BM presupposes that the project completion, human resource investment, and engineering work process decisions have been made before the CC/BM methodology is invoked.

4.2 Algorithms

The EVPP subsumes the Job Shop Scheduling Problem, which has been shown to be NP-Hard (Gar79). Therefore, some kind of search algorithm is required for determining an optimal solution to the EVPP. Also, above some problem size, complete search methods will likely become impractical and only suboptimal solutions will be available.

Table 2 surveys the algorithms that have been applied to previous project scheduling problems. These algorithms can be divided into three categories. Many of the papers shown in Table 2 are reviewed in (Dem02) or (Bap02).

Algorithm		Problem	Citations
Complete search	Linear (network) programming	CPM	Dant63, Fulk61, Kel61
	Branch and bound	RCPSP	Lan60, Sti78, Pat89
	Constraint propagation	Job shop	Mack77, Carl90, Smi93
Incomplete constructive methods	List methods w/ heuristics	RCPSP	Kel63, Bro65, Law85
	Probabilistic methods	RCPSP	Coop76, Alv89
	Limited discrepancy search	Job shop	Harv95
	Beam search	Factory scheduling	Fox83
Iterative improvement	Simulated annealing	RCPSP	Kir83, Sam93
	Tabu search	RCPSP	Glov89, Pin94
	Genetic algorithms	RCPSP	Hol75, Hart98

Table 2. Points of Departure: Algorithms

First, there are the complete search algorithms, which are guaranteed to give an optimal solution. Because CPM problems have limited expressiveness relative to the EVPP, CPM problems can be solved in polynomial time with linear or network programming. Branch and bound is the classic approach to NP-Hard search. Constraint propagation has proven to be a useful adjunct to search in solving the Job Shop Scheduling Problem.

Second, there are the incomplete constructive methods which return suboptimal results in bounded run times. All of these methods start with an empty schedule and successively schedule tasks until all tasks are scheduled.

List methods use heuristics to decide deterministically the order in which tasks are scheduled. Dozens of heuristics have been developed, with no one heuristic dominating the others. Classic heuristics are to schedule the shortest job first or the job with minimum slack first. Many of these heuristics are not immediately applicable to the EVPP because of the flexibility of the EVPP. In the EVPP, it's not clear a priori which job will be shortest, which resource will be the bottleneck, or which jobs are on the critical path.

Probabilistic methods extend list methods by choosing the next task to schedule probabilistically instead of deterministically. Because the underlying list method is extremely fast, probabilistic methods are executed iteratively, typically yielding a different result on each run. The best result from a sequence of runs is reported.

Limited discrepancy search (LDS) assumes that a scheduling heuristic will make the right choice nearly all of the time. LDS first explores schedules with one different scheduling decision relative to the heuristic, then two different decisions, then three, etc.

Beam search is a tree search method like branch and bound. Beam search saves time by pruning branches of the search tree heuristically when they are judged to be suboptimal.

Finally, iterative improvement algorithms start with a complete, feasible schedule and try to improve it. They improve the schedule by changing a single scheduling decision (such as the temporal ordering between two tasks) and evaluating the effect of the change. These algorithms are incomplete because they are susceptible to becoming trapped in schedules that are locally optimal but globally suboptimal. They differ in their strategies for choosing which schedule improvements to accept and for escaping local optimality. Simulated annealing probabilistically accepts changes which worsen the schedule. Tabu search considers suboptimal schedules while disallowing a return to a previously considered local optimum. Genetic algorithms "cross breed" good schedules to produce improved hybrids.

4.3 Research Questions

Past project management formulations have typically used discrete algorithms, which are appropriate for capital-intensive work. They assume rigid specification of task durations, precedence constraints, task resource requirements, and resource availabilities. These formulations rely on clever search space enumeration and on constraint propagation to improve performance. However, they do not scale as flexibility increases. Adding more discrete options (e.g. effort vs. duration tradeoffs or selectable resource levels) increases the dimensionality of the search space and weakens constraint propagation without increasing tractability. Essentially, these formulations treat scheduling like packing wooden blocks in a trunk - suboptimality arises when flexibility creates more candidate solutions than can be explored.

However, practicing managers of knowledge-intensive project portfolios have more degrees of freedom than the past models allow. Managers can change project due dates, hire and release

staff, fast track tasks, and trim task scope to fit available time and resources. The EVPP is more like packing water balloons in a duffel bag - good solutions come from exploiting flexibility.

My research explores the solvability of the EVPP. In particular,

- How can the flexibility of the EVPP be exploited in its solution?
- How do problem parameters (e.g. size, degrees of flexibility) affect solvability?
- Are industrial problems typically easy or hard?

5 A Simplified EVPP

Section 3 defined the EVPP and showed a real world example. The table of models in Section 4 demonstrated that no existing optimizer can address the EVPP. This section presents a formulation and an example of a simplified EVPP. The formulation allows the EVPP to be attacked using a combination of continuous optimization and discrete search techniques.

5.1 Formulation

This section describes the mathematical formulation of the EVPP without portfolio selection, task quality tradeoffs, fast-tracking tasks with a coordination penalty, or task preemption. These features are omitted for clarity of exposition. The formulation will be extended to address these features in section 7.1.

The givens, decision variables, objective function, and constraints of the simplified EVPP are described below.

Given:

1. $\{G_p\}$: a set of projects, each modeled as a directed graph. Nodes represent tasks, labeled with optional release times or deadlines. Each edge represents a start-to-start or finish-to-start precedence relation, labeled with an optional minimum lag. The modeling of a project as a directed graph follows standard practice in project scheduling optimizers.
2. $HC_{p,t,s}(dur_{p,t})$: a convex piecewise linear (PWL) full time equivalent headcount (i.e. effort) vs. duration function for each skill s needed for each task t in each project p . The EVPP supports cross-functional tasks that require work by employees with different skills.
3. $rev_p(ft_p)$: a concave PWL revenue vs. finish time function for each project p .

The headcount vs. duration and revenue vs. finish time functions are as they appeared in section 3.4, “Distinguishing Features of the EVPP”. In practice, these functions are nonlinear. I model them as PWL functions to allow the use of linear programming solvers.

4. $HCcap_s$: the maximum possible headcount for each skill s .
5. sal_s : the cost to employ a worker with skill s and average proficiency for the duration of the planning period. Workers with above (below) average proficiency are assumed to be paid proportionately more (less).

Decision variables:

$st_{p,t}$ and $ft_{p,t}$: start time and finish time for each task on each project

Objective:

$$\text{Maximize profitability contribution} = \sum_p \text{rev}_p(\text{ft}_p) - \sum_s \text{sal}_s * \text{HC}_s$$

Profitability contribution is the difference between revenue recognized by completing projects and the salary cost of the human resource investment. HC_s is the number of employees (headcount) hired with skill s for a planning period of given length. Employees are assumed to have only one skill.

The EVPP is intended to model an ongoing engineering organization rather than a single project organization that disbands when the project is complete. Thus, the EVPP optimizes profitability contribution over a fixed planning period and the salary term of the objective function does not depend on project completion dates. Projects are scoped such that they must be completed within the planning period.

Subject to:

1. task precedence relations (e.g. $\text{ft}_{p1,t1} + \text{lag}_{p1,t1,p2,t2} \leq \text{st}_{p1,t2}$) from $\{G_p\}$
2. task release times and deadline times from $\{G_p\}$
3. $\text{dur}_{p,t} = \text{ft}_{p,t} - \text{st}_{p,t} \forall p, t$ $\text{dur}_{p,t}$ is the duration of task t on project p , not counting preemption time.
4. $\text{hc}_{p,t,s} = \text{HC}_{p,t,s}(\text{dur}_{p,t}) \forall p, t$ $\text{hc}_{p,t,s}$ is the number of employees with skill s assigned to work on task t of project p .
5. $\text{HC}_s \leq \text{HCcap}_s$
6. $\text{HC}_s = \max_{\text{concurrent}}(\text{st}_{p,t}, \text{ft}_{p,t}, \text{hc}_{p,t,s}) \forall s$ For each skill s , sufficient employees must be hired to meet the peak concurrent demand for those employees.

Constraint #6, the HR consistency constraint, is the only constraint that is not readily implemented in a linear programming solver. This constraint is problematic because the LP does not know a priori which of the tasks requiring the same skill s will be concurrent. Thus, the contributing terms in a linear constraint on HCs are unknown.

5.2 Example Problem

This section gives an example of a simple EVPP. The project portfolio for this example is shown in Figure 9. The project portfolio comprises two projects, P1 and P2, each comprising an electrical engineering task, a software engineering task, and a manufacturing engineering task. There are linear finish-to-start 0-lag precedence relations among the tasks. Nominal work volumes, in FTE-months (Fm) are shown below each task node.

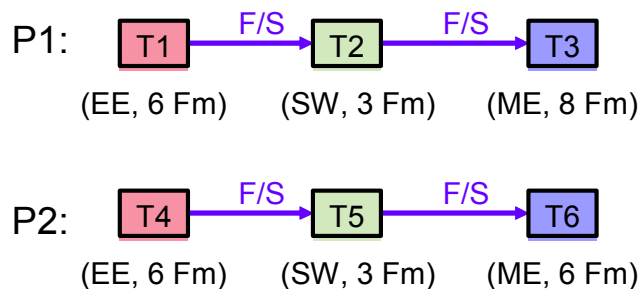


Figure 9. Sample Project Portfolio

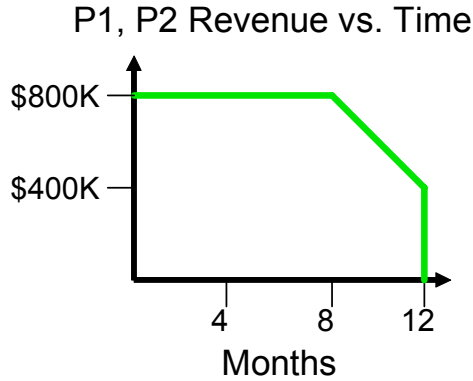


Figure 10. Sample Revenue vs. Completion Time Tradeoff for Projects

P1 and P2 have identical revenue vs. time tradeoff functions (Figure 10). Assume the tasks have elastic headcount vs duration curves (not shown). Assume $HC_{cap_{sw}} = 1$. Figure 11 shows a feasible combination of schedule and staffing profile for the two projects. The area of each shaded rectangle corresponds to the work volume of its task.

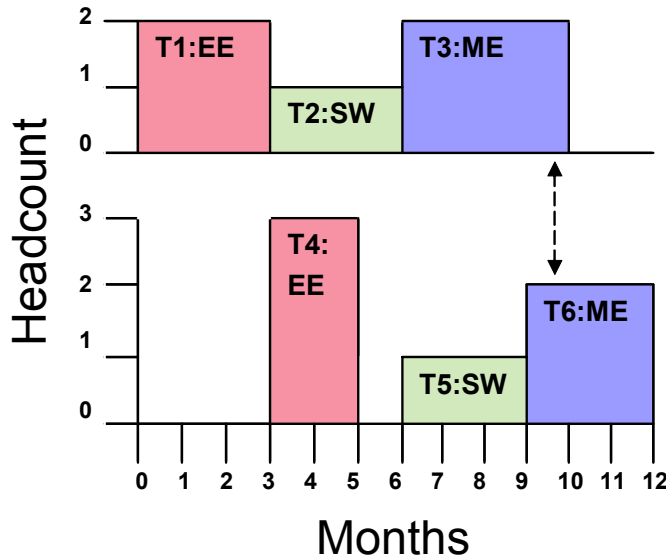


Figure 11. Sample Schedule and Staffing Profile

Three electrical engineers (EEs) must be hired in order to complete T4 in two months as shown. T1 uses only two EEs, hence one EE is assumed idle while T1 executes. Because only one software engineer is available, T5 has been delayed until T2 finishes. Four manufacturing engineers (MEs) must be hired to meet the peak demand during the month when T3 and T6 overlap, as shown by the dotted arrow.

Revenue	\$1000K	P1 = \$600K P2 = \$400K
Cost	\$800K	HC _{EE} = 3 HC _{SW} = 1 HC _{ME} = 4
Profit	\$200K	

Table 3. Profitability Calculation

Table 3 calculates the profit for this schedule and staffing profile, assuming HC_s is \$100K. Revenue is based on absolute project completion time, not start-to-finish time. P2 meets the 12 month deadline in time to realize \$400K in revenue.

This candidate solution could be improved in several different ways. For example

- Three EEs could be assigned to T1, shortening its duration to two months. T2 and T3 could be started and completed one month earlier, leading to \$100K in additional revenue. Also, the overlap between T3 and T6 would be eliminated, allowing ME headcount to be reduced by two, and saving \$200K in salary
- Three EEs could be assigned to T1, shortening its duration to two months. T2, T3, T5, and T6 could all be started and completed one month earlier, leading to \$200K in extra revenue.
- By lengthening the duration of T4 from two months to three, EE headcount could be reduced to two, saving \$100K in salary.

6 Solving the Simplified Formulation

This section describes how linear programming and discrete search can be applied to the simplified formulation of the EVPP. It also gives results from a benchmark test case.

6.1 Linear Programming Approximation

Linear programming (LP) is a highly efficient and scalable optimization technique for problems with continuous variables. As shown in sections 5.1 and 7.1.1 – 7.1.3, LP can capture nearly all of the flexibility in the EVPP. LP’s only limitation is with respect to handling the alternatives for the temporal ordering of tasks. However, this limitation is sufficient to preclude an efficient LP-only solution of the EVPP. In contrast, discrete search is well-suited to exploring temporal ordering alternatives. I propose to solve the EVPP by embedding an LP approximation within a discrete search. The LP approximation is described in this subsection, followed by discrete search in subsection 6.2

The LP approximation estimates the profit of a feasible schedule. (A feasible schedule is one that satisfies the task precedence constraints. A feasible schedule might not be HR consistent.) The formulation of the LP approximation is identical to the formulation of the simplified EVPP, except for the HR consistency constraint

$$HC_s = \max_concurrent(st_{p,t}, ft_{p,t}, hc_{p,t}, cp_{p1,t1,p2,t2}) \forall s$$

In the exact formulation of the EVPP, this constraint makes HC_s equal to the actual peak concurrent demand for employees with skill s . In the LP approximation, this constraint makes HC_s equal to the estimated peak concurrent demand for s -skilled employees. The LP approximation may overestimate or underestimate the maximum concurrent demand for employees, as described below.

Initially, there is no known concurrency among tasks requiring skill s , and this constraint is equivalent to taking the maximum of the headcount allocations of the individual tasks requiring skill s . As the solution of the EVPP progresses, the EVPP optimizer identifies task cliques, i.e. groups of tasks that require a skill s and which are believed all to execute concurrently at some time. For each clique, the sum of the headcount allocations of the tasks in the clique becomes a lower bound constraint on HC_s . However, in some cases, the LP approximation may either overestimate or underestimate the membership of a clique.

The LP approximation is improved iteratively. Each iteration of the approximation comprises a clique estimation phase followed by an LP solution phase. The N th LP approximation estimates clique membership based on the task start and finish times determined in the $(N-1)$ st LP solution. The N th clique estimation uses these estimates to construct HC_s constraints for the N th LP solution. However, as the N th LP is solved, the LP solver shifts tasks temporally, and task cliques may be created or destroyed. The HC_s constraints used in creating the N th LP solution may not correspond to the cliques that actually exist in the solution.

The N th approximation is inexact iff the N th solution is over or under constrained. The N th solution is under constrained if it contains a new s -skill task clique that requires more headcount than has been allocated by the LP solver for skill s . The N th solution is over constrained if the HC_s constraint corresponding to an $(N-1)$ st iteration clique is tight in the N th solution, but that clique no longer exists in the N th solution.

If the N th approximation is exact, the corresponding schedule is HR consistent. A sufficient condition for an exact LP approximation is that no task clique is created or destroyed during the N th LP solution. No task cliques will be created or destroyed if there are sufficiently many constraints on the temporal orderings of the pairs of s -skill tasks.

6.2 Embedding the LP Approximation in Discrete Search

The EVPP solver uses discrete search to specify task temporal orderings for use in the LP approximation. The discrete search engine must determine a set of orderings such that the LP approximation is exact and maximal.

When the discrete search begins, the only task ordering constraints are those given for the individual projects in the precedence graph set $\{G_p\}$. Peaks in demand for s -skilled employees will occur if tasks (typically from different projects) that require skill s are executed concurrently. These peaks inflate salary costs and reduce profit. The discrete search engine attempts to level these peaks without delaying the projects and reducing their revenue.

Consider the example from section 5.2. Manufacturing engineering tasks T3 and T6 are initially scheduled to execute concurrently, creating a spike in demand for manufacturing engineers. There are three alternatives for ordering the tasks in the pair and leveling the spike:

- T3 can be constrained to complete before T6 begins (designated $T3 < T6$), or
- T6 can be constrained to complete before T3 begins (designated $T6 < T3$), or
- Sufficient MEs can be hired to staff both T3 and T6 concurrently, and T3 and T6 can be allocated more time to reduce their individual headcount requirements. (designated $T3 \parallel T6$)

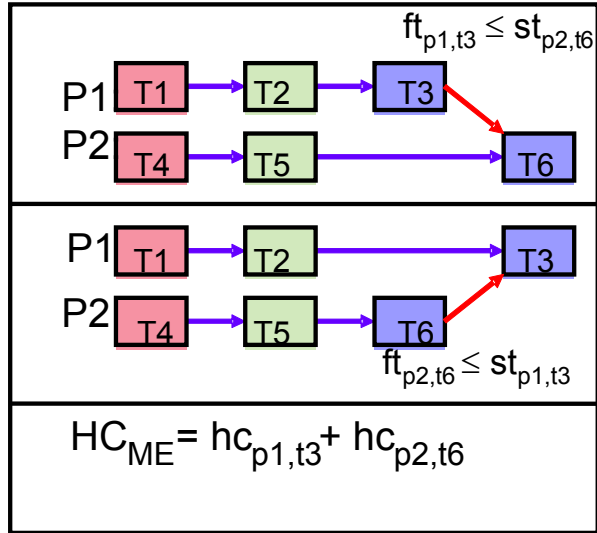


Figure 12. Alternatives for Task Pair Orderings

These alternatives are shown in Figure 12. The necessary constraints are easily expressed in LP form.

The variables in the discrete search are pairs of tasks that require the same skill. The values of these variables are the three ordering alternatives described above. The discrete search's variable and value ordering heuristics are the key to its performance. I have investigated a variety of these heuristics. In the next section, I describe the major features of the heuristic with the best performance.

I believe the complexity of the EVPP makes complete search impractical for nontrivial problems, consequently, I have only investigated incomplete search.

6.3 Heuristics for Discrete Search

6.3.1 The Problem

Consider a feasible, HR consistent schedule with a specific temporal ordering for a pair of tasks, e.g. $T3 < T6$. (Assume such a schedule exists. If not, pick a different ordering for the task pair, or a different task pair.) The profit P of this schedule depends on the specific ordering of $T3$ and $T6$ (i.e. $T3 < T6$) and the orderings chosen for other pairs of tasks. If we hold $T3 < T6$ and vary the orderings for other pairs of tasks, we can construct schedules with a range of profits. Different orderings for $T3$ and $T6$ will yield different profit ranges, as shown in Figure 13. Because one of these orderings must hold in the optimal schedule, at least one of these ranges will include the optimal profit. Call this profit P_{opt} .

Now consider the perfect value selection heuristic for discrete search in the EVPP. For any specific ordering of a task pair, the perfect value selection heuristic would return the maximum profit obtainable in a feasible, HR consistent schedule with that ordering. For example, from Figure 13, this heuristic would return \$800K for $T3 < T6$ and \$600K for $T6 < T3$. Let P_{A1} designate the maximum profit for a feasible, HR consistent schedule with task ordering alternative $A1$, e.g. $P_{T3 < T6}$.

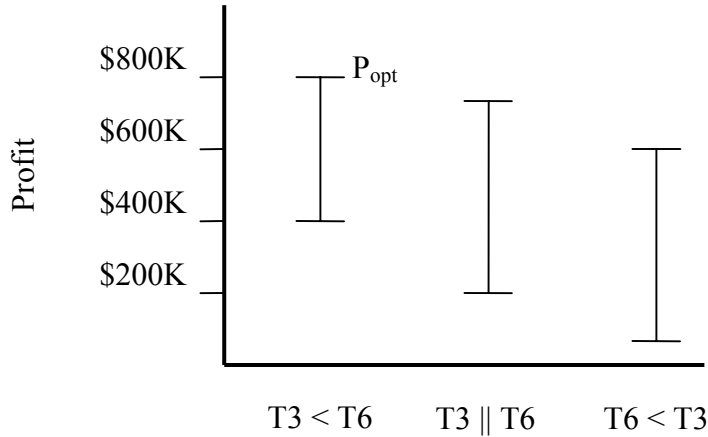


Figure 13: Profit Ranges for Task Pair Ordering Alternatives

Discrete search is trivial given this heuristic. Arbitrarily pick any pair of tasks, and use the heuristic to evaluate the maximum profit for each ordering alternative. Choose the ordering that yields the greatest profit, and continue with another task pair. Repeat until the LP approximation is exact and has profit of P_{opt} . Because the heuristic is perfect, the search will always select the task pair ordering that leads to P_{opt} , and no backtracking will be necessary.

Now consider an imperfect heuristic. For a task pair ordering alternative $A1$, the imperfect heuristic will return a value P_{A1}^I different than P_{A1} . If the heuristic is based on constructing a feasible, HR consistent schedule, then $P_{A1}^I \leq P_{A1}$, otherwise $P_{A1}^I \leq P_{A1}$ or $P_{A1}^I \geq P_{A1}$, depending on the conservatism of the heuristic. For now, assume the imperfect heuristic is based on constructing a feasible, HR consistent schedule, i.e. the heuristic makes ordering choices for other task pairs in addition to the task pair being evaluated. (Call the task pair being evaluated the primary task pair. Call the other task pairs secondary.) Imperfection in the heuristic arises because the heuristic makes an incorrect choice for some or all of the secondary task pairs.

Discrete search is no longer trivial with the imperfect heuristic. In particular, P_{A1}^I may be greater than P_{A2}^I but $P_{A1} \leq P_{A2}$. In this case, discrete search will choose the wrong alternative for the primary task pair ordering and backtracking will be required to find the optimal solution.

The likelihood of an error depends on the difference in maximum profit between the ordering alternatives of the primary task pair and on the degree of imperfection of the heuristic. Assume choosing $A1$ for the primary pair yields P_{opt} , and $A2$ is the next best alternative. If $(P_{A1} - P_{A2}) > (P_{A1} - P_{A1}^I)$, (i.e. $P_{A1}^I > P_{A2}$), discrete search is still guaranteed to make the right choice. Let us consider how to minimize the imperfection of the heuristic, $P_{A1} - P_{A1}^I$. A method for selecting the primary task pair to order (i.e. a variable ordering heuristic) will be considered in section 6.3.3.

Minimizing $P_{A1} - P_{A1}^I$ depends on making the correct choice for the orderings of the secondary task pairs in the schedule. However, making correct ordering choices is the general objective of the discrete search. The problem appears to be recursive.

6.3.2 Value Ordering Heuristic

I believe the recursion can be broken by making ordering choices that are merely sufficiently good for the secondary task pairs. In particular, as long as the most important secondary pair orderings are chosen correctly, the flexibility of the EVPP allows some task pairs to be ordered

incorrectly without dramatically reducing profit. Furthermore, the most important secondary task pair orderings will have a large value for $P_{A1} - P_{A2}$, thus it will be easy to make the correct choice for these pairs.

I believe a fast, greedy, local optimization heuristic can make sufficiently good ordering choices for the secondary task pairs. The local optimizer makes ordering choices that are guaranteed to improve the profit of a schedule. Every choice the optimizer makes reduces $P_{A1} - P_{A1}^1$ for the primary pair. Furthermore, the optimizer makes all such choices it can. It is imperfect only in that different choices for some secondary task orderings might have produced even greater profit improvements.

The local optimizer increases profit by serializing pairs of temporally overlapping tasks. It evaluates the profit improvement of serializing a task pair without calling the LP solver. Consequently, the local optimizer executes quickly. Instead of calling the LP solver, the local optimizer assumes that the start and finish times of all the other tasks in the schedule are as given by last LP solution, and it tries to reorder the secondary pair in place. The local optimizer begins with an LP solution that includes all previous discrete search decisions, plus a new candidate ordering for the primary pair, e.g. A1.

Figures 14a and 14b show the operation of the local optimizer. The local optimizer arbitrarily selects a secondary task pair, e.g. $\{T1, T2\}$. The local optimizer first considers the alternative $T1 < T2$. It calculates the headcounts hc'_{t1} and hc'_{t2} needed to begin T1 at st_{t1} and finish at t' , while starting T2 at t' and finishing at ft_{t2} . (The EVPP assumes that a task may be started later or finished earlier without affecting feasibility.) t' is chosen to minimize $\max(hc'_{t1}, hc'_{t2})$. If $\max(hc'_{t1}, hc'_{t2}) < hc_{t1} + hc_{t2}$, and $HC_s \geq hc_{t1} + hc_{t2}$ was a tight constraint in the LP solution, then serializing T1 and T2 will improve profit in most cases. The local optimizer estimates the profit improvement as

$$[(hc_{t1} + hc_{t2}) - \max(hc'_{t1}, hc'_{t2})] * (\text{shadow cost of } HC_s \geq hc_{t1} + hc_{t2} \text{ constraint}) * sal_s$$

Profit may not improve if T1 or T2 is also involved in a different clique. Figures 15a and 15b show an example. The increase in hc_{t2} to hc'_{t2} necessary to start T2 later makes the $\{T2, T3\}$

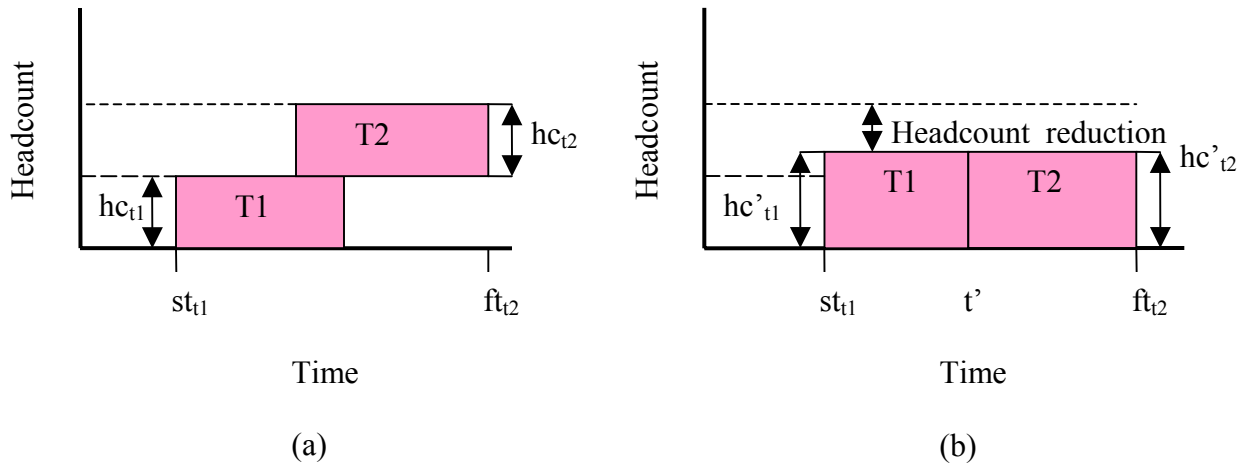


Figure 14. Reducing Peak Headcount by Serializing Tasks

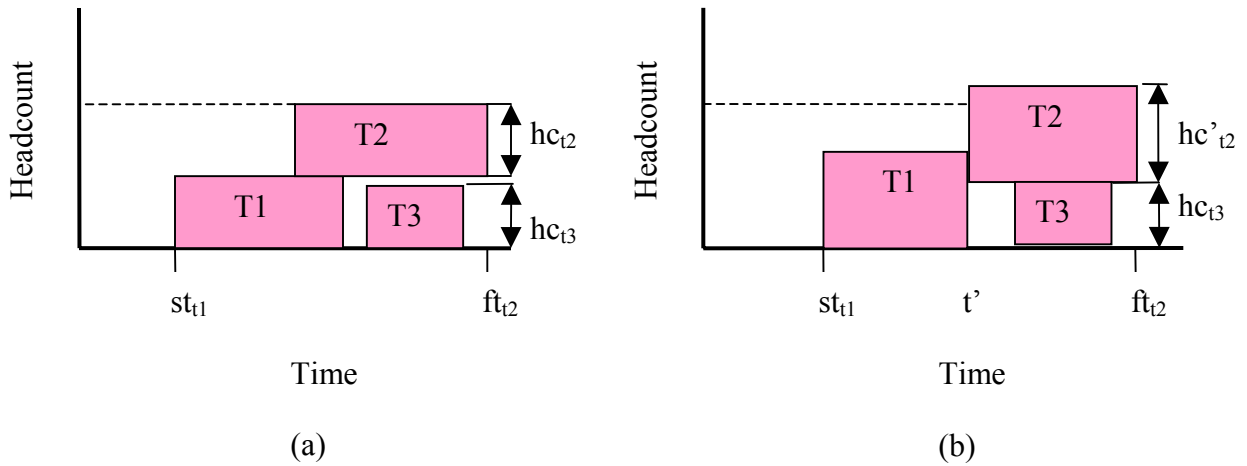


Figure 15. Serializing a Task Pair Can Increase Peak Headcount

clique into a new higher, headcount peak.

Even if $HC_s \geq hc_{t1} + hc_{t2}$ is slack, serializing T1 and T2 may improve profits. (Figures 16a and 16b.) Completing T1 early removes T1 from the {T1, T3} clique, lowering the peak headcount.

After evaluating $T1 < T2$, the local optimizer evaluates $T2 < T1$, and saves the ordering (if any) that produces the greater estimated profit improvement relative to allowing the tasks to overlap.

The local optimizer evaluates all other pairs of overlapping secondary tasks and chooses the serialization that produces the greatest estimated increase in profit. It temporarily adds the chosen serialization to the LP constraint set and calls the LP solver to compute the LP

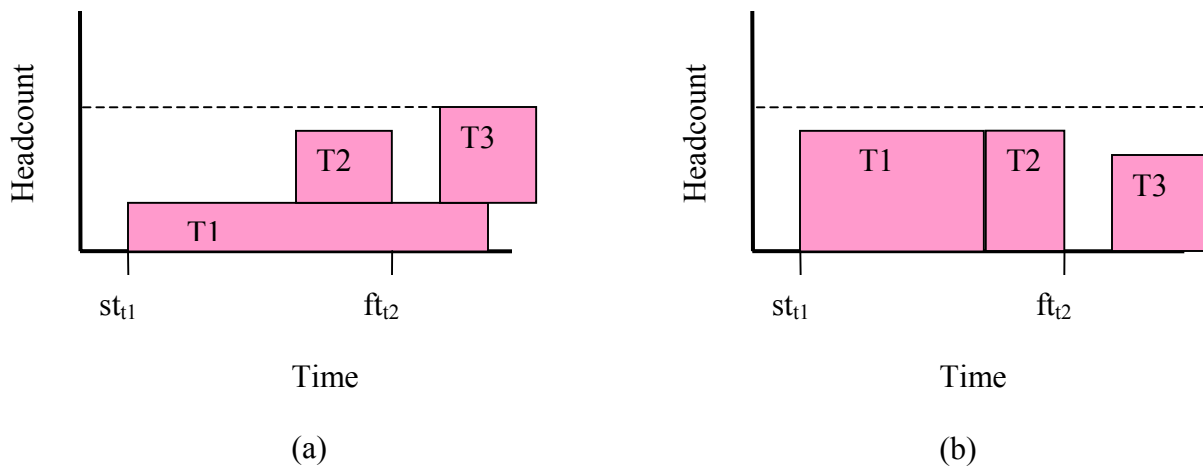


Figure 16. Serializing a Slack Task Pair Can Reduce Peak Headcount

approximation. The local optimizer's chosen serialization may split a clique that existed in the previous LP solution. This split is considered during the clique determination phase of the LP approximation.

The local optimizer repeats the evaluate pairs – choose serialization – solve LP cycle until no secondary pair can be serialized to improve profit. Typically, the LP approximation will be exact at this point. If the LP approximation is inexact, additional temporary serializing constraints are added to the LP until the LP approximation is exact.

Finally, the local optimizer removes the temporary serializing constraints from the LP and returns the profit of the last LP solution as the value of P_{A1}^I . The discrete search engine calls the local optimizer to determine P^I for the other ordering alternatives of the primary task pair, and then chooses the ordering alternative with the greatest profit.

6.3.3 Variable Ordering Heuristic

Section 6.3.2 described how the discrete search engine selects the right value (i.e. task pair ordering) for a given variable (primary task pair). I have done less investigation of variable ordering heuristics, and the optimizer's variable ordering heuristic is crude. The optimizer first evaluates two classes of primary task pairs:

1. task pairs that are members of a task clique whose HC_s constraint has a shadow cost above an arbitrary threshold
2. task pairs for which an ordering decision has already been made by discrete search

If there is no primary task pair in class 1 (2) for which choosing (changing) its ordering increases the schedule's profit, then the optimizer evaluates all other overlapping task pairs. The optimizer chooses (or changes) the task pair ordering to produce the greatest increase in profit. If there is no ordering choice or change of any pair that increases profit, the optimizer terminates.

The optimizer could backtrack instead of terminating. However, I believe that if the optimizer makes discrete search choices carefully and tests for mistakes after every choice, then backtracking will yield little improvement.

6.4 Testing the Optimizer

6.4.1 Test Case

Sadeh (Sad91) developed a suite of 60 job shop scheduling benchmark problems that are frequently used for evaluating job shop scheduling algorithms. I derived an EVPP test case from one of these problems.

The project portfolio comprises 10 mandatory projects; each project comprises 5 tasks. Workers in the organization completing these projects are assumed to have one of 5 skills, labeled A – E. Each task requires a single skill. All of the tasks on a project require different skills, consequently, the tasks on each project collectively require skills A – E.

The tasks in a project are linked by linear precedence relations, i.e. each task has at most one predecessor and at most one successor. The third task of each project requires skill C, otherwise skills are needed in different orders on the different projects.

Each task has a work volume wv that specifies how many time units a single worker will require to complete the task. Overall, task work volumes range from 3 to 15, but work volumes for tasks requiring skill C range from 10 to 15. Task teams with $w \leq 2$ FTE (full time equivalent) workers are assumed to work efficiently, i.e. they will require wv / w time units to complete a task with work volume wv (subject to a piecewise linear approximation). Teams with more than

Project #	Revenue	Total Work Volume	Due Date	Earliest Finish	Roll off	Deadline
0	4	43	50	17.2	0	50
1	3	44	12.5	17.6	25	37.5
2	3	47	50	18.8	0	50
3	3	49	25	19.6	12.5	37.5
4	2	35	37.5	14	0	37.5
5	2	45	25	18	0	25
6	2	37	12.5	14.8	12.5	25
7	2	45	37.5	18	25	50
8	1	42	50	16.8	0	50
9	1	36	25	14.4	25	50

Table 4: Test Case Project Parameters

2 workers are assumed to work inefficiently, for example, if $w = 3$, the team is assumed to complete the task in 40% of the time required by one worker, rather than 33%. I assume that assigning more than 3 workers to a task produces no additional shortening of its duration.

I assume workers are hired for a fixed scheduling period of 50 time units and that $sal_s = 1 \forall s$.

I assigned each project a revenue, a due date, a roll off, and a deadline. Revenue ranges from 1 - 4 and is normalized to the cost of hiring an employee for the 50 unit planning period. A project's due date is the last date at which the project can be completed and still earn the given revenue. The roll off is the period of time after the due date during which the project may be completed and realize pro-rated revenue. A project must be completed by its deadline. Due dates and deadlines occur at quarterly intervals throughout the planning period. Table 4 shows the parameters for the 10 projects in the portfolio. Earliest finish time is calculated by assigning three workers to each of the tasks on a project.

I solved this problem by hand and obtained a profit of 6.77. I believe this is the optimal solution, but I have not verified it with a complete search algorithm.

6.4.2 Results

I implemented the optimizer in approximately 2300 lines of `nawk`, an interpreted text processing language. The optimizer uses Ilog's CPLEX as the LP solver. All tests were run on a Sun Blade workstation

An upper bound for profit can be obtained by solving the test case using only the precedence constraints inherent in the 10 projects and $HC_s = \max(hc_{p,t,s}) \forall s$. This form of the HR consistency constraint assumes that individual tasks using the same resource will not overlap temporally. If this assumption indeed holds true in the solution, then the optimal solution has been found, and further search is unnecessary. Under this assumption, the LP solver calculates profit = 10.89 in 0.04 seconds of compute time.

A lower bound for profit can be obtained by making the upper bound LP solution HR consistent. (HC_s is assumed $\leq HCcap_s$ when HR consistency holds. If not, assume a penalty function.) For the test case, making the upper bound LP solution HR consistent yields a profit lower bound of -10.63. Hereafter, this solution will be referred to as the unoptimized solution.

Project #	Revenue Potential	Due Date	Dead-line	Unoptimized			Optimized		
				Start Time	Finish Time	Revenue Realized	Start Time	Finish Time	Revenue Realized
0	4	50	50	0	50	4	26.2	50	4
1	3	12.5	37.5	0	22.3	1.82	0	29.2	1.00
2	3	50	50	0	50	3	5.2	50	3
3	3	25	37.5	0	25	3	0	25.7	2.84
4	2	37.5	37.5	0	37.5	2	10.2	37.5	2
5	2	25	25	0	25	2	1.4	23.7	2
6	2	12.5	25	0	18.7	1.01	0	19.2	0.93
7	2	37.5	50	0	37.5	2	3.6	37.5	2
8	1	50	50	0	50	1	23.7	48	1
9	1	25	50	0	25	1	22.2	41.7	0.33
Total Revenue						20.8			19.1

Table 5A: Project Results

Skill	Unoptimized		Optimized	
	Headcount	Largest Clique	Headcount	Largest Clique
A	6.0	4 (x 2)	2	1 (x 10)
B	6.1	4 (x 2)	2	1 (x 10)
C	9.0	5 (x 2)	3.95	2 (x 8)
D	4.7	3 (x 1)	2.29	1 (x 10)
E	5.5	4 (x 2)	2	1 (x 10)
Total Headcount	31.3		12.2	

Table 5B: Headcount Results

The local optimizer alone can improve the profit of the unoptimized solution to 4.64 by inserting 15 serializing constraints and 18 concurrency constraints in 11 seconds. Assuming that a profit of 6.77 is optimal, the local optimizer achieves $(4.64 - -10.63) / (6.77 - -10.63) = 87.7\%$ of the potential profit improvement. Thus, the local optimizer yields a good, quick solution to the test case. Also, this result indicates that the local optimizer holds promise for being a near perfect heuristic. (See section 6.3.1).

Enabling both discrete search and the local optimizer yields a profit of 6.77, which agrees with the best hand-optimized solution. Table 5A gives the results for project start and finish times and revenue for the unoptimized and optimized solutions. Table 5B gives headcounts and largest task clique size by skill for the optimized and unoptimized solutions. The (x y) entry in the largest clique column indicates that there were y different task cliques of the largest size. Some tasks were members of more than one clique of largest size.

The optimizer reduces headcount from 31.3 to 12.1, while reducing revenue by only 1.7 and meeting all deadlines. The optimizer levels headcount by delaying the start of tasks with late due dates and by trading off on-time project completion for headcount reduction. Projects #1, #3, #6, #7, and #9 each have a deadline after their due date or earliest possible finishing time. For all of these projects except #7, the optimizer elects to finish the project after its due date and earliest finish time. Most of the revenue reduction during optimization comes from delaying the completions of projects 1 and 9.

The optimizer finishes in 2 hours and 14 minutes. It spends 72% of this time executing the `nawk` code and 28% of this time executing Cplex, the LP solver. The optimizer adds 32 serializing constraints and 8 concurrency constraints to the original problem

7 Next Steps

7.1 *Extending the EVPP Formulation*

The formulation presented in section 5.1 and the solution methods presented in section 6.1 to 6.3 can be extended to support portfolio selection, task quality tradeoffs, task coordination penalties, and preemption. Portfolio selection, task quality tradeoffs, and task coordination penalties are implemented by extending the effort vs. duration tradeoff curve for tasks. Preemption is implemented by extending the choices available to the discrete search engine and the local optimizer.

7.1.1 Portfolio Selection

Portfolio selection can be implemented by extending the notion of the effort vs. duration curve for tasks. Effort for a task can be made a function of a “project fraction” variable in addition to duration. The effort required on a task for a given duration can be reduced by reducing the project fraction, but reducing project fraction also reduces the revenue realized from the project. In the LP, project fraction is constrained to be in the interval $[0,1]$.

In practice, projects are not planned to be partially completed, but the LP solver may return a project fraction value in $(0,1)$ for some projects. In this case, the discrete search engine will need to evaluate profit when the project fraction is constrained to be 0 and constrained to be 1.

An alternative approach to portfolio selection would be to have two optimization loops, one embedded in the other. The inner optimization loop would optimize a given portfolio; the outer loop would optimize the choice of projects in the portfolio given to the inner loop. The approach described above should be faster because portfolio optimization and task scheduling decisions can be made concurrently.

7.1.2 Task Quality

Implementing task quality is similar to implementing portfolio selection. The effort vs. duration curve is made a function of a task quality variable in addition to duration. The effort required on a task for a given duration can be reduced by reducing the task’s quality, but reducing quality on a task reduces the revenue realized from the task’s project.

7.1.3 Fast-Tracked Tasks with Coordination Penalty

Coordination penalties are implemented by making a task’s effort vs. duration curve sensitive to the task’s temporal overlap with its predecessors or successors and to the effort applied to the

predecessors or successors. This is a more complex sensitivity than those described under portfolio selection and task quality because it involves two variables per penalty and is nonlinear. Consequently, implementing this sensitivity will require updating the coefficients of the coordination penalty equation after every iteration of the LP approximation. This updating may prevent the LP approximation from converging unless additional constraints are added to the LP approximation. For example, the LP approximation could bound the change in the length of the temporal overlap or the change in the overlapping task's effort that is allowed in each iteration.

7.1.4 Preemption

Task preemption can be implemented as an additional task pair ordering alternative available to discrete search and to the local optimizer. Suppose discrete search wants to evaluate preempting task T1 of project P1 by task T2 of project P2. Conceptually, task T1 is replaced by two tasks T1a and T1b. The preemption can be expressed as follows.

1. Add decision variables $st_{p1,t1a}$, $ft_{p1,t1a}$, $st_{p1,t1b}$, and $ft_{p1,t1b}$ to the LP.
2. Redefine the task duration
 - a. Remove the constraint $dur_{p1,t1} = ft_{p1,t1} - st_{p1,t1}$ from the LP.
 - b. Add the constraint $dur_{p1,t1} = ft_{p1,t1a} - st_{p1,t1a} + ft_{p1,t1b} - st_{p1,t1b}$
3. Ensure headcount continuity in T1.
 - a. Add the constraint $hc_{p1,t1a,s} = hc_{p1,t1b,s} = hc_{p1,t1,s}$ for each skill s required by T1.
4. Preempt T1 by T2.
 - a. Add the constraint $ft_{p1,t1a} \leq st_{p2,t2}$
 - b. Add the constraint $ft_{p2,t2} \leq st_{p1,t1b}$
5. If T1 has already been serialized with respect to another task T4, modify the associated precedence constraints to refer to T1A or T1B instead of T1.
6. Require total preemption of T1's headcount. Any T1 workers who do not work on T2 are assumed to be idle until T2 finishes.
 - a. Add the constraints $hc_{p2,t2,s} \geq hc_{p1,t1,s}$ for each skill required by T1, or if T2 preempts additional tasks besides T1, add the constraints $hc_{p2,t2,s} \geq \sum \text{skill } s \text{ headcounts of tasks preempted by T2.}$
 - b. Change constraint 4 of the simple formulation to an inequality, i.e $hc_{p2,t2,s} \geq HC_{p2,t2,s}(dur_{p2,t2})$

I have chosen to require total preemption of T1's headcount, rather than allow for some of the T1 workers to continue working on T1 or to be deployed to another task. Mathematically, allowing partial preemption makes the problem nonlinear. Managerially, allowing partial preemption may make it difficult to maintain continuity of the team members working on a task. Because the optimizer can adjust HC_s , $hc_{p1,t1,s}$, and $hc_{p2,t2,s}$, I expect that when the optimizer chooses to use preemption, there will be a good match between $hc_{p1,t1,s}$ and $hc_{p2,t2,s}$, and total preemption will not cause significantly more idleness than partial preemption.

The local optimizer can be similarly augmented to calculate the headcount changes resulting from preempting a task.

7.2 Testing

Because the optimizer uses incomplete search, I do not know if the optimizer's solution to my test case is optimal. I have designed a complete branch and bound search algorithm. I plan to implement it, determine the optimal solution for the test case, and compare it with the result from incomplete search.

I have tested the EVPP optimizer on only the one test case described in section 6.4.1. I plan to test the EVPP optimizer on additional academic and industrial test cases.

PSPLIB (Kol96) is a Web-accessible set of project scheduling benchmark problems that is frequently cited in the RCPSP literature. I plan to construct EVPP project portfolios by combining individual projects from PSPLIB. I also plan to solicit test cases from the industrial partners in Stanford's Center for Integrated Facilities Engineering.

7.3 Performance Tuning

My research so far has focused on rapidly prototyping different solution strategies rather than on tuning the algorithms and software implementation for efficiency. If solving additional test cases confirms the validity of the combination of linear programming, discrete search, and local optimization, I can begin performance tuning. There are several opportunities for reducing the optimizer's run time.

First, the interaction of the local optimizer and the variable ordering heuristic can be optimized. Currently, the local optimizer gives a good result fairly quickly (e.g. capturing 87.7% of the potential profit improvement in 11 seconds. See section 6.4.2.) However, the local optimizer is imperfect – it does not capture all of the potential profit improvement. Given that the local optimizer is imperfect, in order for the discrete search engine to be confident that A_1 is the preferred ordering for a task pair, $P_{A_1}^I - P_{A_2}^I$ for that pair must exceed a threshold value. Not all primary task pairs have a sufficiently large $P_{A_1}^I - P_{A_2}^I$ value. Reducing the imperfection in the local optimizer reduces the $P_{A_1}^I - P_{A_2}^I$ threshold value, which in turn reduces the number of primary pairs that must be evaluated before a decision can be made.

In solving the test case, the optimizer typically evaluated 20+ primary pairs before making a decision. I propose that doing more extensive local optimization may improve the local optimizer's quality, such that only a handful of primary pairs needs to be evaluated per decision. The reduction in the number of primary pairs evaluated may compensate for the additional time spent doing local optimization. Also, finding a heuristic to identify primary pairs with large values of $P_{A_1} - P_{A_2}$ would reduce primary pair evaluations.

Using a less sophisticated local optimization that produces poorer results faster is probably not useful. As the optimizer nears the end of its run, the number of primary pairs becomes small. Furthermore, the $P_{A_1} - P_{A_2}$ values for these pairs are small, too. A less sophisticated local optimizer will be unlikely to find a decision that it can make with confidence.

The second performance enhancement involves the LP solver. The dual simplex algorithm used by the LP solver allows the solver to exploit a previous solution when finding a solution to a slight modification of the preceding problem. Exploiting a past solution can reduce the LP run time by a factor of two or more. The local optimizer adds and deletes LP constraints redundantly, and I believe this redundancy defeats the LP solver's dual simplex algorithm. The local optimizer can be tuned to send only the minimal set of constraint changes to the LP.

Third, rewriting the optimizer in a compiled language such as C++ would significantly increase its speed. Finally, the task sorting and clique membership algorithms in the optimizer were chosen for quick implementation. They can be profiled and tuned for speed.

7.4 Future Research

There are four possible directions for longer term follow-on research to the EVPP.

First, the EVPP formulation has some limitations that could be addressed in future research.

The EVPP uses a linear formulation in order to increase the likelihood of solving the EVPP quickly. For example, in practice, the effort vs. duration functions for tasks and the revenue vs. finish time functions for projects are nonlinear. The EVPP approximates them with piecewise linear functions. Also, the planning period length is treated as fixed. Treating planning period length as a variable (T) would create a nonlinear salary term in the objective function.

$$\sum_p \text{rev}_p(\text{ft}_p) - \sum_s \text{sal_rate}_s * \text{HC}_s * T$$

An intermediate alternative would be to treat the planning period as comprising several fixed length intervals (e.g. planning for a year by quarters). Headcount could be adjusted at the start of each quarter.

Project network structure is not always known a priori. For example, Fridsma (Fri03) and Cain (Cain03) investigated work in hospitals, where a patient's test results and initial response to treatment determine the subsequent treatment procedures. The EVPP does not address uncertainty in project network structure.

The EVPP assumes that projects in a portfolio can be selected independently. In practice, there are constraints that require projects to be performed only in certain combinations.

Second, the value ordering heuristic used to improve the efficiency of discrete search in the EVPP optimizer may be applicable to other optimization problems. I believe the heuristic is best suited to problems where correct choices for many variables are required to achieve a global optimum, but there are few rigid constraints among those variables.

Third, the EVPP optimizer could be integrated with Bijan KHosraviani's work in optimizing full VDT models. I designed the EVPP model to be compatible with core VDT work model. However, VDT also models decision wait time, organizational structure, and organizational communication policies, and these features are not addressed by the EVPP optimizer. Micro allocation issues (e.g. two engineers who will not together), micro scheduling issues, (e.g. employee family commitments), and employee learning issues are also important in developing an optimal schedule and staffing profile. The diversity of candidate constraints and objective functions in the VDT model suggests using a genetic programming approach to optimize it. The solution of the EVPP would be used as a guide for the genetic programming population. The EVPP solution would constrain the project schedule and staffing profile sufficiently to allow crossover to succeed in optimizing the other organizational variables.

Finally, the EVPP optimizer could be used as a tool in developing project management and organization theory. The EVPP captures more strategic factors in engineering management than previous models. Hence, organizations should be willing to follow the recommendations of the EVPP optimizer. If not, why don't they? For example, are there political actions by organizational actors, deliberate inefficiencies in the work process, or significant deficiencies in the EVPP model? Using the EVPP to develop a more accurate model of organizational behavior and performance would facilitate the development of the next generation of EVPP optimizer.

8 Conclusion

This section reviews the contributions of my research conducted under Grants No. 9980109 and 9907403. I believe that my research into the Engineering Vice-President's Problem has made four contributions to engineering management science and to optimization science. I have

1. recognized that real-world engineering management problems have too much flexibility to be represented and solved using existing constraint-based scheduling approaches. Constraint based scheduling, particularly in capital intensive industries, is like packing wooden blocks in a trunk. Engineering management problems are more like packing water balloons in a duffel bag. Because the EVPP is a different problem, it requires a different approach for its solution.
2. developed a mathematical formulation that allows a natural expression of the interactions, options, and constraints in engineering management. This formulation can be solved with a combination of linear programming (LP) and discrete search.
3. shown how to improve the effectiveness of discrete search in this formulation. The binding of a single variable should be made within the context of projected bindings for all of the other variables. This improvement is essential to obtaining high quality EVPP solutions quickly.
4. written a solver that has solved an academic EVPP test case efficiently. The solver can produce a good answer to this test case in seconds and can produce better answers given more time.

9 References

Alvarez-Valdéz, R. and J.M. Tamarit (1989), Heuristic Algorithms for Resource-Constrained Project Scheduling, in Slowinski, R. and J. Weglarz (Eds.), *Advances in Project Scheduling*, Elsevier, The Netherlands, 3-28.

Baptitste, P., C. Le Pape, and W. Nuijten (2001), *Constraint Based Scheduling: Applying Constraint Programming to Scheduling Problems*, Kluwer Academic Publishers, Norwell, Massachusetts.

Brooks, G.H. and C.R. White (1965), An Algorithm for Finding Optimal or Near Optimal Solutions to the Production Scheduling Problem, *Journal of Industrial Engineering*, January-February Issue, 34-40.

Cain, C.C. (2003), Ph.D. Thesis, Stanford University.

Carlier, J. and E. Pinson (1990), A Practical Use of Jackson's Preemptive Schedule For Solving the Job-Shop Problem, *Annals of Operations Research*, 26, 269-287.

Cooper, D.F. (1976), Heuristics for Scheduling Resource Constrained Projects: An Experimental Investigation, *Management Science*, 22, 1186-1194.

Dantzig, G.B., (1963), *Linear Programming and Extensions*, Princeton University Press, New Jersey.

Demeulemeester, E.L. (1995), Minimizing Resource-Availability Costs in Time-Limited Project Networks, *Management Science*, 41, 1590-1598.

- Demeulemeester, E.L. and W.S. Herroelen (2002), *Project Scheduling A Research Handbook*, Kluwer Academic Publishers, Norwell, Massachusetts.
- De Reyck, B. and W. Herroelen (1998), An Optimal Procedure for the Resource-Constrained Project Scheduling Problem with Discounted Cash Flows and Generalized Precedence Relations, *Computers and Operations Research*, 25, 1-17.
- De Reyck, B. and W. Herroelen (1999), The Multi-Mode Resource-Constrained Project Scheduling Problem with Generalized Precedence Relations, *European Journal of Operational Research*, 119 538-556.
- Elmaghraby, W.E.E. and J. Kamburowski (1992), The Analysis of Activity Networks under Generalized Precedence Relations (GPRs), *Management Science*, 38, 1245-1263.
- Fox, M.S. (1983), Constraint-Directed Search: A Case Study of Job Shop Scheduling, Ph.D. Thesis, Carnegie Mellon University, CMU-RI-TR-85-7, Intelligent Systems Laboratory, The Robotics Institute.
- Fridsma, D. (2003), Ph.D. Thesis, Stanford University
- Fulkerson, D.R., (1961), A Network Flow Computation for Project Cost Curves, *Management Science*, 7, 167-178.
- Garey, M.R. and D.S. Johnson (1979), *Computers and Intractability - Guide to the Theory of NP – Completeness*, W.H. Freeman & Company, San Francisco.
- Ghhasemzadeh, M., N. Archer, and P. Iyogun (1999), A Zero-One Model for Project Portfolio Selection and Scheduling, *Journal of the Operational Research Society*, 50, 745-755.
- Glover, F. (1989), Tabu Search, Part I, *ORSA Journal on Computing*, 1, 190-206.
- Goldratt, E.M. (1997), *Critical Chain*, The North River Press, Great Barrington, NJ.
- Hans, E.W. (2001), Resource Loading by Branch-and-Price Techniques, Ph.D. Thesis, University of Twente, www.tup.utwente.nl/uk/catalogue/technical/resource-loading .
- Hartmann, S. (1998), A Competitive Genetic Algorithm for Resource-Constrained Project Scheduling, *Naval Research Logistics*, 45, 733-750.
- Harvey, W.D and M.L. Ginsberg (1995), Limited Discrepancy Search, Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, 607-615.
- Holland, J.H. (1975), *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor.

- Kaplan, L.A. (1988), Resource-Constrained Project Scheduling with Preemption of Jobs, Ph.D. Thesis, University of Michigan.
- Kelley, J.E. Jr. (1961), Critical Path Planning and Scheduling: Mathematical Basis, *Operations Research*, 9, 296-320.
- Kelley, J.E. Jr. (1963), The Critical-Path Method: Resources Planning and Scheduling, in Muth, J.F. and G.L. Thompson (Eds.), *Industrial Scheduling*, Prentice Hall, Englewood Cliffs, 347-365.
- Kolisch, R. and A. Sprecher (1996), PSPLIB – A Project Scheduling Library, *European Journal of Operational Research*, 96, 205-216, <http://www.bwl.uni-kiel.de/Prod/psplib/index.html> .
- Kirpatrick, S., C.D. Gelatt and M.P. Vecchi (1983), Optimization by Simulated Annealing, *Science*, 220, 671-680.
- Land, A.H. and A.G. Doig (1960), An Automatic Method for Solving Discrete Programming Problems, *Econometrica*, 28, 497-520.
- Lawrence, S. (1985), Resource Constrained Project Scheduling – A Computational Comparison of Heuristic Scheduling Techniques, Technical Report, Graduate School of Industrial Administration, Carnegie Mellon University.
- Le Pape, C., P. Couronné , D. Gergamini and V. Gosselin (1994), Time-versus-Capacity Compromises in Project Scheduling, in Proceedings 13th Workshop of the UK Planning Special Interest Group.
- Levitt, R.E., G.P. Cohen, J.C. Kunz, C.I. Nass, T. Christiansen, and Y. Jin (1994), The Virtual Design Team: Simulating How Organization Structure and Information Processing Tools Affect Team Performance, in Carley, K.M. and M.J. Prietula (Eds.), *Computational Organization Theory*, Lawrence Erlbaum Associates, Hillsdale, NJ.
- Mackworth, A.K. (1977), Consistency in Networks of Relations, *Artificial Intelligence*, 8, 99-118.
- Moder, J.J., C.R. Phillips and E.W. Davis (1983), *Project Management with CPM, PERT, and Precedence Diagramming*, Van Nostrand Reinhold Company, Third Edition.
- Patterson, J.H., (1989), An Algorithm for a General Class of Precedence and Resource Constrained Scheduling Problems, in Slowinski, R. and J. Weglarz (Eds.), *Advances in Project Scheduling*, Elsevier, The Netherlands, 3-28.
- Pinson, E., C. Prins and F. Rullier (1994), Using Tabu Search for Solving the Resource-Constrained Project Scheduling Problem, Proceedings of the Fourth International Workshop on Project Management and Scheduling, Leuven, 102-106.

Pritsker, A.A.B., L.J. Waters and P.M. Wolfe (1969), Multi Project Scheduling with Limited Resources: A Zero-One Programming Approach, *Management Science*, 16, 93-108.

Sadeh, N.M.(1991), Look-ahead Techniques for Micro-Opportunistic Job Shop Scheduling. CMU-CS-91-102, School of Computer Science, Carnegie Mellon University.

Salazar-Kish, J.M. (2001), Modeling Concurrency Tradeoff and Their Effects on Project Duration and Rework, Ph.D Thesis, Stanford University.

Sampson, S.E. and E.N. Weiss (1993), Local Search Techniques for the Generalized Project Scheduling Problem, *Naval Research Logistics*, 40, 665-675.

Smith, S.F. and Cheng, C-C. (1993), Slack-Based Heuristics for Constraint Satisfaction Scheduling, Proceedings 11th National Conference on Artificial Intelligence, 139-144.

Stinson, J.P., E.W. Davis, and B.M. Khumawala (1978), Multiple Resource-Constrained Scheduling Using Branch-and-Bound, *AIIE Transactions*, 10, 252-259.

Talbot, F.B. (1982), Resource-Constrained Project Scheduling Problem with Time-Resource Tradeoffs: The Nonpreemptive Case, *Management Science*, 28, 1197-1210.

Vanhoucke, M., E. Demeulemeester and W. Herroelen (2000), A New Random Generator for Activity-on-the-Node Networks, Proceedings of the Seventh International Workshop on Project Management and Scheduling, Osnabrück, Germany, 290-293,
http://www.projectmanagement.ugent.be/Vanhoucke_Mario.htm