ON THE COMPLEXITY OF SCHEDULING UNIVERSITY COURSES

A Thesis

Presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

By

April Lin Lovelace

March 2010

COMMITTEE MEMBERSHIP

TITLE:	On the Complexity of Scheduling University Courses
AUTHOR:	April Lin Lovelace
DATE SUBMITTED:	March 2010
COMMITTEE CHAIR:	Lois Brady Professor Emeritus
COMMITTEE MEMBER:	Gene Fisher, Professor
COMMITTEE MEMBER:	Hasmik Gharibyan, Professor

ABSTRACT

On the Complexity of Scheduling University Courses

April Lin Lovelace

It has often been said that the problem of creating timetables for scheduling university courses is hard, even as hard as solving an NP-Complete problem. There are many papers in the literature that make this assertion but rarely are precise problem definitions provided and no papers were found which offered proofs that the university course scheduling problem being discussed is NP-Complete.

This thesis defines a scheduling problem that has realistic constraints. It schedules professors to sections of courses they are willing to teach at times when they are available without overloading them. Both decision and optimization versions are precisely defined. An algorithm is then provided which solves the optimization problem in polynomial time. From this it is concluded that the decision problem is unlikely to be NP-Complete because indeed it is in P.

A second more complex timetable design problem, that additionally seeks to assign appropriate rooms in which the professors can teach the courses, is then introduced. Here too both decision and optimization versions are defined. The second major contribution of this thesis is to prove that this decision problem is NP-Complete and hence the corresponding optimization problem is NP-Hard.

TABLE OF CONTENTS

LIST OF FIGURES

Figure ……..Page

1. INTRODUCTION

It has often been said that the problem of creating timetables for scheduling university courses is hard, even as hard as solving an NP-Complete problem. There are many different scheduling problems presented in the literature because every university has its own set of constraints and requirements. Most scheduling problems presented in the literature are modifications of the well known Timetable Design (TTD)[1] problem which schedules craftsmen to perform tasks. This problem was proven to be NP-Complete by Even et al.[2] (1976). Many papers make the assertion that their problem is as hard as an NP-Complete problem, or even that the problem presented is actually NP-Complete, simply based on similarity between the problem being discussed and the TTD problem. But this is an inappropriate assertion to make unless the problem under discussion is proved to be NP-Complete because, as this thesis will show, not every "modification" to an NP-Complete problem is itself NP-Complete.

The point of departure for this thesis is that same Timetable Design (TTD) problem. The TTD along with all the scheduling problems discussed in the thesis, are defined in section 1.1, then section 1.2 discusses the prior literature. In Chapter 2 the problem definitions presented in this chapter are explained in plain English and examples are provided. Polynomial time algorithms are then provided in Chapter 3 for both the Basic Course Scheduling (BCS) Optimization and BCS Decision problems. Chapter 4 goes on to prove that the Extended Course Scheduling Decision problem defined herein is NP-Complete and that the ECS Optimization problem is NP-Hard. Chapter 5 concludes the

---

[1] M. Garey, and D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness (New York: W.H. Freeman, 1979): 243

[2] S. Even , A. Itai, and A. Shamir, "On the Complexity of Timetable and Multicommodity Flow Problems." Siam Journal of Computing Vol. 5 (1976) 691-703

thesis with a summary and suggestions as to how to attack the ECS Optimization problem now that it is known to be NP-Hard and offers suggestions for future work.

1.1 Some Scheduling Problems

1.1.1 Timetable Design

The Timetable Design (TTD) problem appears in Garey and Johnson's Computers and Intractability[3], a well known collection of NP-Complete problems and is copied below in Figure 1. All definitions found in this thesis are stated in the format of problem definitions from Garey and Johnson[4], although only TTD is actually in that collection.

INSTANCE:
- a set H of "work periods"
- a set C of "craftsmen"
- a set T of "tasks"
- a set $A(c) \subseteq H$ of "available hours" for each craftsmen $c \in C$
- a set $A(t) \subseteq H$ of "available hours" for each task $t \in T$
- for each pair $(c, t) \in C \times T$, a number $R(c, t) \in Z_0^+$ "of required work periods"

QUESTION:
Is there a timetable for completing all tasks, i.e. a functions $f: C \times T \times H \rightarrow \{0, 1\}$ (where $f(c, t, h)=1$ means that craftsmen c works on task t during period h) that satisfies the following constraints:

1) $f(c, t, h) = 1$ only if $h \in A(c) \cap A(t)$
2) for each $h \in$ H and $c \in C$ there is at most one $t \in$ T for which $f(c, t, h) = 1$
3) for each $h \in$ H and $t \in T$ there is at most one $c \in$ C for which $f(c, t, h) = 1$
4) for each pair $(c, t) \in C \times T$ there are exactly $R(c, t)$ values of h for which $f(c, t, h) = 1$

Figure 1 Timetable Design Problem – Craftsmen and Tasks

As with all NP-Complete problems, TTD is a Decision problem and, as such, asks for a *yes*/*no* answer to the question, "Is there a timetable for completing all tasks that satisfies the given constraints?"

This problem is very similar to the problem of creating timetables for scheduling professors and courses at a university as well as other educational institutions. Simply replacing craftsmen and tasks with professors and courses in the TTD problem leads to

[3] Garey and Johnson, 243
[4] Ibid.

the problem given in Figure 2. This problem, called Timetable Design – Professors and Courses (TTD-P&C), is simply a restatement of the TTD problem using different terminology with professors and courses having the role of craftsman and tasks, and is therefore NP-Complete. The expression "craftsman *c* works on task *t* during period *h*" is replaced by "professor *p* teaches a section of course *c* during hour *h*." This problem has the basic elements involved with creating timetables for scheduling courses at a university but it is not realistic for most universities.

INSTANCE:
- a set H of "hours"
- a set P of "professors"
- a set C of "courses"
- a set $A(p) \subseteq H$ of "available hours" for each professor $p \in P$
- a set $A(c) \subseteq H$ of "available hours" for each course $c \in C$
- for each pair $(p, c) \in P \times C$, a number $R(p, c) \in Z_0^+$ "required sections"

QUESTION:
Is there a timetable that schedules all required sections of each course, i.e. a functions $f: P \times C \times H \rightarrow \{0, 1\}$ (where $f(p, c, h) = 1$ means that professor p teaches a section of course c during hour h) that satisfies the following constraints:

5) $f(p, c, h) = 1$ only if $h \in A(p) \cap A(c)$
6) for each $h \in H$ and $p \in P$ there is at most one $c \in C$ for which $f(p, c, h) = 1$
7) for each $h \in H$ and $c \in C$ there is at most one $p \in P$ for which $f(p, c, h) = 1$
8) for each pair $(p, c) \in P \times C$ there are exactly $R(p, c)$ values of h for which $f(p, c, h) = 1$

Figure 2 Timetable Design Problem – Professors and Courses

It should be noted that in the TTD problem defined in Garey and Johnson the set *H* is referred to as "work periods" in one place and "hours" in another[5], it seems clear that the original paper intended these "hours" to be non-overlapping work periods. Henceforth in this thesis, for convenience, the term, "hours," will be used rather than the longer, "non-overlapping work or academic periods." However the results are valid if the non-overlapping work periods are periods of time other than actual hours. Also, in most educational institutions, whenever a course is being offered more than once it is termed a

[5] Garey and Johnson, 243

"section," i.e. a course that needs to be taught *x* number of times is said to have *x* sections that need to be scheduled.

1.1.2 Basic Course Scheduling

A more realistic university scheduling problem would include upper bounds on professor loads and would also have a desired number of sections for each course. Additionally it would have a way to insure that professors are only assigned to teach courses they are willing and able to teach. Figure 3 defines this more realistic university scheduling problem precisely. In this thesis this problem is called the Basic Course Scheduling (BCS) Decision problem. In Chapter 2, the meaning of each of the bulleted items in the instance and each of the numbered items in the question are discussed in more detail, less formally and examples are given.

In the real world situations discussed above, it seems unnecessary to require that "no course has more than one professor assigned to it at any given hour" as in constraint three of the TTD-P&C because it is possible in the real world for two sections of the same course to be taught at the same hour, by different professors of course, so constraint three from TTD-P&C has been omitted. The question now becomes, "Is there timetable that schedules all desired sections while respecting the revised constraints?" The changes in the instance and the constraints are somewhat subtle but the changes in the results are very significant.

INSTANCE:

- a set H of "hours"
- a set P of "professors"
- a set C of "courses"
- a set $A(p) \subseteq H$ of "available hours" for each professor $p \in P$
- a set $A(c) \subseteq H$ of "available hours" for each course $c \in C$
- †a number $L(p) \in Z_0^+$ for each $p \in P$, the maximum number of sections p can teach
- †a number $S(c) \in Z_0^+$ for each $c \in C$, the desired number of sections for course c
- †a function $P \times C \rightarrow$ {true, false}, $WTT(p, c)$, the Willing To Teach function, where $WTT(p, c)$ = true if and only if p is willing to teach c

QUESTION:

Is there a timetable which schedules all desired sections, i.e. a function $f: P \times C \times H \rightarrow \{0, 1\}$, (where $f(p, c, h) = 1$ means professor p teaches a section of course c during hour h) that schedules all that desired sections and satisfies the following constraints:

1) $f(p, c, h) = 1$ only if $h \in A(p) \cap A(c)$
2) for each pair $h \in H$ and $p \in P$ there is at most one $c \in C$ for which $f(p, c, h) = 1$
3) *for each $p \in P$, $c \in C$ and $h \in H$ $f(p, c, h) = 1$ only if $WTT(p, c)$ = true
4) *for each $p \in P$, $\Sigma_{c, h} f(p, c, h) \leq L(p)$
5) *for each $c \in C$, $\Sigma_{p, h} f(p, c, h) \leq S(c)$

† *Note the last three bullets in the BCS Decision problem instance together replace the last bullet in the TTB-P&C problem instance*

* *Note criteria 3, 4 and 5 of the BCS Decision problem question together replace criteria 4 of TTD-P&C problem question*

Figure 3 Basic Course Scheduling Decision Problem

Like most decision problems there is an optimization version corresponding to the BCS Decision problem. It is called the BCS Optimization problem and is presented below in Figure 4.

As the reader can see, the optimization problem (Figure 4) does not change the inputs or constraints of the original decision problem (Figure 3) in any way (i.e. there is a direct mapping between the two instances and the constraints). Instead, it transforms the decision question "Is there a timetable which schedules all desired sections …" into an optimization goal "Find a timetable that maximizes the number of sections scheduled …" In this way every instance of the BCS Decision problem has a corresponding instance of the BCS Optimization problem.

At most educational institutions that schedule courses the goal is not to simply know whether or not all the desired sections can be scheduled but rather to find a timetable that schedules the maximum number of sections possible while respecting the criteria specified. Such a maximum timetable may schedule all sections desired but if that is not possible it schedules the most that can be scheduled while still respecting all the constraints.

INSTANCE:

- a set H of "hours"
- a set P of "professors"
- a set C of "courses"
- a set $A(p) \subseteq H$ of "available hours" for each professor $p \in P$
- a set $A(c) \subseteq H$ of "available hours" for each course $c \in C$
- a number $L(p) \in Z_0^+$ for each $p \in P$, the maximum number of sections p can teach
- a number $S(c) \in Z_0^+$ for each $c \in C$, the desired number of sections for course c
- a function $P \times C \rightarrow \{\text{true, false}\}$, $WTT(p, c)$, the Willing To Teach function, where $WTT(p, c)$ = true if and only if p is willing to teach c otherwise it is false

GOAL:

Find a timetable that maximizes the number of sections scheduled, i.e. a function $f: P \times C \times H \rightarrow \{0, 1\}$ which maximizes $\Sigma_{p, c, h} f(p, c, h)$ (where $f(p, c, h) = 1$ means professor p teaches a section of course c during hour h) that satisfies the following constraints:

1) $f(p, c, h) = 1$ only if $h \in A(p) \cap A(c)$
2) for each pair $h \in$ H and $p \in P$ there is at most one $c \in$ C for which $f(p, c, h) = 1$
3) for each $p \in P$, $c \in C$ and $h \in$ H $f(p, c, h) = 1$ only if $WTT(p, c)$ = true
4) for each $p \in P$, $\Sigma_{c, h} f(p, c, h) \leq L(p)$
5) for each $c \in C$, $\Sigma_{p, h} f(p, c, h) \leq S(c)$

Figure 4 Basic Course Scheduling Optimization Problem

In this section it has been shown how the Basic Course Scheduling problems have evolved from the well know NP-Complete problem called the Timetable Design problem. BCS Optimization problem is of practical interest because it closely fits the criteria for scheduling courses at many educational institutions. From the first moment of specifying the BCS Decision problem this thesis writer and her supervisor were interested in the question of whether or not this decision problem is NP-Complete. One important original work of this thesis is the creation of a polynomial time algorithm for solving the BCS

Optimization problem and then using that result and an additional comparison that is polynomial in the length of the input to solve the corresponding BCS Decision problem. This is the subject of Chapter 3. Thus in theoretical terms the BCS Decision problem is herein proved to be in the class P. Therefore most Computer Scientists would believe that it is not in the class NP-Complete even though the related TTD problem is in NP-Complete. A review of the problem classes including P, NP and NP-Complete along with a discussion of computational complexity can be found in Appendix A.

1.1.3 Extended Course Scheduling

In many course scheduling situations it is not only necessary to assign professors to teach courses at a specified times but also to assign rooms in which the professors are to teach those courses. This problem is called the Extended Course Scheduling (ECS) Decision problem in this thesis and is stated below in Figure 5. Note that it extends BCS Decision problem by adding the following parameters to an instance: a set, R, of rooms, a set of available hours $A(r)$ for each room, a bound $U(r)$ for each room, and the Room Suitability function $RS(c, r)$.

INSTANCE

- a set H of "hours"
- a set P of "professors"
- a set C of "courses"
- a set R of "rooms"
- a set $A(p) \subseteq H$ of "available hours" for each professor $p \in P$
- a set of $A(c) \subseteq H$ of "available hours" for each course $c \in C$
- a set of $A(r) \subseteq H$ of "available hours" for each room $r \in R$
- a number $L(p) \in Z_0^+$ for each $p \in P$, the maximum number of sections p can teach
- a number $S(c) \in Z_0^+$ for each $c \in C$, the desired number of sections for course c
- a number $U(r) \in Z_0^+$ for each $r \in R$, the maximum number of sections that can be taught in room r
- a function $P \times C \rightarrow$ {true, false}, $WTT(p, c)$, the Willing To Teach function, where $WTT(p, c)$ = true if and only if p is willing to teach c
- a function $C \times R \rightarrow$ {true, false}, $RS(c, r)$ the Room Suitability (RS) function, where $RS(c, r)$ = true if and only if r is a suitable room in which to teach c

QUESTION

Is there a timetable that schedules all desired sections of each course, i.e. a function $f: P \times C \times H \times R \rightarrow \{0, 1\}$ (where $f(p, c, h, r) = 1$ means that professor p teaches a section of course c in room r during hour h) that schedules all that desired sections and satisfies the following constraints:

Availability Constraints:

1) $f(p, c, h, r) = 1$ only if $h \in A(p) \cap A(c) \cap A(r)$]

Physical Constraints:

2) for each pair $(p,h) \in H \times P$ there is at most one pair $(c,r) \in C \times R$ for which $f(p,c,h,r) = 1$
3) for each pair $(r,h) \in R \times H$ there is at most one pair $(p,c) \in P \times C$ for which $f(p,c,h,r) = 1$

Academic Constraints:

4) for each $p \in P$, $c \in C$, $r \in R$ and $h \in H$, $f(p, c, h, r) = 1$ only if $WTT(p, c)$ = true
5) for each $p \in P$, $c \in C$, $r \in R$ and $h \in H$, $f(p, c, h, r) = 1$ only if $RS(c, r)$ = true

Bounding Constraints:

6) for each $p \in P$, $\Sigma_{c, h, r} f(p, c, h, r) \leq L(p)$
7) for each $c \in C$, $\Sigma_{p, h, r} f(p, c, h, r) \leq S(c)$
8) for each $r \in R$, $\Sigma_{p, c, h} f(p, c, h, r) \leq U(r)$

Figure 5 Extended Course Scheduling Decision Problem

The second important complexity result of this thesis is a proof that the ECS Decision problem shown above in Figure 5 is in NP-Complete. This will be proven in Chapter 4 (along with Appendix A which will prove the ECS Decision is in NP), and hence the ECS Optimization (see Figure 6) is NP-Hard.

INSTANCE

- a set H of "hours"
- a set P of "professors"
- a set C of "courses"
- a set R of "rooms"
- a set $A(p) \subseteq H$ of "available hours" for each professor $p \in P$
- a set of $A(c) \subseteq H$ of "available hours" for each course $c \in C$
- a set of $A(r) \subseteq H$ of "available hours" for each room $r \in R$
- a number $L(p) \in Z_0^+$ for each $p \in P$, the maximum number of sections p can teach
- a number $S(c) \in Z_0^+$ for each $c \in C$, the desired number of sections for course c
- a number $U(r) \in Z_0^+$ for each $r \in R$, the maximum number of sections that can be taught in room r
- a function $P \times C \rightarrow$ {true, false}, $WTT(p, c)$, the Willing To Teach function, where $WTT(p, c)$ = true if and only if p is willing to teach c
- a function $C \times R \rightarrow$ {true, false}, $RS(c, r)$ the Room Suitability (RS) function, where $RS(c, r)$ = true if and only if r is a suitable room in which to teach c

GOAL

Find a timetable that maximizes the number of sections scheduled, i.e. a function $f: P \times C \times H \times R \rightarrow \{0, 1\}$ which maximizes $\Sigma_{p, c, h, r} f(p, c, h, r)$ (where $f(p, c, h, r) = 1$ means professor p teaches a section of course c in room r at hour h) that satisfies the following constraints:

Availability Constraints:

1) $f(p, c, h, r) = 1$ only if $h \in [A(p) \cap A(c) \cap A(r)]$

Physical Constraints:

2) for each pair $(p,h) \in H \times P$ there is at most one pair $(c,r) \in C \times R$ for which $f(p,c,h,r) = 1$
3) for each pair $(r,h) \in R \times H$ there is at most one pair $(p,c) \in P \times C$ for which $f(p,c,h,r) = 1$

Academic Constraints:

4) for each $p \in P$, $c \in C$, $r \in R$ and $h \in H$, $f(p, c, h, r) = 1$ only if $WTT(p, c)$ = true
5) for each $p \in P$, $c \in C$, $r \in R$ and $h \in H$, $f(p, c, h, r) = 1$ only if $RS(c, r)$ = true

Bounding Constraints:

6) for each $p \in P$, $\Sigma_{c, h, r} f(p, c, h, r) \leq L(p)$
7) for each $c \in C$, $\Sigma_{p, h, r} f(p, c, h, r) \leq S(c)$
8) for each $r \in R$, $\Sigma_{p, c, h} f(p, c, h, r) \leq U(r)$

Figure 6 Extended Course Scheduling Optimization Problem

In this section it has been shown how the Extended Course Scheduling problems have also evolved from the well know NP-Complete problem called the Timetable Design problem. ECS Optimization problem is of practical interest because it closely fits the criteria for scheduling courses at many educational institutions. Again this thesis writer and her supervisor were interested in the question of whether or not this ECS

Decision problem is NP-Complete. Another important original work of this thesis is the creation of the proof that the ECS Decision problem is NP-Complete; this is the subject of Chapter 4.

1.2 Related Work

Although it is often stated informally that the problem of scheduling university courses is in NP-Complete, nothing was found in the literature that both defined the problem precisely and provided a proof of the claim.

The literature on Course Scheduling Problems is quite extensive. Schaerf [6] offers an excellent survey of all the existing approaches to timetabling, as of 1999. The research performed in conjunction with this work yielded well over a thousand references which seem to fall into two categories: the first type of papers discuss algorithmic approaches to specialized situations to create a timetable. In this type of research there is often little or no discussion regarding complexity or optimality of the given algorithm. By far the majority of research papers fall into this first category. The second types are papers which discuss complexity of the problem but they do so exclusively and in generalities, none introduces a specific algorithm and analyzes its complexity.

Most of the papers which introduce algorithms are of limited research interest as they present particular algorithms used by a particular institution for their specific purpose and do not discuss the complexity of either the problem or of their algorithm. Some rare few discuss general approaches into automating the course scheduling problem but very few of these even acknowledged the difficulty of the problem, only touching upon the fact that the course scheduling problem is related to an NP-Complete problem.

---

[6] A Schaerf, "A Survey of Automated Timetabling." Artificial Intelligence Review Vol. 13, No. 2 (1999) 87-127

Furthermore, the algorithms in the literature do not consider the fact that indeed there may be no timetable that meets the given constraints.

There have been various attempts to formulate an automated system for scheduling using various methods; the most recent approaches to the scheduling problem involve randomized local search heuristics. Schaerf and Di Gaspero [7] provide a clear introduction to the various randomized local search heuristic algorithms but they do not give a comparative analysis, nor do they explain that the main reason these heuristics approaches are so popular is because there is no known polynomial time algorithm which will produce exact answers to an NP-Complete problem.

The network flow algorithm used herein for the BCS problem was discovered independently in this thesis but afterwards two papers were found which apply network flows to the course scheduling problem. Dinkel et. al., [8] presents an interactive program which utilizes a network flow methodology but it requires human intervention during the process, as such, it is not a fully automated algorithm such as the ones presented here. Dyer and Mulvey [9] present an automatic solution but it is very limited in scope, only scheduling courses to professors. Neither work specifies the network flow algorithm used nor was the complexity of their solution addressed.

---

[7] A. Schaerf and L. Di Gaspero, "Local Search Techniques for Educational Timetabling Problem." Proceeding of the 6th International Symposium on Operational Research in Slovenia (2001) 13-23

[8] J. Dinkel, J. Mote, M. Venkataramanan. "An Efficient Decision Support System for Academic Course Scheduling." Operations Research, Vol. 37, No. 6 (1989): 853-864

[9] J. Dyer, and J. Mulvey. "An Integrated Optimization/Information System for Academic Departmental Planning." Management Science, Vol. 22, No. 12, (1976): 1332-1341

2. EXAMPLES AND DISCUSSION

Here more explanation of the problems defined in Chapter 1 is given as well as examples of the problems. The reader who clearly understood the definitions in Chapter 1 may skip this chapter. In section 2.1 the original NP-Complete TTD problem is discussed in detail while the TDD-P&C version is discussed in section 2.2. The two BCS problems, both decision and optimization, are presented along with examples in sections 2.3 and 2.4 respectively. Finally in sections 2.5 and 2.6 discussions of the ECS Decision and Optimization problems are given and examples provided.

2.1 The Timetable Design Problem

The TTD problem is NP-Complete[1], therefore is no known algorithm with computational complexity polynomial in the size of the input for solving it. The TTD problem was given in Figure 1 and is repeated here for the reader's convenience.

INSTANCE:

- a set H of "work periods"
- a set C of "craftsmen"
- a set T of "tasks"
- a set $A(c) \subseteq H$ of "available hours" for each craftsmen $c \in C$
- a set $A(t) \subseteq H$ of "available hours" for each task $t \in T$
- for each pair $(c, t) \in C \times T$, a number $R(c, t) \in Z_0^+$ "of required work periods"

QUESTION:

Is there a timetable for completing all tasks, i.e. a functions $f: C \times T \times H \rightarrow \{0, 1\}$ (where $f(c, t, h)=1$ means that craftsmen c works on task t during period h) that satisfies the following constraints:

1) $f(c, t, h) = 1$ only if $h \in A(c) \cap A(t)$
2) for each $h \in H$ and $c \in C$ there is at most one $t \in T$ for which $f(c, t, h) = 1$
3) for each $h \in H$ and $t \in T$ there is at most one $c \in C$ for which $f(c, t, h) = 1$
4) for each pair $(c, t) \in C \times T$ there are exactly $R(c, t)$ values of h for which $f(c, t, h) = 1$

Copy of Figure 1 Timetable Design Problem – Craftsmen and Tasks

An instance of TTD problem consists of three sets, H, C, and T, two collection of sets, A(c) for each $c \in C$ and A(t) for each $t \in T$, which are self-explanatory, and a

[1] Even, Itai, and Shamir

function, “of required work periods,” where R(c, t) is a function indicating the number of work periods that craftsman c must work on task t.

In the TTD problem a timetable for completing the tasks is referred to as a function $f: C \times T \times H \rightarrow \{0, 1\}$ where $f(c, t, h) = 1$ means craftsman c works on task t during work period h. Alternately a timetable could be represented by the set $S = \{(c, t, h): f(c, t, h) = 1\}$, S is a subset of $C \times T \times H$ and the function f is called the characteristic function of the set S by mathematicians. A third way to represent a schedule is with a matrix, where the craftsmen are listed along the side, the tasks are along the top with the work periods inserted in the appropriate $C \times T$ cell are those for which the craftsman c works on task t. All three ways of representing a timetable are shown in Figure 7.

The constraints in Figure 1 are quite cryptic and mathematical. Here their effects are presented in plain English:

1) Assures that a craftsman is assigned to work on a task only during a period when both the craftsman and the task are available.
2) Assures that no craftsman is assigned to work more on than one task at any given work period.
3) Assures that no task has more then one craftsman assigned to work on it at any given work period.
4) Assures that the number of assigned work periods for each pair of craftsman c and task t is the required number according to the function $R(c, t)$.

Two examples of the TTD problem are given below along with their answers. In Figure 7 TTD Instance #1 the answer is *yes*, and two different witnesses are given to verify this. A witness serves to demonstrate that the answer is indeed *yes*. When

showing a witness as a function, for brevity, elements of $C \times T \times H$ for which $f(c, t, h) = 0$ are not listed. Witness 2 serves to demonstrate that it sometimes happens that two or more timetables satisfy all the constraints of a given problem instance.

Set of Work Periods, Craftsman and Tasks

$H = \{h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9, h_{10}\}$
$C = \{c_1, c_2, c_3\}$
$T = \{t_1, t_2, t_3\}$

Availability of Craftsman

$A(c_1) = \{h_1, h_2, h_5, h_6, h_9, h_{10}\}$
$A(c_2) = \{h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9, h_{10}\}$
$A(c_3) = \{h_3, h_4, h_7, h_8, h_9\}$

Availability of Tasks

$A(t_1) = \{h_2, h_5, h_6, h_7, h_{10}\}$
$A(t_2) = \{h_3, h_4, h_5, h_8, h_{10}\}$
$A(t_3) = (h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9, h_{10}\}$

Required Number of Work Periods

$R(c, t) =$

	t_1	t_2	t_3
c_1	2	1	0
c_2	0	2	2
c_3	1	1	1

Answer = Yes

Witness 1 (Characteristic Function)

$f(c_1, t_1, h_2) = 1$, $f(c_1, t_1, h_6) = 1$,
$f(c_1, t_2, h_5) = 1$, $f(c_2, t_2, h_8) = 1$,
$f(c_2, t_2, h_{10}) = 1$, $f(c_2, t_3, h_1) = 1$,
$f(c_2, t_3, h_7) = 1$, $f(c_3, t_1, h_3) = 1$,
$f(c_3, t_2, h_4) = 1$, $f(c_3, t_3, h_9) = 1$
$f(c, t, h) = 0$ for all (c, t, h) not listed

Witness 1 (Matrix)

	t_1	t_2	t_3
c_1	h_2, h_6	h_5	
c_2		h_8, h_{10}	h_1, h_7
c_3	h_3	h_4	h_9

Witness 2 (Characteristic Function)

$f(c_1, t_1, h_2) = 1$, $f(c_1, t_1, h_6) = 1$,
$f(c_1, t_2, h_{10}) = 1$, $f(c_2, t_2, h_8) = 1$,
$f(c_2, t_2, h_5) = 1$, $f(c_2, t_3, h_1) = 1$,
$f(c_2, t_3, h_9) = 1$, $f(c_3, t_1, h_3) = 1$,
$f(c_3, t_2, h_4) = 1$, $f(c_3, t_3, h_7) = 1$
$f(c, t, h) = 0$ for all (c, t, h) not listed

Witness 2 (Set Notation)

$S = \{(c_1, t_1, h_2), (c_1, t_1, h_6), (c_1, t_2, h_{10}), (c_2, t_2, h_8), (c_2, t_2, h_5), (c_2, t_3, h_1), (c_2, t_3, h_9), (c_3, t_1, h_3), (c_3, t_2, h_4), (c_3, t_3, h_7)\}$
$(c, t, h) \notin S$ for all $f(c, t, h) = 0$

Figure 7 TTD Instance #1

Below in Figure 8 there is no timetable which satisfies the given constraints. This can be verified if desired by checking that all the 2^{3*3*10} functions from $C \times T \times H$ to $\{0, 1\}$ fail to satisfy the constraints.

Set of Work Periods, Craftsman and Tasks

$H = \{h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9, h_{10}\}$
$C = \{c_1, c_2, c_3\}$
$T = \{t_1, t_2, t_3\}$

Availability of Craftsman

$A(c_1) = \{h_1, h_2, h_5, h_6, h_9\}$
$A(c_2) = \{h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9, h_{10}\}$
$A(c_3) = \{h_1, h_2, h_5, h_7, h_{10}\}$

Availability of Tasks

$A(t_1) = \{h_1, h_2, h_3, h_5, h_8\}$
$A(t_2) = \{h_3, h_4, h_5, h_8, h_{10}\}$
$A(t_3) = (h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9, h_{10}\}$

Required Number of Work Periods

$R(c, t) =$

	t_1	t_2	t_3
c_1	2	1	0
c_2	0	2	2
c_3	1	1	1

Answer = No

Figure 8 TTD Instance #2

2.2 Timetable Design - Professors and Courses

TTD-P&C is repeated below in Figure 2. Note that this is an exact translation of TTD (only the notation and terminology has been changed). Thus the problem is also NP-Complete.

INSTANCE:
- a set H of "hours"
- a set P of "professors"
- a set C of "courses"
- a set $A(p) \subseteq H$ of "available hours" for each professor $p \in P$
- a set $A(c) \subseteq H$ of "available hours" for each course $c \in C$
- for each pair $(p, c) \in P \times C$, a number $R(p, c) \in Z_0^+$ "required sections"

QUESTION:
Is there a timetable that schedules all required sections of each course, i.e. a functions $f: P \times C \times H \rightarrow \{0, 1\}$ (where $f(p, c, h) = 1$ means that professor p teaches a section of course c during period h) that satisfies the following constraints:

1) $f(p, c, h) = 1$ only if $h \in A(p) \cap A(c)$
2) for each $h \in H$ and $p \in P$ there is at most one $c \in C$ for which $f(p, c, h) = 1$
3) for each $h \in H$ and $c \in C$ there is at most one $p \in P$ for which $f(p, c, h) = 1$
4) for each pair $(p, c) \in P \times C$ there are exactly $R(p, c)$ values of h for which $f(p, c, h) = 1$

Copy of Figure 2 Timetable Design Problem – Professors and Courses

As in the original TTD problem, an instance of TTD-P&C problem consists of three sets, H, P, and C, two collection of sets, $A(p)$ for each $p \in P$ and $A(c)$ for each $c \in C$, again these are self-explanatory. The last input is the "required sections" where $R(p, c)$ is a function indicating the number of sections professor p must teach. The question of

TTD-P&C problem expresses the timetable function $f: P \times C \times H \rightarrow \{0, 1\}$. Alternately the timetable can be expressed as a set S, where $S \subseteq P \times C \times H$ such that $(p, c, h) \in$ S if and only if $f(p, c, h) = 1$, f is the characteristic function S. The constraints in Figure 2 have the following effects:

1) Assures that a professor teaches a section of course only during an hour when both the professor and the course are available.
2) Assures that no professor is teaching more then one course at the given hour. (Please note that f is a function to {0, 1}. This, together with constraint 2, guarantees that no professor is assigned to teach more than one section of the same or different course simultaneously.)
3) Assures that no course has more then one professor assigned to teach it at any given hour.
4) Assures that the number of sections for each pair of professor p and course c are the required number as assigned by $R(p, c)$.

The two examples of TTD-P&C given below are translations of the two TTD problem examples given in section 2.1. They are presented to demonstrate the notation and terminology of the course scheduling problems in this thesis.

Set of Hours, Professors and Courses

$H = \{h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9, h_{10}\}$
$P = \{p_1, p_2, p_3\}$
$C = \{c_1, c_2, c_3\}$

Availability of Professors

$A(p_1) = \{h_1, h_2, h_5, h_6, h_9, h_{10}\}$
$A(p_2) = \{h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9, h_{10}\}$
$A(p_3) = \{h_3, h_4, h_7, h_8, h_9\}$

Availability of Courses

$A(c_1) = \{h_2, h_5, h_6, h_7, h_{10}\}$
$A(c_2) = \{h_3, h_4, h_5, h_8, h_{10}\}$
$A(c_3) = (h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9, h_{10}\}$

Required number of Sections

$R(p, c) =$

	c_1	c_2	c_3
p_1	2	1	0
p_2	0	2	2
p_3	1	1	1

Answer = Yes

Witness 1 Matrix Form

	c_1	c_2	c_3
p_1	h_2, h_6	h_5	
p_2		h_8, h_{10}	h_1, h_7
p_3	h_3	h_4	h_9

Witness 2 Matrix Form

	c_1	c_2	c_3
p_1	h_2, h_6	h_{10}	
p_2		h_8, h_5	h_1, h_9
p_3	h_3	h_4	h_7

Figure 9 TTD-P&C Instance #1

For brevity both witnesses for the example in Figure 9 TTD-P&C Instance #1 are shown in matrix form only. Figure 10 TTD-P&C Instance #2 shows a problem instance whose answer is *no*.

Set of Hours, Professors, and Courses

$H = \{h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9, h_{10}\}$
$P = \{p_1, p_2, p_3\}$
$C = \{c_1, c_2, c_3\}$

Availability of Professors

$A(p_1) = \{h_1, h_2, h_5, h_6, h_9\}$
$A(p_2) = \{h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9, h_{10}\}$
$A(p_3) = \{h_1, h_2, h_5, h_7, h_{10}\}$

Availability of Courses

$A(c_1) = \{h_1, h_2, h_3, h_5, h_8\}$
$A(c_2) = \{h_3, h_4, h_5, h_8, h_{10}\}$
$A(c_3) = (h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9, h_{10}\}$

Required number of Sections

$R(p, c) =$

	c_1	c_2	c_3
p_1	2	1	0
p_2	0	2	2
p_3	1	1	1

Answer = No

Figure 10 TTD-P&C Instance #2

2.3 Basic Course Scheduling Decision Problem

As stated in Chapter 1, the BCS Decision problem is a modification of TTD-P&C which defines more realistic constraints for educational institutions. Rather than professors having specified numbers of "required sections" ($R(p, c)$) there is a Willing To

Teach function (*WTT: P*×*C* → {true, false}) which is a function indicating whether or not a professor is willing (and able) to teach a particular course. No one, least of all the students, wants a professor to teach a course she is unwilling (or unable) to teach. In addition to the Willing To Teach (*WTT*) function each professor has an upper bound, $L(p)$, on the number of sections she can be scheduled to teach. Finally there is a desired number of sections, $S(c)$, for each course. Figure 3 below provides a repeat of the definition of the BCS Decision problem which was originally presented in section 1.1.2.

INSTANCE:

- a set H of "hours"
- a set P of "professors"
- a set C of "courses"
- a set $A(p) \subseteq H$ of "available hours" for each professor $p \in P$
- a set $A(c) \subseteq H$ of "available hours" for each course $c \in C$
- †a number $L(p) \in Z_0^+$ for each $p \in P$, the maximum number of sections p can teach
- †a number $S(c) \in Z_0^+$ for each $c \in C$, the desired number of sections for course c
- †a function $P \times C \rightarrow$ {true, false}, $WTT(p, c)$, the Willing To Teach function, where $WTT(p, c)$ = true if and only if p is willing to teach c

QUESTION:

Is there a timetable which schedules all desired sections, i.e. a function $f: P \times C \times H \rightarrow \{0, 1\}$, (where $f(p, c, h) = 1$ means professor p teaches a section of course c during period h) that schedules all that desired sections and satisfies the following constraints:

1) $f(p, c, h) = 1$ only if $h \in A(p) \cap A(c)$
2) for each pair $h \in H$ and $p \in P$ there is at most one $c \in C$ for which $f(p, c, h) = 1$
3) *for each $p \in P$, $c \in C$ and $h \in H$ $f(p, c, h) = 1$ only if $WTT(p, c)$ = true
4) *for each $p \in P$, $\Sigma_{c, h} f(p, c, h) \leq L(p)$
5) *for each $c \in C$, $\Sigma_{p, h} f(p, c, h) \leq S(c)$

† *Note the last three bullets in the BCS Decision problem instance together replace the last bullet in the TTB-P&C problem instance*

* *Note criteria 3, 4 and 5 of the BCS Decision problem question together replace criteria 4 of TTD-P&C problem question*

Copy of Figure 3 Basic Course Scheduling Decision Problem

An instance of BCS problem consists of three sets, H, P, and C, two collection of sets, $A(p)$ for each $p \in P$ and $A(c)$ for each $c \in C$, all of which are self-explanatory. The sixth and seventh inputs are the bounds for the professors and courses and finally the last input is the "Willing to Teach" function where $WTT(p, c)$ is *true* if professor p is willing

(and able) to teach course *c* and *false* otherwise. The question for the BCS problem expresses the timetable function $f: P \times C \times H \rightarrow \{0, 1\}$. This function can also be represented as a set S, where $S \subseteq P \times C \times H$ such that $(p, c, h) \in$ S if and only if $f(p, c, h) = 1$, or as a matrix. In addition to the change in the instance that replaces the function $R(p,c)$ with the function $WTT(p, c)$ and adds the bounding functions $L(p)$ and $S(c)$ there are also several constraint changes from the TTD-P&C. Constraints 3, 4, and 5 are the new constraints that involve the WTT and the bounds $L(p)$ and $S(c)$.

The constraints in Figure 3 above have the following effects:

1) Assures that a professor teaches a section of course only during an hour when both the professor and the course are available.
2) Assures that no professor is assigned to teach more than one course during any given hour. (This constraint coupled with the fact that the function $f(p, c, h)$ maps to $\{0, 1\}$ ensures that no professor is assigned to more than one section of the same, or different courses, during any given hour.)
3) Assures that the professor is willing to teach each course he is scheduled to teach.
4) Assures that no professor is assigned to teach more sections than his load bound allows.
5) Assures that no course is being taught more than the desired number of sections.

Two examples of BCS-Decision are given below. These examples use data similar to the data used in the examples for both TTD and TTD-P&C but have a *WTT*() function

and the appropriate upper bounds instead of the "required number of sections." Some *yes*/*no* answers are also different.

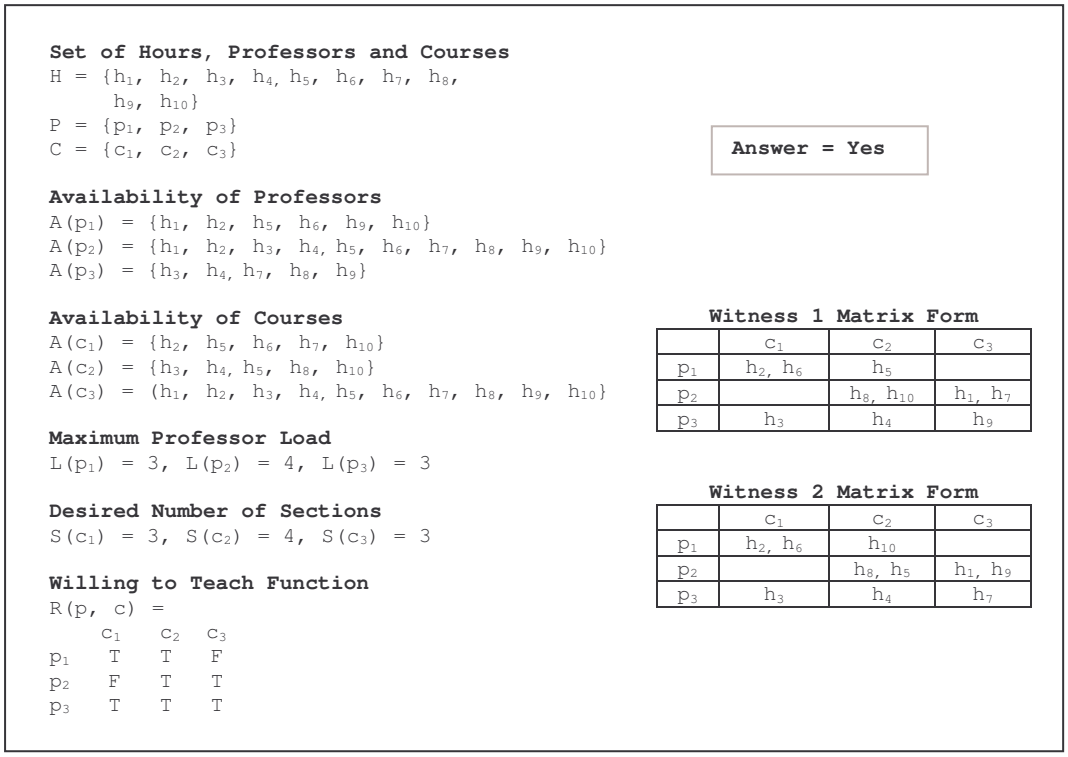

Figure 11 BCS-Decision Instance #1

Figure 11 BCS-Decision Instance #1 shows the two upper bounds functions "Maximum Professor Load" and "Desired Number of Sections" as well as the "Willing to Teach" function which maps to {true, false}. All these together replace the "Required to Number of Sections" from section 2.2.

Set of Hours, Professors and Courses

$H = \{h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9, h_{10}\}$
$P = \{p_1, p_2, p_3\}$
$C = \{c_1, c_2, c_3\}$

Availability of Professors

$A(p_1) = \{h_1, h_2, h_5, h_6, h_9\}$
$A(p_2) = \{h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9, h_{10}\}$
$A(p_3) = \{h_1, h_2, h_5, h_7, h_{10}\}$

Availability of Courses

$A(c_1) = \{h_1, h_2, h_3, h_5, h_8\}$
$A(c_2) = \{h_3, h_4, h_5, h_8, h_{10}\}$
$A(c_3) = (h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9, h_{10}\}$

Maximum Professor Load

$S(c_1) = 3$, $S(c_2) = 4$, $S(c_3) = 3$,

Desired Number of Sections

$A(p_1) = 4$, $A(p_2) = 3$, $A(p_3) = 3$

Willing to Teach Function

$R(p, c) =$

	c_1	c_2	c_3
p_1	T	T	F
p_2	F	T	T
p_3	T	T	T

Answer = Yes

Witness 1

	c_1	c_2	c_3
p_1	h_1, h_2	h_5	
p_2		h_3, h_4	h_6, h_8
p_3	h_1	h_{10}	h_7

Witness 2

	c_1	c_2	c_3
p_1	h_1, h_2	h_5	
p_2		h_4, h_8	h_7, h_5
p_3	h_1	h_{10}	h_7

Figure 12 BCS-Decision Instance #2

Figure 12 BCS-Decision Instance #2, shown above, has an answer of *yes* while the similar Figure 10 TTD-P&C Instance #2, has an answer of *no*. This is a direct result of removing constraint three of TTD-Professors and Courses which assured "no course has more then one professor assigned to it at any given time." As we can clearly see in Witness 1 two different sections of course $c1$ are being taught at $h1$ one by $p1$ and the other by $p3$. In Witness 2 in addition to two sections of course $c1$ being taught at $h1$ we also have two different sections of course $c3$ being taught by at $h7$, by different professors of course.

2.4 Basic Course Scheduling Optimization Problem

Below in Figure 4 the formal definition for the BCS Optimization problem is repeated.

INSTANCE:

- a set H of "hours"
- a set P of "professors"
- a set C of "courses"
- a set $A(p) \subseteq H$ of "available hours" for each professor $p \in P$
- a set $A(c) \subseteq H$ of "available hours" for each course $c \in C$
- a number $L(p) \in Z_0^+$ for each $p \in P$, the maximum number of sections p can teach
- a number $S(c) \in Z_0^+$ for each $c \in C$, the desired number of sections for course c
- a function $P \times C \rightarrow$ {true, false}, $WTT(p, c)$, the Willing To Teach function, where $WTT(p, c)$ = true if and only if p is willing to teach c otherwise it is false

GOAL:

Find a timetable that maximizes the number of sections scheduled, i.e. a function $f: P \times C \times H \rightarrow \{0, 1\}$ which maximizes $\Sigma_{p, c, h} f(p, c, h)$ (where $f(p, c, h) = 1$ means professor p teaches a section of course c during period h) that satisfies the following constraints:

1) $f(p, c, h) = 1$ only if $h \in A(p) \cap A(c)$
2) for each pair $h \in$ H and $p \in P$ there is at most one $c \in$ C for which $f(p, c, h) = 1$
3) for each $p \in P$, $c \in C$ and $h \in$ H $f(p, c, h) = 1$ only if $WTT(p, c)$ = true
4) for each $p \in P$, $\Sigma_{c, h} f(p, c, h) \leq L(p)$
5) for each $c \in C$, $\Sigma_{p, h} f(p, c, h) \leq S(c)$

Copy of Figure 4 Basic Course Scheduling Optimization Problem

As the reader can see, the instance sets and constraints for BCS Optimization problem are exactly the same as they are for the Corresponding BCS Decision problem. The difference is that instead of a question the BCS Optimization problem has a goal which is to schedule the maximum number of section possible while respecting the contracts.

Below in Figure 13 and Figure 15 are two examples based on the two BCS Decision problem instances from the previous section. The most obvious difference is that the solution is not a *yes*/*no* answer but an optimal timetable. For BCS Optimization Instance #1 in Figure 13 below an optimal timetable is shown in the three possible forms. The number of sections scheduled is not part of the answer sought but is included here for interest.

Set of Hours, Professors and Courses

$H = \{h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9, h_{10}\}$
$P = \{p_1, p_2, p_3\}$
$C = \{c_1, c_2, c_3\}$

Availability of Professors

$A(p_1) = \{h_1, h_2, h_5, h_6, h_9, h_{10}\}$
$A(p_2) = \{h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9, h_{10}\}$
$A(p_3) = \{h_3, h_4, h_7, h_8, h_9\}$

Availability of Courses

$A(c_1) = \{h_2, h_5, h_6, h_7, h_{10}\}$
$A(c_2) = \{h_3, h_4, h_5, h_8, h_{10}\}$
$A(c_3) = (h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9, h_{10}\}$

Professor Loads

$L(p_1) = 3,\ l(p_2) = 4,\ L(p_3) = 3$

Desired Number of Sections

$S(c_1) = 3,\ S(c_2) = 4,\ S(c_3) = 3,$

Willing to Teach Function

WTT(p, c) =

	c_1	c_2	c_3
p_1	T	T	F
p_2	F	T	T
p_3	T	T	T

An Optimal Timetable (Characteristic Function)

$f(p_1, c_1, h_2) = 1,\ f(p_1, c_1, h_5) = 1,$
$f(p_1, c_2, h_5) = 1,\ f(p_2, c_2, h_8) = 1,$
$f(p_2, c_2, h_{10}) = 1,\ f(p_2, c_3, h_1) = 1,$
$f(p_2, c_3, h_7) = 1,\ f(p_3, c_1, h_3) = 1,$
$f(p_3, c_2, h_4) = 1,\ f(p_3, c_3, h_9) = 1$

$f(c, t, h) = 0$ for all (c, t, h) not listed

An Optimal Timetable (Matrix)

	c_1	c_2	c_3
p_1	h_2, h_6	h_5	
p_2		h_8, h_{10}	h_1, h_7
p_3	h_3	h_4	h_9

An Optimal Timetable (Set Notation)

$S = \{(p_1, c_1, h_2), (p_1, c_1, h_5), (p_1, c_2, h_5), (p_2, c_2, h_8), (p_2, c_2, h_{10}), (p_2, c_3, h_1), (p_2, c_3, h_7), (p_3, c_1, h_3), (p_3, c_2, h_4), (p_3, c_3, h_9)\}$

$(c, t, h) \notin$ for all $f(c, t, h) = 0$

Optimal Number of Sections Scheduled = 10 of 10

Figure 13 BCS Optimization Instance #1

Because the optimal timetable shown in Figure 13 has scheduled all ten sections, we can easily say that the answer to the corresponding BCS-Decision Instance #1 is *yes* using this timetable as a witness. The timetable given in Figure 13 is an optimal timetable with all ten sections being scheduled but is not the only optimal answer. Just as there can be multiple witnesses to a decision problem there can be multiple solutions to an optimization problem. Below in Figure 14 is another possible answer to BCS Optimization Instance #1 given above.

Another Optimal Timetable

	c_1	c_2	c_3
p_1	h_2, h_6	h_{10}	
p_2		h_8, h_5	h_1, h_9
p_3	h_3	h_4	h_7

Optimal Number of Sections Scheduled = 10 of 10

Figure 14 Second Solution for BCS Optimization Instance #1

Is one of the optimal timetables better then the other? That can only be determined by the institution creating the timetable, according to the given institution's own additional criteria.

For Figure 15 BCS Optimization Instance #2 a maximum timetable for the problem schedules only nine sections even though ten sections were desired. This means the answer to corresponding BSC Decision problem would be *no*.

Set of Hours, Professors and Courses

$H = \{h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9, h_{10}\}$

$P = \{p_1, p_2, p_3\}$

$C = \{c_1, c_2, c_3\}$

Availability of Professors

$A(p_1) = \{h_1, h_2, h_5, h_6, h_9\}$

$A(p_2) = \{h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9, h_{10}\}$

$A(p_3) = \{h_4, h_6, h_7, h_9, h_{10}\}$

Availability of Courses

$A(c_1) = \{h_1, h_2, h_3, h_5, h_8\}$

$A(c_2) = \{h_3, h_4, h_5, h_8, h_{10}\}$

$A(c_3) = \{h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9, h_{10}\}$

Professor Loads

$L(p_1) = 3, L(p_2) = 4, L(p_3) = 3$

Desired Number of Sections

$S(c_1) = 4, S(c_2) = 3, S(c_3) = 3$

Willing to Teach Function

WTT(p, c) =

	c_1	c_2	c_3
p_1	T	T	F
p_2	F	T	T
p_3	T	F	T

An Optimal Timetable =

	c_1	c_2	c_9
P_1	h_1, h_2	h_5	
p_2		h_3, h_4	h_8, h_{10}
p_3		h_{10}	h_7

Optimal Number of Sections Scheduled = 9 of 10

Figure 15 BCS Optimization Instance #2

2.5 Extended Course Scheduling Decision Problem

As stated in Chapter 1, the ECS Decision is an extension of the BCS Decision problem that includes the scheduling of rooms in addition to courses, professors and hours. In this section the ECS Decision problem is discussed and examples are given. The formal definition is repeated below for convenience.

INSTANCE

- a set H of "hours"
- a set P of "professors"
- a set C of "courses"
- a set R of "rooms"
- a set $A(p) \subseteq H$ of "available hours" for each professor $p \in P$
- a set of $A(c) \subseteq H$ of "available hours" for each course $c \in C$
- a set of $A(r) \subseteq H$ of "available hours" for each room $r \in R$
- a number $L(p) \in Z_0^+$ for each $p \in P$, the maximum number of sections p can teach
- a number $S(c) \in Z_0^+$ for each $c \in C$, the desired number of sections for course c
- a number $U(r) \in Z_0^+$ for each $r \in R$, the maximum number of sections that can be taught in room r
- a function $P \times C \rightarrow$ {true, false}, $WTT(p, c)$, the Willing To Teach function, where $WTT(p, c)$ = true if and only if p is willing to teach c
- a function $C \times R \rightarrow$ {true, false}, $RS(c, r)$ the Room Suitability (RS) function, where $RS(c, r)$ = true if and only if r is a suitable room in which to teach c

QUESTION

Is there a timetable that schedules all desired sections of each course, i.e. a function $f: P \times C \times H \times R \rightarrow \{0, 1\}$ (where $f(p, c, h, r) = 1$ means that professor p teaches a section of course c in room r during period h) that schedules all that desired sections and satisfies the following constraints:

Availability Constraints:

1) $f(p, c, h, r) = 1$ only if $h \in A(p) \cap A(c) \cap A(r)$]

Physical Constraints:

2) for each pair $(p,h) \in H \times P$ there is at most one pair $(c,r) \in C \times R$ for which $f(p,c,h,r) = 1$
3) for each pair $(r,h) \in R \times H$ there is at most one pair $(p,c) \in P \times C$ for which $f(p,c,h,r) = 1$

Academic Constraints:

4) for each $p \in P$, $c \in C$, $r \in R$ and $h \in H$, $f(p, c, h, r) = 1$ only if $WTT(p, c)$ = true
5) for each $p \in P$, $c \in C$, $r \in R$ and $h \in H$, $f(p, c, h, r) = 1$ only if $RS(c, r)$ = true

Bounding Constraints:

6) for each $p \in P$, $\Sigma_{c, h, r} f(p, c, h, r) \leq L(p)$
7) for each $c \in C$, $\Sigma_{p, h, r} f(p, c, h, r) \leq S(c)$
8) for each $r \in R$, $\Sigma_{p, c, h} f(p, c, h, r) \leq U(r)$

Copy of Figure 5 Extended Course Scheduling Decision Problem

An instance of ECS problem consists of four sets, H, P, C, and R three collections of sets, $A(p)$ for each $p \in P$, $A(c)$ for each $c \in C$ and $A(r)$ for each $r \in R$ all of which are self-explanatory. The next three inputs are bounds for the professors, courses and rooms. The last two inputs are the "Willing to Teach" function, WTT, where $WTT(p, c)$ is *true* if professor p is willing (and able) to teach course c and *false* otherwise, and the "Room

Suitability" function, RS, where $RS(c, r)$ is *true* if room r is suitable for teaching course c and *false* otherwise.

The question for the ECS Decision problem expresses the timetable as a function $f: P \times C \times H \times R \rightarrow \{0, 1\}$. As with the BCS Decision problem, this function can also be represented as a set S, where $S \subseteq P \times C \times H \times R$ such that $(p, c, h, r) \in S$ if and only if $f(p, c, h, r) = 1$, but because there are four variables this timetable cannot be easily expressed as a matrix.

The constraints have increased to eight in ECS problems from five in the BCS problems. These constraints, which are listed in Figure 5, have the following effects:

Availability Constraints:

1) Assures that a meet takes place only during an hour when the professor, course, and room are all available.

Physical Constraints:

2) Assures that no professor is assigned to teach more than one course during any given hour, nor in more than one room. (This constraint coupled with the fact that the function $f(p, c, h, r)$ maps to $\{0, 1\}$ ensures that no professor is assigned to more than one section of the same, or different courses, during any given hour.)
3) Assures that, at any given hour, no room has more than one professor assigned to teach in it nor does the room have more than one course being taught in it.

Academic Constraints:

4) Assures that each professor is willing to teach each course he is scheduled to teach.

5) Assures that each room is suitable for each course scheduled to be taught in it.

Bounding Constraints:

6) Assures that no professor is assigned to teach more sections than his load bound allows.

7) Assures that no course is being taught more than the desired number of sections.

8) Assures that no room is assigned more than the maximum allowable number of sections.

Below in Figure 16 is an example instance of the ECS Decision problem. Note that, the answer to the question, "Is there a timetable that schedules all sections?" is *no*.

Set of Hours, Professors and Courses

$H = \{h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9\}$
$P = \{p_1, p_2, p_3\}$
$C = \{c_1, c_2, c_3\}$
$R = \{r_1, r_2, r_3\}$

Availability of Professors

$A(p_1) = \{h_1, h_2, h_4, h_5, h_6, h_9\}$
$A(p_2) = \{h_1, h_2, h_3, h_4, h_7, h_8, h_9\}$
$A(p_3) = \{h_3, h_4, h_7, h_8, h_9\}$

Availability of Courses

$A(c_1) = \{h_2, h_5, h_6, h_7, h_9\}$
$A(c_2) = \{h_3, h_4, h_5, h_8, h_9\}$
$A(c_3) = \{h_1, h_4, h_5, h_6, h_8, h_9\}$

Availability of Rooms

$A(r_1) = \{h_1, h_3, h_5, h_7, h_9\}$
$A(r_2) = \{h_1, h_2, h_4, h_5, h_6, h_7, h_8\}$
$A(r_3) = \{h_3, h_4, h_5, h_6, h_7\}$

Answer = No

Professor Loads

$L(p_1) = 3, l(p_2) = 4, L(p_3) = 3$

Desired Number of Sections

$S(c_1) = 3, S(c_2) = 4, S(c_3) = 4$

Room Loads

$U(r_1) = 6, U(r_3) = 7, U(r_3) = 4$

Willing to Teach Function

WTT(p, c) =

	c_1	c_2	c_3
p_1	T	T	F
p_2	F	T	T
p_3	T	T	F

Room Suitability Function

RS(c, r) =

	r_1	r_2	r_3
c_1	F	T	T
c_2	T	T	F
c_3	T	F	T

Figure 16 ECS Decision Instance #1

The example in Figure 17 has an answer of *yes*, and a witness is provided

Set of Hours, Professors and Courses

$H = \{h_1, h_2, h_3, h_4\}$
$P = \{p_1, p_2, p_3\}$
$C = \{c_1, c_2, c_3\}$
$R = \{r_1, r_2, r_3\}$

Availability of Professors

$A(p_1) = \{h_1, h_2, h_3\}$
$A(p_2) = \{h_1, h_2, h_3, h_4\}$
$A(p_3) = \{h_2, h_3, h_4\}$

Availability of Courses

$A(c_1) = \{h_2, h_3, h_4\}$
$A(c_2) = \{h_1, h_2, h_3\}$
$A(c_3) = \{h_1, h_2, h_3, h_4\}$

Availability of Rooms

$A(r_1) = \{h_1, h_2, h_4\}$
$A(r_2) = \{h_1, h_2, h_3, h_4\}$
$A(r_3) = \{h_1, h_2, h_3, h_4\}$

Professor Loads

$L(p_1) = 3,\ l(p_2) = 3,\ L(p_3) = 2$

Desired Number of Sections

$S(c_1) = 2,\ S(c_2) = 3,\ S(c_3) = 3$

Room Loads

$U(r_1) = 3,\ U(r_3) = 3,\ U(r_3) = 4$

Willing to Teach Function

WTT(p, c) =

	c_1	c_2	c_3
p_1	T	T	T
p_2	T	T	F
p_3	F	T	T

Room Suitability Function

RS(c, r) =

	r_1	r_2	r_3
c_1	T	F	T
c_2	F	T	F
c_3	F	T	T

Answer = Yes

Witness =

$\{f(p_1, c_1, h_1, r_1) = 1,\ f(p_1, c_1, h_2, r_1) = 1,$
$f(p_2, c_1, h_4, r_1) = 1,\ f(p_2, c_2, h_1, r_2) = 1,$
$f(p_2, c_2, h_2, r_2) = 1,\ f(p_3, c_2, h_2, r_3) = 1,$
$f(p_1, c_3, h_3, r_3) = 1,\ f(p_3, c_3, h_4, r_3) = 1\}$
$f(c, t, h, r) = 0$ for all (c, t, h, r) not listed

Figure 17 ECS Decision Instance #2

2.6 Extended Course Scheduling Optimization Problem

Again, educational institutions do not to simply want know whether or not all the desired sections can be scheduled but rather they need an actual timetable that schedules the maximum number of sections possible (all sections preferably) while respecting the availability, physical, academic and bounding constraints as stated. Such a maximal timetable may schedule all sections desired but if that is not possible, it schedules the most that can be scheduled while still respecting all the constraints.

INSTANCE

- a set H of "hours"
- a set P of "professors"
- a set C of "courses"
- a set R of "rooms"
- a set $A(p) \subseteq H$ of "available hours" for each professor $p \in P$
- a set of $A(c) \subseteq H$ of "available hours" for each course $c \in C$
- a set of $A(r) \subseteq H$ of "available hours" for each room $r \in R$
- a number $L(p) \in Z_0^+$ for each $p \in P$, the maximum number of sections p can teach
- a number $S(c) \in Z_0^+$ for each $c \in C$, the desired number of sections for course c
- a number $U(r) \in Z_0^+$ for each $r \in R$, the maximum number of sections that can be taught in room r
- a function $P \times C \rightarrow$ {true, false}, $WTT(p, c)$, the Willing To Teach function, where $WTT(p, c)$ = true if and only if p is willing to teach c
- a function $C \times R \rightarrow$ {true, false}, $RS(c, r)$ the Room Suitability (RS) function, where $RS(c, r)$ = true if and only if r is a suitable room in which to teach c

GOAL

Find a timetable that maximizes the number of sections scheduled, i.e. a function $f: P \times C \times H \times R \rightarrow \{0, 1\}$ which maximizes $\Sigma_{p, c, h, r} f(p, c, h, r)$ (where $f(p, c, h, r) = 1$ means professor p teaches a section of course c in room r at hour h) that satisfies the following constraints:

Availability Constraints:

1) $f(p, c, h, r) = 1$ only if $h \in [A(p) \cap A(c) \cap A(r)]$

Physical Constraints:

2) for each pair $(p,h) \in H \times P$ there is at most one pair $(c,r) \in C \times R$ for which $f(p,c,h,r) = 1$
3) for each pair $(r,h) \in R \times H$ there is at most one pair $(p,c) \in P \times C$ for which $f(p,c,h,r) = 1$

Academic Constraints:

4) for each $p \in P$, $c \in C$, $r \in R$ and $h \in H$, $f(p, c, h, r) = 1$ only if $WTT(p, c)$ = true
5) for each $p \in P$, $c \in C$, $r \in R$ and $h \in H$, $f(p, c, h, r) = 1$ only if $RS(c, r)$ = true

Bounding Constraints:

6) for each $p \in P$, $\Sigma_{c, h, r} f(p, c, h, r) \leq L(p)$
7) for each $c \in C$, $\Sigma_{p, h, r} f(p, c, h, r) \leq S(c)$
8) for each $r \in R$, $\Sigma_{p, c, h} f(p, c, h, r) \leq U(r)$

Copy of Figure 6 Extended Course Scheduling Optimization Problem

As the reader can see, the instance sets and constraints for ECS Optimization problem are exactly the same as they are for the Corresponding ECS Decision problem. The difference is that instead of a question the ECS Optimization problem has a goal which is to find a timetable which schedules the maximum number of section possible while respecting the constraints.

Below, in Figure 18 and Figure 19, are two examples of ECS Optimization problem along with their optimal timetables. In Figure 18, the first timetable, Timetable 1, is shown in terms of the function $f(p, c, h, r)$ while the second solution, Timetable 2, is shown in as a set S. The Figure 18 ECS Optimization Instance #1corresponds to the ECS Decision Instance #1 presented in Figure 16 and although the answer to ECS Decision Instance #1 has an answer of *no* the optimization problem is does have an optimal timetable. It simply does not schedule all the sections desired.

Set of Hours, Professors and Courses
$H = \{h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9\}$
$P = \{p_1, p_2, p_3\}$
$C = \{c_1, c_2, c_3\}$
$R = \{r_1, r_2, r_3\}$

Availability of Professors
$A(p_1) = \{h_1, h_2, h_4, h_5, h_6, h_9\}$
$A(p_2) = \{h_1, h_2, h_3, h_4, h_7, h_8, h_9\}$
$A(p_3) = \{h_3, h_4, h_7, h_8, h_9\}$

Availability of Courses
$A(c_1) = \{h_2, h_5, h_6, h_7, h_9\}$
$A(c_2) = \{h_3, h_4, h_5, h_8, h_9\}$
$A(c_3) = \{h_1, h_4, h_5, h_6, h_8, h_9\}$

Availability of Rooms
$A(r_1) = \{h_1, h_3, h_5, h_7, h_9\}$
$A(r_2) = \{h_1, h_2, h_4, h_5, h_6, h_7, h_8\}$
$A(r_3) = \{h_3, h_4, h_5, h_6, h_7\}$

Professor Loads
$L(p_1) = 3,\ l(p_2) = 4,\ L(p_3) = 3$

Desired Number of Sections
$S(c_1) = 3,\ S(c_2) = 4,\ S(c_3) = 4$

Room Loads
$U(r_1) = 6,\ U(r_3) = 7,\ U(r_3) = 4$

Willing to Teach Function
WTT(p, c) =

	c_1	c_2	c_3
p_1	T	T	F
p_2	F	T	T
p_3	T	T	F

Room Suitability Function
RS(c, r) =

	r_1	r_2	r_3
c_1	F	T	T
c_2	T	T	F
c_3	T	F	T

Timetable 1 (Characteristic Function) =
$\{f(p_1, c_1, h_2, r_2) = 1$
$f(p_1, c_1, h_5, r_2) = 1,$
$f(p_1, c_1, h_6, r_2) = 1,$
$f(p_3, c_2, h_3, r_1) = 1,$
$f(p_3, c_2, h_4, r_2) = 1,$
$f(p_3, c_2, h_8, r_2) = 1,$
$f(p_3, c_2, h_9, r_1) = 1,$
$f(p_2, c_3, h_1, r_1) = 1,$
$f(p_2, c_3, h_4, r_3) = 1,$
$f(p_2, c_3, h_9, r_3) = 1\}$
$f(c, t, h, r) = 0$ for all
(c, t, h, r) not listed

Optimal Number of Sections Scheduled = 10 of 11

Timetable 2 (Set Notation) =
$S = \{(p_1, c_1, h_2, r_2),$
$(p_1, c_1, h_5, r_3),$
$(p_1, c_1, h_6, r_3),$
$(p_3, c_2, h_3, r_1),$
$(p_3, c_2, h_4, r_2),$
$(p_3, c_2, h_8, r_2),$
$(p_3, c_2, h_9, r_1),$
$(p_2, c_3, h_1, r_1),$
$(p_2, c_3, h_4, r_3),$
$(p_2, c_3, h_9, r_3)\}$
$(c, t, h, r) \notin S$ for all
$f(c, t, h, r) = 0$

Optimal Number of Sections Scheduled = 10 of 11

Figure 18 ECS Optimization Instance #1

For Figure 18 ECS Optimization Instance #1 a maximal timetable would include eleven sections; this is derived from the $S(c)$ (3+4+4). The maximum number of sections scheduled is bounded by a number of variables; the availability of courses, professors and/or rooms, the room loads and most significantly in this example is the professor loads. In this example an upper bound is imposed by the total number of sections that all the professors can teach (i.e. the load bound $L(p)$ which is 3+4+3 = 10). Thus an optimal timetable can contain no more than ten sections as is the case with the two solutions shown. Reaching the upper bound allows one to be confident that the timetable is indeed optimal.

Set of Hours, Professors and Courses
$H = \{h_1, h_2, h_3, h_4\}$
$P = \{p_1, p_2, p_3\}$
$C = \{c_1, c_2, c_3\}$
$R = \{r_1, r_2, r_3\}$

Availability of Professors
$A(p_1) = \{h_1, h_2, h_3\}$
$A(p_2) = \{h_1, h_2, h_3, h_4\}$
$A(p_3) = \{h_2, h_3, h_4\}$

Availability of Courses
$A(c_1) = \{h_2, h_3, h_4\}$
$A(c_2) = \{h_1, h_2, h_3\}$
$A(c_3) = \{h_1, h_2, h_3, h_4\}$

Availability of Rooms
$A(r_1) = \{h_1, h_2, h_4\}$
$A(r_2) = \{h_1, h_2, h_3, h_4\}$
$A(r_3) = \{h_1, h_2, h_3, h_4\}$

Professor Loads
$L(p_1) = 3,\ l(p_2) = 3,\ L(p_3) = 2$

Desired Number of Sections
$S(c_1) = 2,\ S(c_2) = 3,\ S(c_3) = 3$

Room Loads
$U(r_1) = 3,\ U(r_3) = 3,\ U(r_3) = 4$

Willing to Teach Function
WTT(p, c) =

	c_1	c_2	c_3
p_1	T	T	T
p_2	T	T	F
p_3	F	T	T

Room Suitability Function
RS(c, r) =

	r_1	r_2	r_3
c_1	T	F	T
c_2	F	T	F
c_3	F	T	T

Timetable (Characteristic Function) =
$\{f(p_1, c_1, h_1, r_1) = 1,\ f(p_1, c_1, h_2, r_1) = 1,$
$f(p_2, c_1, h_4, r_1) = 1,\ f(p_2, c_2, h_1, r_2) = 1,$
$f(p_2, c_2, h_2, r_2) = 1,\ f(p_3, c_2, h_2, r_3) = 1,$
$f(p_1, c_3, h_3, r_3) = 1,\ f(p_3, c_3, h_4, r_3) = 1\}$
$f(c, t, h, r) = 0$ for all (c, t, h, r) not listed

Maximal Number of Sections Scheduled = 8 of 8

Figure 19 ECS Optimization Instance #2

A maximal timetable is shown in Figure 19. Note that two sections of c_2 are schedule to be taught at h_2. There is no constraint that prevents this; in a university

setting multiple sections of the same course may be taught at the same hour. As long as the professors and the rooms are not double booked this is not constraint violation.

3. BCS DECISION PROBLEM IS IN P

This chapter presents an algorithm for solving the BCS Optimization problem that has computational complexity that is polynomial in the size of the inputs. This algorithm is original work of this thesis. It was implemented by the author and the result for the example shown in this chapter was produced by that implementation. The implementation was used to produce timetable solutions for other examples as well but those examples are not included in this thesis. This algorithm is used to build another algorithm that solves the BCS Decision Problem also in polynomial time. Figure 4, the definition of the BCS Optimization Problem, is repeated for the reader's convenience.

INSTANCE:

- a set H of "hours"
- a set P of "professors"
- a set C of "courses"
- a set $A(p) \subseteq H$ of "available hours" for each professor $p \in P$
- a set $A(c) \subseteq H$ of "available hours" for each course $c \in C$
- a number $L(p) \in Z_0^+$ for each $p \in P$, the maximum number of sections p can teach
- a number $S(c) \in Z_0^+$ for each $c \in C$, the desired number of sections for course c
- a function $P \times C \rightarrow \{\text{true, false}\}$, $WTT(p, c)$, the Willing To Teach function, where $WTT(p, c)$ = true if and only if p is willing to teach c otherwise it is false

GOAL:

Find a timetable that maximizes the number of sections scheduled, i.e. a function $f: P \times C \times H \rightarrow \{0, 1\}$ which maximizes $\Sigma_{p, c, h} f(p, c, h)$ (where $f(p, c, h) = 1$ means professor p teaches a section of course c during hour h) that satisfies the following constraints:

1) $f(p, c, h) = 1$ only if $h \in A(p) \cap A(c)$
2) for each pair $h \in$ H and $p \in P$ there is at most one $c \in$ C for which $f(p, c, h) = 1$
3) for each $p \in P$, $c \in C$ and $h \in$ H $f(p, c, h) = 1$ only if $WTT(p, c)$ = true
4) for each $p \in P$, $\Sigma_{c, h} f(p, c, h) \leq L(p)$
5) for each $c \in C$, $\Sigma_{p, h} f(p, c, h) \leq S(c)$

Copy of Figure 4 Basic Course Scheduling Optimization Problem

Section 3.1 describes the algorithm and provides an example to facilitate understanding. In section 3.2 the algorithm is proved to be correct, i.e. it is proved that the algorithm creates a timetable that meets all the constraints of the BCS Optimization problem as given in Figure 4 and is guaranteed to be optimal. The computational

complexity of the algorithm is computed and shown to be polynomial in the size of the inputs in section 3.3. Finally, in section 3.4, the algorithm for the BCS Optimization problem is used to prove the BCS Decision problem is in P.

3.1 Algorithm for BCS Optimization Problem

The algorithm presented here has three stages. In the first stage a flow network, G*, is created from the inputs of the BCS Optimization problem instance. The second stage uses the Edmonds-Karp Algorithm[1], a version of the Ford-Fulkerson Method[2], to find a maximum flow, F*, through the network created in stage one. Finally, in stage three an optimal timetable, *f**, for the given problem instance is created from the maximum flow found in stage two. The algorithm is illustrated using a particular instance of the BCS Optimization problem but the algorithm presented is entirely general and can be used with any instance of the BCS Optimization problem.

In stage one the problem inputs are used to create a flow network. A flow network is a directed graph in which each arc has a nonnegative capacity and in which one of the nodes is designated as the source node, and another as the sink node.

In particular, using the data of an instance, the algorithm creates a flow network, G*, with six levels of nodes. The nodes, respectively from left to right, are the source node; a set of nodes corresponding to the set of courses (C); a set of nodes corresponding to the set of Course × Hour ($C \times H$); a set of nodes corresponding to the set of Professor × Hour ($P \times H$); a set of nodes corresponding to the set of professors (P) and the sink node. Between successive levels of nodes there are sets of arcs.

---

[1] J. Edmonds and R Karp, "Theoretical Improvements in the Algorithmic Efficiency for Network Flow Problems," Journal of ACM Vol. 19 (1972), 248-264

[2] T. Cormen, C. Leiserson, R. Rivest and C. Stein, Introduction to Algorithms 2nd Edition (Cambrige. The MIT Press, 2001), 651-664

The particular instance used to illustrate the algorithm is shown in Figure 20 and the flow network created from it is shown in Figure 21.

Set of Hours, Professors and Courses
$H = \{h_1, h_2, h_3\}$
$P = \{p_1, p_2, p_3, p_4\}$
$C = \{c_1, c_2, c_3, c_4\}$

Availability of Professors
$A(p_1) = \{h_1, h_2, h_3\}$
$A(p_2) = \{h_1, h_2, h_3\}$
$A(p_3) = \{h_1, h_2, h_3\}$
$A(p_4) = \{h_2, h_3\}$

Availability of Courses
$A(c_1) = \{h_1, h_3\}$
$A(c_2) = \{h_1, h_2, h_3\}$
$A(c_3) = \{h_1, h_2, h_3\}$
$A(c_4) = \{h_1, h_2, h_3\}$

Desired Number of Sections
$S(c_1) = 2$, $S(c_2) = 3$,
$S(c_3) = 3$, $S(c_4) = 3$,

Maximum Professor Load
$L(p_1) = 2$, $L(p_2) = 3$,
$L(p_3) = 3$, $L(p_4) = 2$,

Willing to Teach Function
WTT(p, c) =

	c_1	c_2	c_3	c_4
p_1	T	T	F	T
p_2	T	F	T	T
p_3	F	T	T	T
p_4	T	T	T	T

Figure 20 BCS Instance: Inputs

The first set of arcs in G* is from the source to each *C* node, *c*. Each of these arcs has a capacity, *S*(*c*), representing the number of desired sections for course *c*. The capacity of the flow is indicated in Figure 21 by the "*S*(*c*)" at the bottom of the graph directly underneath the first set of arcs.

The second set of arcs in G* connects the *C* nodes to the *C*×*H* nodes for each matching *c*. Each of these arcs has a capacity of one if $h \in A(c)$ and a capacity of zero if $h \notin A(c)$. This is indicated by the "*c*=*c* & *h* in *A*(*c*) / 1, *c*=*c* & *h* not in *A*(*c*) / 0" at the bottom of the graph in Figure 21 directly underneath the second set of arcs.

There is a third set of arcs in G* connecting a *C*×*H* node to a *P*×*H* for each identical *h* if *WTT*(*p, c*) = *true* for the *c* and *p*, $h \in A(c)$, and $h \in A(p)$. In other words, there is an arc from a *C*×*H* node, (*c*, *h*), to a *P*×*H* node, (*p*, *c*), if professor *p* is willing to teach course *c* and both *p* and *c* are available at hour *h*. Each of these arcs has a capacity of one. This is indicated by the "*h*=*h* & *h* in *A*(*c*) & *h* in *A*(*p*) & *WTT*(*p*, *c*) = *true* / 1" at the bottom of the graph directly underneath the third set of arcs. This set of arcs is very

important for stage three when an optimal timetable is created from the maximum flow found in stage two.

The flow from the next set of arcs in G* leaves from the $P \times H$ nodes and enters the P nodes for which the p's correspond. These each have a capacity of one if $h \in A(p)$ and a capacity of zero if $h \notin A(p)$. This is indicated by the "$p=p$ & h in $A(p)$ / 1, $p=p$ & h not in $A(p)$ / 0" at the bottom of the graph in Figure 21 directly under the fourth set of arcs.

The last set of arcs in G* leading from the P nodes to the sink have a capacity of $L(p)$ representing the maximum number of sections each professor can teach. Again the capacity of the arcs is indicated in Figure 21 by the "$L(p)$" at the bottom of the graph directly underneath the last set of arcs.

Figure 21 below shows the G* corresponding to the problem instance given in Figure 20.

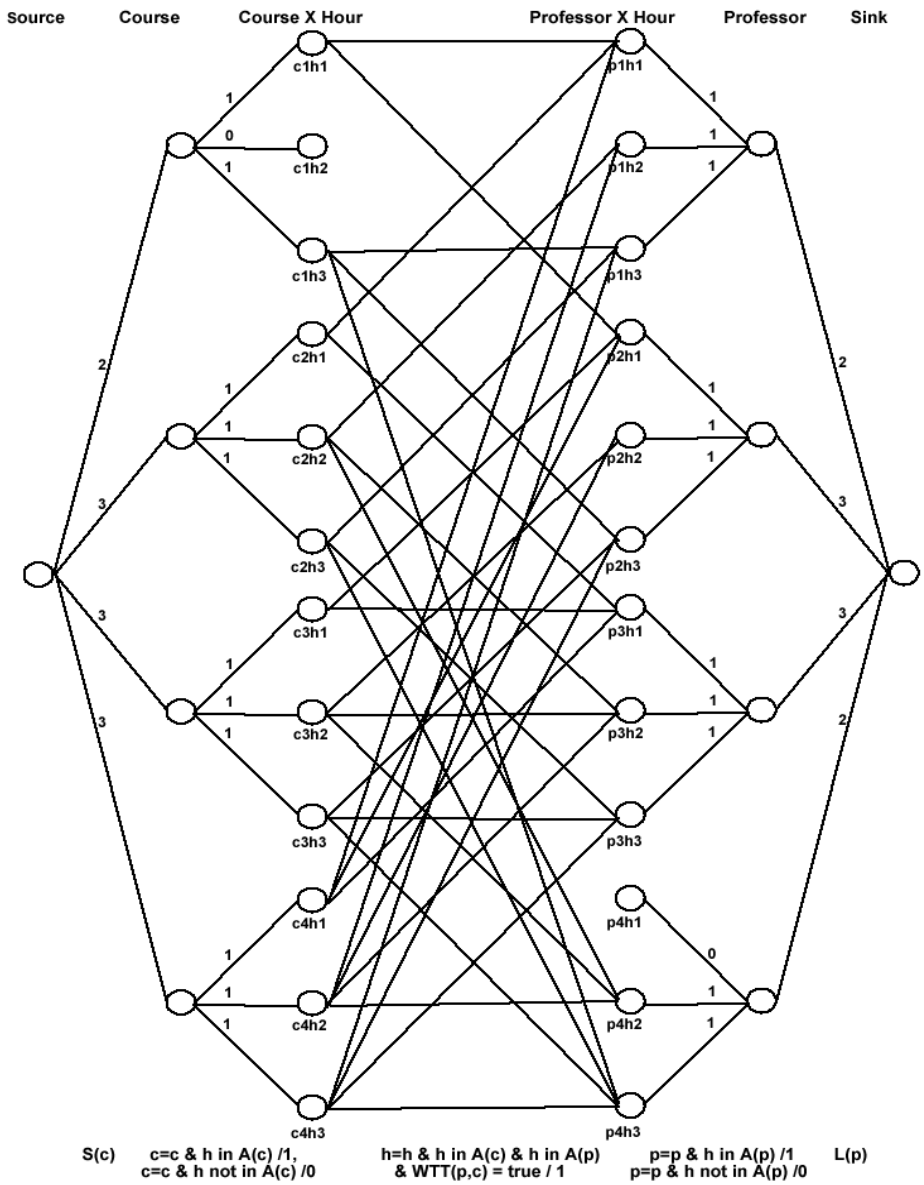

Figure 21 Flow Network, G*, for Figure 20

The second stage of this algorithm uses the Ford-Fulkerson Method (specifically the Edmonds-Karp algorithm) to determine a maximal flow from the source node to the sink node in the network, G*, constructed in stage one. The Ford-Fulkerson Method begins with a flow of zero (0) on all arcs and proceeds iteratively to increase the flow by finding augmenting paths. An augmenting path is a path from the source to the sink along which more flow can be sent. The Ford-Fulkerson Method also uses a *residual*

graph which keeps track of current remaining capacity for each arc while the *flow* graph keeps track of the current flow.

```
The Ford Fulkerson Method (G, s, t)
   /* G is a flow network        */
   /* s is the source node in G  */
   /* t is the sink node in G    */

   initialize residual flow R* := G
   initialize flow F* := G with 0 on all arcs

   while there exists an augmenting path p*
      augment flow F* along p*
      update residual flow R* along p*
   return F*
```

Figure 22 Ford-Fulkerson Method Pseudocode

The pseudocode for the Ford-Fulkerson Method is shown in Figure 22. With each iteration through the loop, as a new augmenting path is found, the *residual* and *flow* graphs are updated. This loop continues until there are no more augmenting paths. Edmonds-Karp algorithm uses a breath first search to find augmenting paths. This is very important for two reasons: first with a breath first search the loop is guaranteed to terminate and secondly, the computational complexity is polynomial in the size of the inputs[3]. Figure 23 shows the resulting maximal flow after stage two, the Edmonds-Karp algorithm, is complete for the network in Figure 21. The maximal flow is indicated by the bold arcs and the bold flow values on them. This result was produced by the implementation of the algorithm done for the thesis.

In stage three the maximal flow, F*, produced in stage two is used to create a timetable. In particular the existence, or not, of arcs between the $C \times H$ nodes and the $P \times H$ nodes and the flows on them are used. Create a timetable as a function $f^*: P \times C \times H \rightarrow \{0, 1\}$ as follows: $f^*(p, c, h) = 1$ if there exists an arc from (c, h) to (p, h) in the flow

[3]Cormen et al., 660-663

network, G*, of stage 1 and that arc has a flow of 1 in the maximum network flow, F*, in stage two, $f^*(p, c, h) = 0$ otherwise.

In section 3.2 the correctness of this three stage algorithm is proven. But first the trace of the algorithm for the example of Figure 20 that produced the flow network G* of Figure 21 is completed.

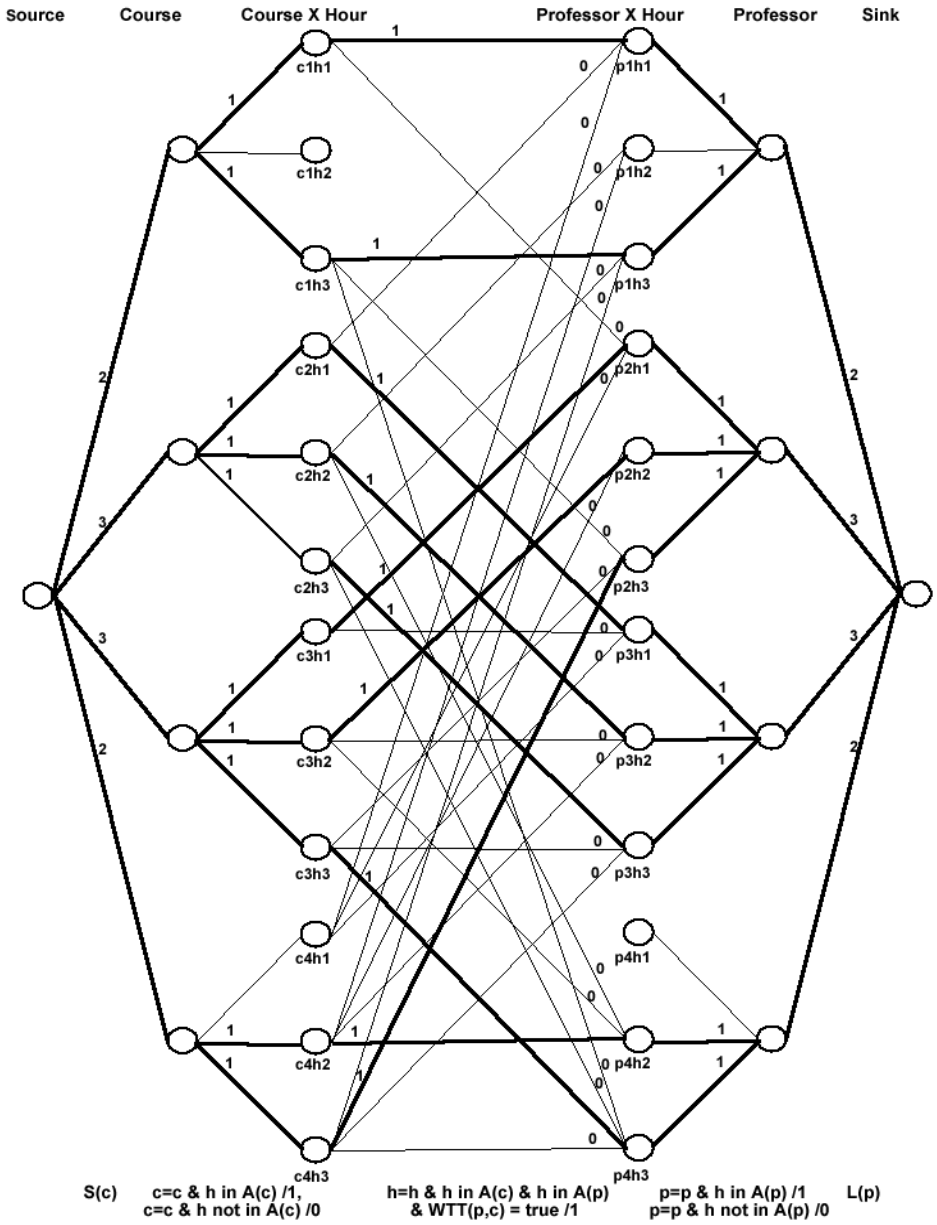

Figure 23 Maximum Flow, F*, for Network in Figure 21

Figure 24, below, shows the timetable, f^*, constructed from the maximal flow in Figure 23 in the manner described above. For example, the top bold flow of one in Figure 23 is from node (c_1, h_1) to the node (p_1, h_1), hence $f^*(p_1, c_1, h_1) = 1$ indicating professor p_1 will teach a section of c_1 at hour h_1. Note that there are ten such bold arcs with flow of one between (c, h) nodes and (p, h) nodes. Hence the timetable, f^*, schedules the ten section as shown in Figure 24.

The Maximal Timetable ***f****

$\{f^*(p_1, c_1, h_1) = 1,\ f^*(p_1, c_1, h_3) = 1,\ f^*(p_2, c_3, h_1) = 1,$
$f^*(p_2, c_3, h_2) = 1,\ f^*(p_2, c_4, h_3) = 1,\ f^*(p_3, c_2, h_1) = 1,$
$f^*(p_3, c_2, h_2) = 1,\ f^*(p_3, c_2, h_3) = 1,\ f^*(p_4, c_3, h_3) = 1,$
$f^*(p_4, c_4, h_2) = 1\}$
$f^*(p, c, h) = 0$ for all (p, c, h) not listed

Optimal Number of Sections Scheduled = 10 of 11

Figure 24 Optimal Timetable for Figure 20

3.2 Correctness of the Algorithm

In section 3.1 an algorithm was described to create a function f^*: $P \times C \times H \rightarrow \{0, 1\}$ for any instance of the BCS Optimization problem in three stages; in stage one a flow network G* was created, then a maximum flow F* was extracted from G* in phase two and finally, in stage three, a function f^* was created from the maximum flow F*. Here it will be shown that the algorithm will indeed produce correct results for the corresponding BCS Optimization problem no matter what BCS Optimization instance is input into the algorithm. This requires showing that f^* is a valid timetable for the given instance. In other words, f^* must satisfy constraints 1-8 of the BCS Optimization problem as defined in Figure 4. This also requires showing that f^* is optimal among all such valid timetables.

Before proving correctness of f^*, and for convenience in later sections, it is shown that the flow in F* on each arc from (c, h) to (p, h) in G* must be either 0 or 1. This is

accomplished by showing that the flows of F* on all arcs are integer. Then because the capacity on each arc between from (c, h) to (p, h) is 1 and the flows are non-negative each such flow is 0 or 1.

The capacity of each arc in the network, G*, created in stage one is an integer. This is known because the S(c) and L(p) inputs which provide the capacities of the first and last set of arcs respectively are non-negative integers by definition (Z_0^+). And the capacities of all the middle arcs are set to either 0 or 1, hence they are also integers. This means the residual flow R* is initialized with integers and the flow F* is initialized with zeros (0), which are also integers. It is guaranteed the maximal flow on each augmenting path is also an integer because it is determined by the 'bottle neck" of the path (i.e. the least of the residual capacities of arcs in R* on the path p*). Since initially the all the arcs have integer capacities, for each iteration, the augmenting path will add an integer flow to the selected arcs in the flow F*. In addition, the update to the residual flow R* will be by an integer on the selected arcs. Thus, at any given iteration, both the flow in F* on each arc and the residual capacity in R* are always integers. Hence, when the algorithm terminates, the flows in F* are all integer.

Now it is shown that the timetable f^*, created in stage three satisfies the five constraints in Figure 4:

1) and 3) Show for each $(p, c, h) \in P \times C \times H$, $f^*(p, c, h) = 1$ only if $h \in A(p) \cap A(c)$. Show for each $p \in P$, $c \in C$, and $h \in H$, $f^*(p, c, h) = 1$ only if $WTT(p, c) = true$. For each $(p, c, h) \in P \times C \times H$, by the definition of f^* in stage three, $f^*(p, c, h) = 1$ <u>if and only if</u> there is an arc from (c, h) to (p, h) in G* and it has a flow of 1 in F*. Weakening the statement gives $f^*(p, c, h) = 1$ <u>only if</u> there is an arc from

(c, h) to (p, h) in G* and it has a flow of 1 in F*. From the construction of G* in stage one there is an arc from (c, h) to (p, h) if and only if $h \in A(p)$, $h \in A(c)$ and $WTT(p, c) = true$. Combining the two statements above one gets the following statement: for each $(p, c, h) \in P \times C \times H$, $f^*(p, c, h) = 1$ only if $h \in A(p) \cap A(c)$ and $WTT(p, c) = true$. This includes the two statements to be proven. QED.

2) Show for each pair $h \in H$ and $p \in P$ there is at most one $c \in C$ for which $f^*(p, c, h) = 1$. It is known, by the construction of G* in stage one, the total capacity out of each (p, h) node is all on the arc from (p, h) to p and is 0 or 1. Further the inflow to each (p, h) node in F* comes through certain (c, h) to (p, h) arcs, in particular exactly those that correspond to (p, c, h) for which $f^*(p, c, h) = 1$ and each contributes a flow of 1. Hence input to $(p, h) = \Sigma_c f^*(p, c, h)$. Since, in a valid network flow, a particular node's inflow must equal that same node's outflow (i.e., inflow = outflow) and the outflow capacity of each (p, h) node is 0 or 1 it is concluded that $\Sigma_c f^*(p, c, h) \leq 1$. Since each $f^*(p, c, h)$ is 0 or 1, it follows that there is at most one $c \in C$ for which $f^*(p, c, h) = 1$.

3) See item number 1) above.

4) Show that for each $p \in P$, $\Sigma_{c, h} f^*(p, c, h) \leq L(p)$. Since F* is a valid flow, for each $p \in P$ the inflow at p is equal to the outflow at p. Since the inflow at p is $\Sigma_{c, h} f^*(p, c, h)$ consisting of 1 for each arc between a $C \times H$ and a $P \times H$ node with a flow of 1 in F*, it follows that the outflow of p is also $\Sigma_{c, h} f^*(p, c, h)$. The total outflow from p is contained on the single arc from p to the sink node.

The capacity of this arc is $L(p)$ and since a valid flow respects the capacity bounds it is concluded that for each $p \in P$, $\Sigma_{c,\,h} f^*(p, c, h) \leq L(p)$. QED.

5) Show that for each $c \in C$, $\Sigma_{p,\,h} f^*(p, c, h) \leq S(c)$. Since F* is a valid flow, for each $c \in C$ the inflow at c is equal to the outflow at c. Since the outflow at c is $\Sigma_{p,\,h} f^*(p, c, h)$ consisting of 1 for each arc between a $C \times H$ and a $P \times H$ node with a flow of 1 in F*, it follows that the inflow of c is also $\Sigma_{p,h} f^*(p, c, h)$. The total inflow to c is contained on the single arc from the source node to c. The capacity of this arc is $S(c)$ and since a valid flow respects the capacity bounds it is concluded that for each $c \in C$, $\Sigma_{p,\,h} f^*(p, c, h) \leq S(c)$. QED.

Thus any timetable f^* created in stage three has been shown to respect the five constraints of the BCS Optimization problem as given in Figure 4 and is thus a valid timetable.

Now it is left to show that f^* is optimal among all valid timetables. Suppose to the contrary that f^* does not schedule the maximal number of sections, then there is some timetable, $f^{\dagger}$, which schedules more sections then f^*. From the timetable $f^{\dagger}$ a flow $F^{\dagger}$ can be constructed on the network G*.

Note that if $f^{\dagger}(p, c, h) = 1$ then since $f^{\dagger}$ is a valid timetable it can be concluded that $h \in A(p)$, $h \in A(c)$ and $WTT(p, c) = true$. Hence because of the way that G* was constructed there is an arc between (c, h) and (p, h) in G^* and it has a capacity of 1. It satisfies the capacity constrains to set the flow on the arc from (c, h) *to* (p, h) in $F^{\dagger}$ to be 1. Construct $F^{\dagger}$ as follows; if $f^{\dagger}(p, c, h) = 1$ set the flow on the arc from (c, h) to (p, h) in $F^{\dagger}$ to be 1, on the other hand if $f^{\dagger}(p, c, h) = 0$ and there is an arc from (c, h) to (p, h) in G* set the flow on this arc in $F^{\dagger}$ to be 0.

An examination of the flow network G* shows that once the flow on all arcs from the $C \times H$ nodes to the $P \times H$ nodes are set, the whole network flow $F^{\dagger}$ can be uniquely determined by respecting conservation of flow, so complete the network flow, $F^{\dagger}$, by respecting the conservation of flow (i.e. node inflow = node outflow).

Each unit of flow, from source to sink, in the network $F^{\dagger}$, passes through some arc from some $C \times H$ node to some $P \times H$ node. There are exactly as many such arcs each with a flow of 1 in $F^{\dagger}$ as there are triples (p, c, h) for which $f^{\dagger}(p, c, h) = 1$. This is the number of sections scheduled in $f^{\dagger}$. Thus the flow from source to sink in the flow network $F^{\dagger}$ would have a greater flow than the maximal flow F* given by the Edmonds-Karp Algorithm. This would be a clear contradiction.

Therefore, f^* is both a valid timetable, respecting all five constraints of the BCS Optimization problem and is optimal among all such valid timetables.

3.3 Computational Complexity of the Algorithm

The computational complexity of the three stage algorithm presented in section 3.1 will now be shown to be polynomial in the size of the inputs.

Both stages one and three involve expressing numbers. In analyzing the computational complexity of an algorithm that involves numbers, the lengths of the numbers are important. The function $L_{en}(k)$ is used to indicate the length of a number k. It means the number of digits (or bits) needed to represent the number k and is approximately $log_b k$ where b is the base (usually ten or two) in which k is expressed. For example the length of the (base ten) number 9397 is four whereas the length of the binary number 10011010 is eight. Typically the base is unimportant in the analysis because logs

with two different bases are constant multiples of each other and the constant is absorbed in the Big O notation.

Therefore let $K = \Sigma_i (L_{en}(L(p_i))) + \Sigma_j (L_{en}(S(c_j)))$. K is the total length of the numbers that occur in the input of an example of this BCS Problem. And, for purposes of this computational complexity discussion, let $c = |C|$, $p = |P|$, and $h = |H|$.

Stage one of the algorithm is $O((p*c*h)+K)$ which includes:

- $O(2+c+(c*h)+(p*h)+p) = O((c*h)+(p*h))$for creating the nodes,
- $O(c+(c*h)+(p*c*h)+ (p*h)+p) = O(p*c*h)$ for creating the arcs,
- $O(\Sigma_j (L_{en}(S(c_j)))+(c*h)+(p*c*h)+(p*h)+\Sigma_i (L_{en}(L(p_i)))) =$ $O(K+(c*h)+(p*c*h)+(p*h)) = O(K+(p*c*h))$ for creating the weights on the arcs, where:
  - $\Sigma_j (L_{en}(S(c_j)))$ for the lengths of the numbers representing the desired sections for each course (source to C arc values);
  - $c*h$ for the length of the weights of 0s or 1s on the C to $C\times H$ arcs;
  - $p*c*h$ for the length of the weights of 1s on the $C\times H$ to $P\times H$ arcs;
  - $p*h$ for the length of the weights of 0s or 1s on the $P\times H$ to P arcs;
  - $\Sigma_i (L_{en}(L(p_i)))$ is the length of the numbers representing the maximum professor load for each professor (P to sink arc values).

Thus stage one is polynomial in *c, p, h* and K jointly, which is a fortiori polynomial in the size of the input which includes sets of size c, p, and h in addition to the total length K for the number of desired section and professor load bounds.

The computational complexity of the FFM using the Edmonds-Karp algorithm on any network is $O(VE^2)$, where V is the number of vertices and E is the number of arcs in

the network[4]. In this particular algorithm $V = c+(c*h)+(p*h)+p+2$ and $E \leq c+(c*h)+(p*c*h)+(p*h)+p$. So V is $O(p*c*h)$ and E is $O(p*c*h)$, thus stage two has a complexity of $O((p*c*h) * (p*c*h)^2) = O(p*c*h)^3$. This is polynomial in c, p, and h jointly, which is a fortiori polynomial in the size of the input which includes sets of size c, p, and h.

In stage three let $J = L_{en}(c) + L_{en}(p) + L_{en}(h)$ which is the sum of the lengths of the largest course number, largest professor number and largest hour. Extracting the assignment from the maximum flow is $O(c*p*h)$ to examine the results and $O(p*c*h*J)$ to write out the triples (c, p, h), each of which has a course number, professor number and hour, as well as, the f^* value of 1 or 0 for each triple. This means stage three is $O(p*c*h*J)$ which is polynomial in c, p, h and J jointly. If the courses, professors and hours are input with subscripts (e.g. $c_1, \ldots c_{|c|}$) then the length of the input includes $L_{en}(c) + L_{en}(p) + L_{en}(h)$, i.e. J.

Therefore the overall computational complexity of this algorithm which solves the BCS Problem is $O((K+(p*c*h))+(p*c*h)^3+(p*c*h*J))$, which is a 9^{th} degree polynomial in c, p, h, K, and J jointly.

3.4 BCS Decision Problem is in P

Section 3.1 presented an algorithm for solving the BCS Optimization problem and section 3.3 proved its computational complexity is polynomial in the size of the input. That algorithm can, in turn, be used to determine the answer to the corresponding BCS Decision problem as defined in Figure 3, with polynomial time amount of extra work. This is done by first transforming the given BCS Decision problem instance into a corresponding BCS Optimization problem instance; this is ***no*** additional work as the BCS

[4] Cormen et al., 660-663

Optimization problem instance has the exact same input as the corresponding BCS Decision problem instance. Next, using the algorithm provided in this chapter find an optimal timetable for the corresponding BCS Optimization problem. This has been shown above to be polynomial in the length of the input. After the algorithm is complete count up all the sections scheduled in the optimal timetable, $\Sigma_{ijk}(f^*(p_i,c_j,h_k))$, and also sum up all the desired number of sections, $\Sigma_j(S(c_j))$, and then compare the two values to see if they are equal. The answer to the BCS Decision problem is *yes* if the equality is *true* and *no* if the equality is *false*.

The computational complexity of this 'extra amount of work' involves the two summations and the comparison. For purposes of this computational complexity discussion, let $c = |C|$, $p = |P|$, and $h = |H|$. The computational complexity of the comparison depends on the sizes of the two numbers being compared, or more specifically the size of the larger of the two numbers being compared.

- Summing all the scheduling sections $\Sigma_{i\,j\,k}(f(p_i,c_j,h_k))$ is $O((p^*c^*h) * L_{en}((p^*c^*h))$. And the summation has a length $\leq L_{en}(p^*c^*h)$,
  - $L_{en}(p^*c^*h)$ is $O(L_{en}(c) + L_{en}(p) + L_{en}(h)) = O(J)$,
- Summing all the desired sections $\Sigma_j(S(c_j))$is $O(c * L_{en}(\Sigma_j(S(c_j))))$. And the number that results from the this summation has a length $\leq L_{en}(\Sigma_j(S(c_j))$,
  - $L_{en}(\Sigma_j(S(c_j))$ is $O(\Sigma_j\,(L_{en}(S(c_j))) = O(K)$,
- Comparing the two numbers is $O(J + K)$.

Thus the computational complexity of the 'extra amount of work' needed to solve the BCS Decision problem is $O(J + K + J + K) = O(J + K)$ which is polynomial in c, p, h and K jointly, which is a fortiori polynomial in the size of the input which includes sets

of size c, p, and h and numbers of total length K. And the total computational complexity of the BCS Decision problem is $O((K+(p*c*h))+ (p*c*h)^3+ (p*c*h*J)+ (J + K)$ which is a 9^{th} degree polynomial in c, p, h, K, J and M jointly. Therefore it can be said that the BCS Decision problem can be solved in polynomial time.

The question was raised in section 1.1.2 regarding whether or not the BCS Decision problem is NP-Complete. Because the algorithm in section 3.1 presented for the BCS Optimization problem solves it with computation complexity polynomial in the length of the input and given the simple modification presented above shows that the BCS Decision problem can also be solved in polynomial time in the length of the input, it is not likely to be NP-Complete. It is NP-Complete only if P = NP (i.e. the set of all Polynomial problems is equal to the set of Nondeterministic Polynomial problems). It is believed by most Computer Scientist the P ≠ NP but that has not been proven.

4. ECS DECISION PROBLEM IS IN NP-COMPLETE

This chapter presents a proof that the ECS Decision problem is NP-Complete. This proof is original work of this thesis. Section 4.1 gives the definition of NP-Complete while the concept of NP-Completeness is discussed in more detail in Appendix A.

4.1 Definition of NP-Complete

For a problem, Π, to be NP-Complete it must meet both criteria of the definition:

1) $\Pi \in NP$
2) From each problem in the class NP there is a Polynomial Transformation to the problem, Π, that Preserves Answers.

To directly show the second criterion is met is a very daunting task. However it has been made easier by the work of Cook[5] who provided a proof that the problem known as Satisfiability (SAT) met both criteria and by other Computer Scientists who proved additional problems are in NP-Complete while developing the concept of a Polynomial Transformation that Preserves Answers (PTPA). Because of their work a new problem, Π, can be shown to be NP-Complete by proving it meet both of the following criteria

1) $\Pi \in NP$
2) There exists PTPA from a known NP-Complete problem to the new problem, Π.

The proof in this thesis that ECS Decision problem is in NP is straight forward although long and tedious. For this reason it has been put in Appendix B.

[5] Stephen Cook, "The Complexity of Theorem-Proving Procedures." Proceedings of the 3rd Annual AMC Symposium on Theory of Computing (1971), 155-158.

In this chapter it is shown that the second criterion is met by showing there is a PTPA from the known NP-Complete problem Three Dimensional Matching (3DM) to the ECS Decision problem. 3DM was proven to be NP-Complete by Karp[6] in 1972.

In section 4.2 the 3DM problem is reviewed and section 4.3 describes a transformation from 3DM problems to ECS Decision problems, while in sections 4.4 and 4.5 the transformation is shown to have the requisite properties. Choosing 3DM for this purpose and creating the PTPA from 3DM to ECS Decision is original work of this thesis.

4.2 Three Dimensional Matching (3DM)

The Three Dimensional Matching[7] (3DM) problem is given below in Figure 25.

INSTANCE:
Set $M \subseteq X \times Y \times Z$, where X, Y and Z are disjoint sets having the same number q of elements.

QUESTION:
Does M contain a matching, i.e., a subset $M` \subseteq M$ such that $|M`| = q$ and no two elements of M' agree in any coordinate?

Figure 25 Three Dimensional Matching

The instance describes the inputs for the problem; in this case the inputs consist of the following:

1) A number q, a set X, a set Y, a set Z where:
   i. The sets X, Y and Z have the same number, q, of elements ($|X| = |Y| = |Z| = q$)
   ii. The sets X, Y and Z are disjoint ($X \cap Y = \varnothing$ and $X \cap Z = \varnothing$ and $Y \cap Z = \varnothing$)
2) A set M which is a set of triples of the form (x, y, z) and is a subset of the cross product of X, Y and Z ($M \subseteq X \times Y \times Z$)

[6] R. Karp, "Reducibility among combinatorial problems," in Complexity of Computer Computations, ed. R. Miller and J Thatcher, (Plenum Press, New York, 1972), 85-103.
[7] Garey and Johnson, 221

The 3DM is a decision problem which asks, "Does M contain a matching, *M`*," that respects the following constraints:

1) *M`* is a set of triples of the form $(x, y, z)$ that appear in $M$ ($M` \subseteq M$).

2) There are q of these triples in $M`$ ($|M`| = q$).

3) Each $x \in X$, $y \in Y$ and $z \in Z$ appears in exactly one triple of $M`$ (no two elements of $M`$ agree in any coordinate and there are q of them).

Three instances of 3DM are given below. The first two have an answer of *yes* and witnesses are given for verification. The third has an answer of *no*.

```
q = 5
X = {x1, x2, x3, x4, x5}
Y = {y1, y2, y3, y4, Y5}
Z = {z1, z2, z3, z4, Z5}
M = {
    1)  (x4, y1, z5),
    2)  (x1, y5, z3),
    3)  (x2, y4, z4),
    4)  (x3, y3, z4),
    5)  (x1, y2, z2),
    6)  (x2, y4, z3),
    7)  (x5, y3, z2),
    8)  (x4, y1, z1),
    9)  (x3, y2, z1),
    10) (x5, y5, z5)}
```

Answer = Yes

Witness

```
M` = {(x3, y3, z4),
      (x1, y2, z2),
      (x2, y4, z3),
      (x4, y1, z1),
      (x5, y5, z5)}
OR
M` = {4, 5, 6, 8, 10}
```

Figure 26 3DM Instance #1

```
q = 4
X = {Jon, Charlie, Mickey, Calvin}
Y = {Lucy, Minnie, Daisy, Patty}
Z = {Pluto, Snoopy, Garfield, Hobbes}
M = {
    1)  (Calvin, Lucy, Snoopy),
    2)  (Jon, Patty, Garfield),
    3)  (Charlie, Patty Hobbes),
    4)  (Mickey, Daisy, Hobbes),
    5)  (Jon, Minnie, Snoopy),
    6)  (Charlie, Patty, Garfield),
    7)  (Calvin, Daisy, Snoopy),
    8)  (Calvin, Lucy, Pluto),
    9)  (Mickey, Minnie, Pluto) }
```

Answer = Yes

Witness

```
M` = {(Mickey, Daisy, Hobbes),
      (Jon, Minnie, Snoopy),
      (Charlie, Patty, Garfield),
      (Calvin, Lucy, Pluto)}
OR
M` = {4, 5, 6, 8}
```

Figure 27 3DM Instance #2

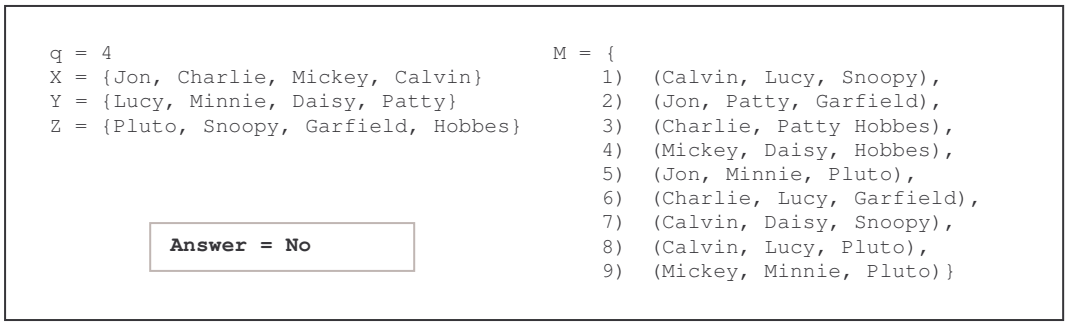

Figure 28 3DM Instance #3

4.3 Transformation from 3DM to ECS Decision

Here the transformation, *T*, from 3DM to ECS Decision created for this thesis is described. It is described in terms of a general method for constructing instances of the ECS Decision problem from instances of the 3DM problem. It will be shown in section 4.4 that answers are preserved by this transformation and in section 4.5 that this transformation is polynomial. This will establish that the transformation is indeed a PTPA.

The construction of *T*(*I*) is described in steps 1) - 6) below and an example is given in Figure 29.

1) For any given instance of the 3DM problem, label the elements in sets *X*, *Y* and *Z* as follows: *X* = {x1, x2 … xq}, *Y* = {y1, y2 … yq} , and *Z* = {z1, z2 … zq}

   i. For each *i* in *1,…q* construct a professor pi and construct *P* to be the set {p1, p2 … pq}. Note |*P*| = |*X*| = *q* and there is a natural one to one correspondence between elements of *X* and elements of *P* with matching subscripts.

   ii. For each *j* in *1,…q* construct a course cj and construct *C* to be the set {c1, c2 … cq}. Note |*C*| = |*Y*| = *q* and there is a natural one to one correspondence between elements of *Y* and elements of *C* with matching subscripts.

iii. For each k in $1,\ldots q$ construct a room rk and construct R to be the set {r1, r2 … rq}. Note $|R| = |Z| = q$ and there is a natural one to one correspondence between elements of Z and elements of R with matching subscripts.

2) For each triple (xi, yj, zk) in M construct an hour $h_{i\,j\,k}$. Let H be the set of hours constructed in this way, note $|H| = |M|$ and there is a natural one to one correspondence between the elements of M in 3DM and the elements of H in ECS Decision provided by the subscripts.

3) Construct the availability sets in the following manner:

i. For each pn ∈ P, construct A(pn) = $\{h_{i\,j\,k}. \in H : i = n\}$ (e.g. for Figure 29 $A(p_1) = \{h_{1\,1\,1}, h_{1\,1\,3}, h_{1\,3\,3}\}$, $A(p_2) = \{h_{2\,1\,3}, h_{2\,2\,2}\}$, etc.)

ii. For each $c_n \in C$, construct $A(c_n) = \{h_{i\,j\,k}. \in H : j = n\}$ (e.g. Figure 29 $A(c_1) = \{h_{1\,1\,1}, h_{1\,1\,3}, h_{2\,1\,3}, h_{1\,1\,2}\}$, $A(c_2) = \{h_{2\,2\,2}, h_{3\,2\,1}\}$, etc.)

iii. For each rn ∈ R, construct A(rn) = $\{h_{i\,j\,k}. \in H : k = n\}$ (e.g. Figure 29 $A(r_1) = \{h_{1\,1\,1}, h_{3\,2\,1}\}$, $A(r_2) = \{h_{2\,2\,2}, h_{3\,1\,2}\}$, etc.)

Note: if $(x_i, y_j, z_k) \in M$ then $h_{i\,j\,k} \in H$ and $A(p_i) \cap A(c_j) \cap A(r_k) = \{h_{i\,j\,k}\}$

and if $(x_i, y_j, z_k) \notin M$ then $h_{i\,j\,k}$ does not exist in H and $A(p_i) \cap A(c_j) \cap A(r_k) = \varnothing$

4) Construct the bounding numbers as follows:

i. $L(p) = 1$ for each p ∈ P.

ii. $S(c) = 1$ for each c ∈ C.

iii. $U(r) = 1$ for each r ∈ R.

5) Construct the Willing to Teach function such that $WTT(p, c) = True$ for each pair $(p, c) \in P \times C$.

6) Construct the Room Suitability function such that $RS(c, r) = True$ for each pair $(c, r) \in C \times R$.

This completes the construction of the instance of ECS Decision that corresponds to a given instance of 3DM according to the transformation, $T(\ )$. So for each instance I of 3DM $T(I)$ is the instance of ECS Decision created in this manner. Below in Figure 29 is a particular 3DM instance, I_1, and the ECS Decision instance, $T(I_1)$ constructed using the transformation described above.

3DM instance I_1

$q = 3$

$X = \{x_1, x_2, x_3\}$
$Y = \{y_1, y_2, y_3\}$
$Z = \{z_1, z_2, z_3\})$

$M = \{(x_1, y_1, z_1), \rightarrow h_{1\,1\,1}$
$(x_1, y_1, z_3), \rightarrow h_{1\,1\,3}$
$(x_1, y_3, z_3), \rightarrow h_{1\,3\,3}$
$(x_2, y_1, z_3), \rightarrow h_{2\,1\,3}$
$(x_2, y_2, z_2), \rightarrow h_{2\,2\,2}$
$(x_3, y_1, z_2), \rightarrow h_{3\,1\,2}$
$(x_3, y_2, z_1), \rightarrow h_{3\,2\,1}$
$(x_3, y_3, z_3)\} \rightarrow h_{3\,3\,3}$

Created ECS Decision instance $T(I_1)$

$P = \{p_1, p_2, p_3\}$
$C = \{c_1, c_2, c_3\}$
$R = \{r_1, r_2, r_3\})$

$H = \{h_{1\,1\,1}, h_{1\,1\,3}, h_{1\,3\,3}, h_{2\,1\,3}, h_{2\,2\,2}, h_{3\,1\,2}, h_{3\,2\,1}, h_{3\,3\,3}\}$

$A(p_1) = \{h_{1\,1\,1}, h_{1\,1\,3}, h_{1\,3\,3}\}$
$A(p_2) = \{h_{2\,1\,3}, h_{2\,2\,2}\}$
$A(p_3) = \{h_{3\,1\,2}, h_{3\,2\,1}, h_{3\,3\,3}\}$

$A(c_1) = \{h_{1\,1\,1}, h_{1\,1\,3}, h_{2\,1\,3}, h_{3\,1\,2}\}$
$A(c_2) = \{h_{2\,2\,2}, h_{3\,2\,1}\}$
$A(c_3) = \{h_{1\,3\,3}, h_{3\,3\,3}\}$

$A(r_1) = \{h_{1\,1\,1}, h_{3\,2\,1}\},$
$A(r_2) = \{h_{2\,2\,2}, h_{3\,1\,2}\},$
$A(r_3) = \{h_{1\,1\,3}, h_{1\,3\,3}, h_{2\,1\,3}, h_{3\,3\,3}\}$

$L(p_1) = 1,\ L(p_2) = 1, L(p_3) = 1$

$S(c_1) = 1,\ S(c_2) = 1, S(c_3) = 1$

$U(r_1) = 1,\ U(r_3) = 1, U(r_3) = 1$

$WTT(p, c) =$

	c_1	c_2	c_3
p_1	T	T	T
p_2	T	T	T
p_3	T	T	T

$RS(c, r) =$

	r_1	r_2	r_3
C_1	T	T	T
C_2	T	T	T
C_3	T	T	T

Figure 29 Transformation from 3DM to ECS Decision Application #1

4.4 The Transformation Preserves Answers

It must now be shown that the transformation, T, preserves answers (i.e. answers to each instance I in 3DM and its corresponding instance $T(I)$ in ECS Decision are both *yes* or both *no*). This is done by showing the following:

1) For each instance *I* of 3DM, if answer to *I* is *yes* (i.e. *I* has a witness) then answer to *T*(*I*) is yes (i.e. *T*(*I*) has a witness).

2) For each instance *I* of 3DM, if answer to *T*(*I*) is *yes* (i.e. *T*(*I*) has a witness) then answer to *I* is yes (i.e. *I* has a witness). Note that this part is logically equivalent to showing that if the answer to *I* is *no* then the answer to *T*(*I*) is also *no* by the law of the contrapositive.

This is accomplished by showing how to use a witness of *I*, if it exists, to construct a witness for *T*(*I*) and then showing how to use a witness for *T*(*I*), if it exists, to construct a witness for *I*. The former is done in section 4.4.1 and the latter in section 4.4.2.

4.4.1 If the Answer to 3DM is Yes then the Answer to ECS Decision is Yes

Suppose that *I* is an instance of the 3DM Problem with its particular *X, Y, Z, q* and *M* and its answer is *yes*, then it has a witness *M`*. Thus *M`* is a subset of *M*, $|M`| = q$, and no two element of *M`* agree in any coordinate. In other words:

1) *M`* is a set of triples of the form (*x*, *y*, *z*) that appear in *M*.

2) There are exactly *q* of these triples in *M`*.

3) Each $x \in X$, $y \in Y$ and $z \in Z$ appears in exactly one triple of *M`*.

Here it is shown that *T*(*I*) which is the instance of ECS Decision constructed from *I* as described in section 4.3, also has a witness and hence its answer is also *yes*. Such a witness is described in terms of constructing it by using the witness for *I*, i.e. using *M`*. From the construction of the ECS Decision instance, as described in section 4.3, *T*(*I*) has the following components:

1) Sets *P*, *C* and *R* constructed from *X*, *Y* and *Z*,

2) A set *H* constructed from *M*,

3) For each p ∈ P, c ∈ C, r ∈ R subsets $A(p)$, $A(c)$ and $A(r)$ of H constructed as described in section 4.3,

4) The function WTT: $P \times C \rightarrow \{True, False\}$,

5) The function RS: $C \times R \rightarrow \{True, False\}$,

6) The bounding numbers $L(p)$, $S(c)$ and $U(r)$.

A witness for ECS Decision is constructed in two stages. First construct a subset S of $PxCxHxR$ where P, C, H and R are from $T(I)$ as follows: for each (x_i, y_j, z_k) in $M`$, put $(p_i, c_j, h_{i\,j\,k}, r_k)$ in S such that the i's, j's and k's all correspond. Then, using S, construct the function f: $P \times C \times H \times R \rightarrow \{0, 1\}$ by assigning $f(p, c, h, r) = 1$ if $(p, c, h, r) \in S$ and $f(p, c, h, r) = 0$ if $(p, c, h, r) \notin S$. It remains to show that f is a valid timetable for $T(I)$ that schedules all courses in C and hence serves as a witness to show the answer to $T(I)$ is *yes*.

This construction is illustrated in Figure 30 below for the particular 3DM instance I_1, given above in Figure 29, starting with its witness $M`$.

```
Witness for the 3DM instance I1
M` = {(x1, y1, z1),     → h1 1 1
      (x2, y2, z2),     → h2 2 2
      (x3, y3, z3)}     → h3 3 3

Set S constructed using M`
S = {(p1, c1, h1 1 1, r1),
     (p2, c2, h2 2 2, r2),
     (p3, c3, h3 3 3, r3)}

f constructed using S
{f(p1, c1, h1 1 1, r1) = 1,
 f(p2, c2, h2 2 2, r2) = 1,
 f(p3, c3, h3 3 3, r3) = 1}

f(p, c, h, r) = 0 for (p, c, h, r) ∉ S
```

Figure 30 Construction of *f* from *M'*

Since f is constructed to be a function from $P \times C \times H \times R$ to $\{0, 1\}$ it is the right kind of thing to be a timetable for the ECS Decision problem T(I). It remains to show that f is a valid timetable for $T(I)$ by showing it respects problem constraints 1 - 8, as formally

defined in Figure 5 and that f schedules all the desired sections of each course. First, it will be shown that f respects all the eight constraints in Figure 5 (availability, physical, academic and bounding) and hence is a valid timetable for $T(I)$. Once that is established it will be shown that f schedules all desired sections of each course. Then having established that f is a witness for $T(I)$ it is concluded that the answer to $T(I)$ is *yes*.

Availability Constraints:

1) Show that $f(p, c, h, r) = 1$ only if $h \in [A(p) \cap A(c) \cap A(r)]$. Suppose $f(p, c, h, r) = 1$ then, by the construction of f, $(p, c, h, r) \in S$. From the construction of S it is known that each $(p, c, h, r) \in S$ has the form (p_i, c_j, h_{ijk}, r_k) where $(x_i, y_j, z_k) \in M`$. From the construction of $T(I)$ and in particular the sets $A(p)$ it is known that $h_{ijk} \in A(p_i)$, from the construction of the sets $A(c)$ it is known that $h_{ijk} \in A(c_j)$, and from the construction of the sets $A(r)$ it is known that $h_{ijk} \in A(r_k)$. Hence for each $(p_i, c_j, h_{ijk}, r_k) \in S$, $h_{ijk} \in [A(p_i) \cap A(c_j) \cap A(r_k)]$ by virtue of the construction process for $T(I)$ and S.

Physical Constraints:

2) Show that for each pair $(p, h) \in P \times H$ there is at most one pair $(c, r) \in C \times R$ for which $f(p, c, h, r) = 1$. Suppose to the contrary that there exist (p_i, h^*) in $P \times H$ for which there exists two distinct pairs, say (c_w, r_x) and (c_y, r_z) in $C \times R$, for which $f(p_i, c_w, h^*, r_x) = 1$ and $f(p_i, c_y, h^*, r_z) = 1$ then (p_i, c_w, h^*, r_x) and (p_i, c_y, h^*, r_z) are in S. From (p_i, c_w, h^*, r_x) in S, and by the construction of S, $h^* = h_{iwx}$. Likewise from (p_i, c_y, h^*, r_z) in S, $h^* = h_{iyz}$. Therefore $h^* = h_{iwx} = h_{iyz}$ and it is concluded $w = y$ and $x = z$. Hence $(c_w, r_x) = (c_y, r_z)$ and they are *not* two

distinct pairs. Thus for each $(p, h) \in H \times P$ there is at most one pair $(c, r) \in C \times R$ for which $f(p, c, h, r) = 1$

3) Show that for each pair $(r, h) \in R \times H$ there is at most one pair $(p, c) \in P \times C$ for which $f(p, c, h, r) = 1$. Suppose to the contrary that there exist (r_k, h^*) in $R \times H$ for which there exists two distinct pairs, say (p_w, c_x) and (p_y, c_z) in $P \times C$, for which $f(p_w, c_x, h^*, r_k) = 1$ and $f(p_y, c_x, h^*, r_k) = 1$. Then (p_w, c_x, h^*, r_k) and (p_y, c_z, h^*, r_k) are in S. From (p_w, c_x, h^*, r_k) in S, and by the construction of S, $h^* = h_{wxk}$. Likewise from (p_y, c_z, h^*, r_k) in S, $h^* = h_{yzk}$. Therefore $h^* = h_{wxk} = h_{yzk}$ and it is concluded $w = y$ and $x = z$. Hence $(p_w, c_x) = (p_y, c_z)$ and they are *not* two distinct pairs. Thus for each $(r, h) \in R \times H$ there is at most one pair $(p, c) \in P \times C$ for which $f(p, c, h, r) = 1$

Academic Constraints:

4) Show that for each $p \in P$, $c \in C$, $r \in R$ and $h \in H$, $f(p, c, h, r) = 1$ only if $WTT(p, c)$ = true. Since $WTT(p, c) = True$ for each pair $(p, c) \in P \times C$ per the construction of the $T(I)$, the academic constraint (#4 in Figure 5) is respected.

5) Show that for each $p \in P$, $c \in C$, $r \in R$ and $h \in H$, $f(p, c, h, r) = 1$ only if $RS(c, r)$ = true. Since $RS(c, r) = True$ for each pair $(c, r) \in C \times R$ per the construction of $T(I)$, the academic constraint (#5 in Figure 5) is respected.

Bounding Constraints:

6) Show that for each $p \in P$, $\Sigma_{c,h,r} f(p, c, h, r) \leq L(p)$. Per the construction of $T(I)$ it is know that $L(p) = 1$ for each $p \in P$. Note each $p \in P$ is some p_i by the construction of P. From the construction of f, $f(p_i, c_j, h_{ijk}, r_k) = 1$ if and only if $(p_i, c_j, h_{ijk}, r_k) \in S$ and $(p_i, c_j, h_{ijk}, r_k) \in S$ if and only if $(x_i, y_j, z_k) \in M`$. Since

each x_i appears in exactly one triple of $M`$ it follows that p_i appears in exactly one quadruple of S. Therefore $f(p_i, c, h, r) = 1$ for exactly that one quadruple (p_i, c, h, r) and $f(p_i, c, h, r) = 0$ for all other c, h, r. Thus $\Sigma_{c,h,r} f(p_i, c, h, r) = 1$. Hence for each $p \in P \Sigma_{c,h,r} f(p, c, h, r) = 1$ and since each $L(p) = 1$ it can be concluded that $\Sigma_{c,h,r} f(p, c, h, r) \leq L(p)$, indeed $\Sigma_{c,h,r} f(p, c, h, r) = L(p)$.

7) Show that for each $c \in C, \Sigma_{p,h,r} f(p, c, h, r) \leq S(c)$. Per the construction of $T(I)$ it is know that $S(c) = 1$ for each $c \in C$. From the construction of f, $f(p_i, c_j, h_{ijk}, r_k) = 1$ if and only if $(p_i, c_j, h_{ijk}, r_k) \in S$ and $(p_i, c_j, h_{i\,j\,k}, r_k) \in S$ if and only if $(x_i, y_j, z_k) \in M`$. Since each $y_j$ appears in exactly one triple in $M`$ it follows that c_j appears in exactly one quadruple of S. Therefore $f(p, c_j, h, r) = 1$ for exactly that one quadruple (p, c_j, h, r) and $f(p, c_j, h, r) = 0$ for all other p, h, r. Thus $\Sigma_{c,h,r} f(p, c_j, h, r) = 1$. Hence for each $c \in C \Sigma_{p,h,r} f(p, c, h, r) = 1$ and $\Sigma_{p,h,r} f(p, c, h, r) \leq S(c)$.

8) Show that for each $r \in R, \Sigma_{p,c,h} f(p, c, h, r) \leq U(r)$. Per the construction of $T(I)$ it is know that $U(r) = 1$ for each $r \in R$. From the construction of f, $f(p_i, c_j, h_{ijk}, r_k) = 1$ if and only if $(p_i, c_j, h_{ijk}, r_k) \in S$ and $(p_i, c_j, h_{ijk}, r_k) \in S$ if and only if $(x_i, y_j, z_k) \in M`$. Since each $z_k$ appears in exactly one triple $M`$ it follows that r_k appears in exactly one quadruple of S. Therefore $f(p, c, h, r_k) = 1$ for exactly that one quadruple (p, c, h, r_k) and $f(p, c, h, r_k) = 0$ for all other c, p, h. Thus $\Sigma_{c,h,r} f(p, c, h, r_k) = 1$. Hence for each $r \in R \Sigma_{p,c,h} f(p, c, h, r) = 1$ and $\Sigma_{p,c,h} f(p, c, h, r) \leq U(r)$, indeed $\Sigma_{p,c,h} f(p, c, h, r) = U(r)$.

Thus all eight constraints are respected by f and f is indeed a valid timetable for $T(I)$.

Now it must be shown that f schedules all desired sections. Since each $y \in Y$ occurs in exactly one triple of $M`$, then by the construction of S, each c occurs in exactly one quadruple of S and hence for each c, $f(p, c, h, r) = 1$ for exactly one quadruple (p, c, h, r) in $P \times C \times H \times R$. In other words course c occurs in the timetable f exactly once. From the construction of $T(I)$ it is known that the number of desired section for each course is 1 (i.e. $S(c) = 1$ for each $c \in C$). Therefore it is assured that all desired sections in $T(I)$have been scheduled by f.

It has thus been proven that a *yes* answer for a 3DM instance will guarantee a *yes* answer for the corresponding ECS Decision instance. For example in particular for I_1 in Figure 29 it has been proven that the function f constructed in Figure 30 is indeed a witness for ECS Decision $T(I_1)$ constructed in Figure 29.

4.4.2 If the Answer to ECS Decision is Yes then the Answer to 3DM is Yes

Before starting the formal proof, another example of the construction of an ECS Decision instance $T(I)$ from a 3DM instance I using the Transformation T is given and discussed. This is shown in Figure 31 for a particular instance, I_2, of 3DM.

3DM instance I_2

$X = \{x_1, x_2, x_3, x_4, x_5\}$ $\quad q = 5$

$Y = \{y_1, y_2, y_3, y_4, y_5\}$

$Z = \{z_1, z_2, z_3, z_4, z_5\}$

$M = \{(x_1, y_1, z_2), \rightarrow h_{1\,1\,2}$ $\quad (x_1, y_3, z_3), \rightarrow h_{1\,3\,3}$

$(x_1, y_4, z_2), \rightarrow h_{1\,4\,2}$ $\quad (x_1, y_5, z_3), \rightarrow h_{1\,5\,3}$

$(x_2, y_5, z_1), \rightarrow h_{2\,5\,1}$ $\quad (x_2, y_4, z_1), \rightarrow h_{2\,4\,1}$

$(x_2, y_4, z_3), \rightarrow h_{2\,4\,3}$ $\quad (x_3, y_1, z_2), \rightarrow h_{3\,1\,2}$

$(x_3, y_1, z_5), \rightarrow h_{3\,1\,5}$ $\quad (x_3, y_3, z_1), \rightarrow h_{3\,3\,1}$

$(x_3, y_4, z_3), \rightarrow h_{3\,4\,3}$ $\quad (x_4, y_1, z_5), \rightarrow h_{4\,1\,5}$

$(x_4, y_4, z_2), \rightarrow h_{4\,4\,2}$ $\quad (x_4, y_5, z_2), \rightarrow h_{4\,5\,2}$

$(x_5, y_2, z_4), \rightarrow h_{5\,2\,4}$ $\quad (x_5, y_2, z_5), \rightarrow h_{5\,2\,5}$

$(x_5, y_3, z_3), \rightarrow h_{5\,3\,3}$ $\quad (x_5, y_5, z_5), \rightarrow h_{5\,5\,5}$

Created ECS Decision instance $T(I_2)$

$P = \{p_1, p_2, p_3, p_4, p_5\}$

$C = \{c_1, c_2, c_3, c_4, c_5\}$

$R = \{r_1, r_2, r_3, r_4, r_5\})$

$H = \{h_{1\,1\,2}, h_{1\,3\,3}, h_{1\,4\,2}, h_{1\,5\,3}, h_{2\,5\,1},$
$h_{2\,4\,1}, h_{2\,4\,3}, h_{3\,1\,2}, h_{3\,1\,5}, h_{3\,3\,1},$
$h_{3\,4\,3}, h_{4\,1\,5}, h_{4\,4\,2}, h_{4\,5\,2}, h_{5\,2\,4},$
$h_{5\,2\,5}, h_{5\,3\,3}, h_{5\,5\,5}\}$

$A(p_1) = \{h_{1\,1\,2}, h_{1\,3\,3}, h_{1\,4\,2}, h_{1\,5\,3}\}$

$A(p_2) = \{h_{2\,5\,1}, h_{2\,4\,1}, h_{2\,4\,3}\}$

$A(p_3) = \{h_{3\,1\,2}, h_{3\,1\,5}, h_{3\,3\,1}, h_{3\,4\,3}\}$

$A(p_4) = \{h_{4\,1\,5}, h_{4\,4\,2}, h_{4\,5\,2}\}$

$A(p_5) = \{h_{5\,2\,4}, h_{5\,2\,5}, h_{5\,3\,3}, h_{5\,5\,5}\}$

$A(c_1) = \{h_{1\,1\,2}, h_{3\,1\,2}, h_{4\,1\,5}\}$

$A(c_2) = \{h_{5\,2\,4}, h_{5\,2\,5}\}$

$A(c_3) = \{h_{1\,3\,3}, h_{3\,3\,1}, h_{5\,3\,3}\}$

$A(c_4) = \{h_{1\,4\,2}, h_{2\,4\,1}, h_{2\,4\,3}, h_{3\,4\,3},$
$h_{4\,4\,2}\}$

$A(c_5) = \{h_{1\,5\,3}, h_{2\,5\,1}, h_{4\,5\,2}, h_{5\,5\,5}\}$

$A(r_1) = \{h_{2\,5\,1}, h_{2\,4\,1}, h_{3\,3\,1}\}$

$A(r_2) = \{h_{1\,1\,2}, h_{1\,4\,2}, h_{3\,1\,2}, h_{4\,4\,2},$
$h_{4\,5\,2}\}$

$A(r_3) = \{h_{1\,3\,3}, h_{1\,5\,3}, h_{2\,4\,3},$
$h_{3\,4\,3}, h_{5\,3\,3}\}$

$A(r_4) = \{h_{5\,2\,4}\}$

$A(r_5) = \{h_{4\,1\,5}, h_{5\,2\,5}, h_{5\,5\,5}\}$

$L(p_1) = 1,\ L(p_2) = 1, L(p_3) = 1$

$S(c_1) = 1,\ S(c_2) = 1, S(c_3) = 1$

$U(r_1) = 1,\ U(r_3) = 1, U(r_3) = 1$

WTT(p, c) =

	c_1	c_2	c_3	c_4	c_5
p_1	T	T	T	T	T
p_2	T	T	T	T	T
p_3	T	T	T	T	T
p_4	T	T	T	T	T
p_5	T	T	T	T	T

RS(c, r) =

	r_1	r_2	r_3	r_4	r_5
C_1	T	T	T	T	T
C_2	T	T	T	T	T
C_3	T	T	T	T	T
C_4	T	T	T	T	T
C_5	T	T	T	T	T

Figure 31 Transformation from 3DM to ECS Decision Application #2

In particular, the following parts are created in constructing the instance $T(I_2)$ (see section 4.3 for the formal transformation):

1) Create the P, C and R sets for $T(I_2)$ from the X, Y and Z sets of I_2

   i. Since I_2 has $X = \{x_1, x_2, \ldots, x_5\}$ therefore $T(I_2)$ has $P = \{p_1, p_2, \ldots, p_5\}$.

   ii. Since I_2 has $Y = \{y_1, y_2, \ldots, y_5\}$ therefore $T(I_2)$ has $C = \{c_1, c_2, \ldots, c_5\}$.

   iii. Since I_2 has $Z = \{z_1, z_2, \ldots, z_5\}$ therefore $T(I_2)$ has $R = \{r_1, r_2, \ldots, r_5\}$.

2) Create the H set for $T(I_2)$ from the M set of the I_2.

i. Since I_2 has $M \subseteq X \times Y \times Z$, as shown in Figure 31, therefore $T(I_2)$ has $H=\{h_{ijk}:(x_i, y_j, z_k) \in M\}$

3) Construct the availability sets in the following manor:

i. $A(p_i) = \{h_{ijk} \in H$: where the i's in p_i and h_{ijk} match $\}$ for each $p \in P$,

ii. $A(c_j) = \{h_{ijk} \in H$: where the j's in c_j and h_{ijk} match $\}$ for each $c \in C$,

iii. $A(r_k) = \{h_{ijk} \in H$: where the k's in r_k and h_{ijk} match $\}$ for each $r \in R$,

4) Construct the bounding numbers as follows:

i. $L(p) = 1$ for each $p_i \in P$.

ii. $S(c) = 1$ for each $c_j \in C$.

iii. $U(r) = 1$ for each $r_k \in R$.

5) Construct the Willing to Teach function such that, for each $p \in P$ and $c \in C$, $WTT(p, c) = true$, this too is shown in Figure 31.

6) Construct the Room Suitability function such that, for each $c \in C$ and $r \in R$, $RS(c, r) = true$.

Now attention is turned to the formal proof that if the answer to ECS Decision is *yes* then the answer to 3DM is *yes*. Suppose that $T(I)$ is an instance of ECS Decision that is associated with the instance I of 3DM through the transformation T described in section 4.3, this means $T(I)$ has its particular P, C, R, H, $A(p)$, $A(c)$, $A(r)$, $WTT(p, c)$ and $RS(c, r)$. Also suppose that the answer to $T(I)$ is *yes*, then it has a witness f: $P \times C \times H \times R \rightarrow \{0, 1\}$ Moreover, since f is a witness for the ECS Decision instance, it respects all the problem constraints 1 – 8 from Figure 5 as repeated below. The function f also schedules all desired sections of each courses. This is expressed in item 9 below.

Availability Constraints:

1) For each quadruple (p, c, h, r) in $P \times C \times H \times R$, $f(p, c, h, r) = 1$ only if $h \in [A(p) \cap A(c) \cap A(r)]$.

Physical Constraints:

2) For each pair $(p, h) \in P \times H$ there is at most one pair $(c, r) \in C \times R$ for which $f(p, c, h, r) = 1$.

3) For each pair $(r, h) \in R \times H$ there is at most one pair $(p, c) \in P \times C$ for which $f(p, c, h, r) = 1$.

Academic Constraints:

4) For each $p \in P$, $c \in C$, $r \in R$ and $h \in H$, $f(p, c, h, r) = 1$ only if $WTT(p, c) = true$

5) For each $p \in P$, $c \in C$, $r \in R$ and $h \in H$, $f(p, c, h, r) = 1$ only if $RS(c, r) = true$

Bounding Constraints:

6) For each $p \in P$, $\Sigma_{c,h,r} f(p, c, h, r) \leq L(p)$ (where $L(p) = 1$ for each $T(I)$).

7) For each $c \in C$, $\Sigma_{p,h,r} f(p, c, h, r) \leq S(c)$ (where $S(c) = 1$ for each $T(I)$).

8) For each $r \in R$, $\Sigma_{p,c,h} f(p, c, h, r) \leq U(r)$ (where $U(r) = 1$ for each $T(I)$).

All Sections Scheduled:

9) For each $c \in C$, $\Sigma_{p,h,r} f(p, c, h, r) = S(c)$ (where $S(c) = 1$ for each $T(I)$).

First, using f construct the set S, where $S \subseteq P \times C \times H \times R$ such $(p, c, h, r) \in S$ if and only if $f(p, c, h, r) = 1$ and $(p, c, h, r) \notin S$ if and only if $f(p, c, h, r) = 0$. Note that each $(p, c, h, r) \in S$ has the form (p_i, c_j, h_{ijk}, r_k) because only these can have $f(p, c, h, r) = 1$ for the valid Timetable f. Next construct a subset N of $X \times Y \times Z$ where X, Y, and Z are from I as follows: for each (p_i, c_j, h_{ijk}, r_k) in S put (x_i, y_j, z_k) in N such that the i's, j's and k's all correspond. It remains to show that N a witness to I and hence the answer to I is *yes.*

To show that N is a valid witness for I it must be shown that N is a subset of $XxYxZ$ which respects all three problem constraints for the 3DM problem a) $N \subseteq M$, b) $|N| = q$, and c) that no two elements of N agree in any coordinate. It is clear from the construction of N that $N \subseteq XxYxZ$. It will now be shown that N respects all three problem constraints for 3DM hence is a valid witness for I.

a) Show that $N \subseteq M$. From the construction of N above it is known that for each $(x_i, y_j, z_k) \in N$ there exists a $(p_i, c_j, h_{i\,j\,k}, r_k) \in S$ such that the i's, j's and k's all correspond. Furthermore, by the construction of H in $T(I)$, for each $h_{i\,j\,k} \in H$ the corresponding (x_i, y_j, z_k) is in M. Therefore it can be concluded that $N \subseteq M$

b) Show that $|N| = q$. Since f is a witness for I and in particular schedules each course exactly once, for each $c_j \in C$ there is exactly one quadruple (p, c_j, h, r) of $P \times C \times H \times R$ for which , $f(p, c_j, h, r) = 1$. Since S is defined to be a subset of $P \times C \times H \times R$ for which $f(p, c_j, h, r) = 1$, S has exactly one quadruple for each $c_j \in C$. Hence $|S| = |C| = q$. By construction of N there is exactly one element in N for each element in S, therefore $|N| = |S| = q$.

c) To show N is a witness for I it remains to be shown that N satisfies constraint c (i.e. no two distinct elements of N agree in any coordinate). This will be demonstrated with three similar proofs by contradiction.

   i. Suppose to the contrary that some particular x, say x_α, appears in two different triples of N say (x_α, y_v, z_w) and (x_α, y_s, z_t). Then, because of how N was constructed, $(p_\alpha, c_v, h_{\alpha\,v\,w}, r_w)$ and $(p_\alpha, c_s, h_{\alpha\,s\,t}, r_t)$ are both in S. Furthermore $f(p_\alpha, c_v, h_{\alpha\,v\,w}, r_w) = 1$ and $f(p_\alpha, c_s, h_{\alpha\,s\,t}, r_t) = 1$ by the relationship between f and S. Thus $\Sigma_{c,h,r} f(p_\alpha, c, h, r) \geq 2$ but f, being a valid schedule, meets

constraint 6 that states for each $p \in P$, $\Sigma_{c,h,r} f(p, c, h, r) \leq L(p) \leq 1$. This is a clear contradiction.

ii. Suppose to the contrary that some particular x, say x_α, appears in two different triples of N, say (x_v, y_α, z_w) and (x_s, y_α, z_t). Then, because of how N was constructed, $(p_v, c_\alpha, h_{v\,\alpha\,w}, r_w)$ and $(p_s, c_\alpha, h_{s\,\alpha\,t}, r_t)$ are both in S. Furthermore $f(p_v, c_\alpha, h_{v\,\alpha\,w}, r_w) = 1$ and $f(p_s, c_\alpha, h_{s\,\alpha\,t}, r_t) = 1$ by the relationship between f and S. Thus $\Sigma_{p,h,r} f(p, c_\alpha, h, r) \geq 2$ but f, being a valid schedule, meets constraint 6 that states for each $c \in C$, $\Sigma_{p,h,r} f(p, c, h, r) \leq S(c) \leq 1$. This is a clear contradiction.

iii. Suppose to the contrary that some particular x, say x_α, appears in two different triples of N, say (x_v, y_w, z_α) and (x_s, y_t, z_α). Then, because of how N was constructed, $(p_v, c_w, h_{v\,w\,\alpha}, r_\alpha)$ and $(p_s, c_t, h_{s\,t\,\alpha}, r_\alpha)$ are both in S. Furthermore $f(p_v, c_w, h_{v\,w\,\alpha}, r_\alpha) = 1$ and $f(p_s, c_t, h_{s\,t\,\alpha}, r_\alpha) = 1$ by the relationship between f and S. Thus $\Sigma_{p,c,h} f(p, c, h, r_\alpha) \geq 2$ but f, being a valid schedule, meets constraint 6 that states for each $r \in R$, $\Sigma_{p,c,h} f(p, c, h, r) \leq U(r) \leq 1$. This is a clear contradiction.

Hence it has been shown that $N \subseteq M$, $|N| = |q|$, and no two elements of N agree in any coordinates. Thus a *yes* answer for the ECS Decision Problem $T(I)$ will guarantee a *yes* answer in the 3DM Problem I. Q.E.D.

This construction is illustrated below in Figure 32 for the particular ECS Decision instance $T(I_2)$ given in Figure 31 and its witness f below.

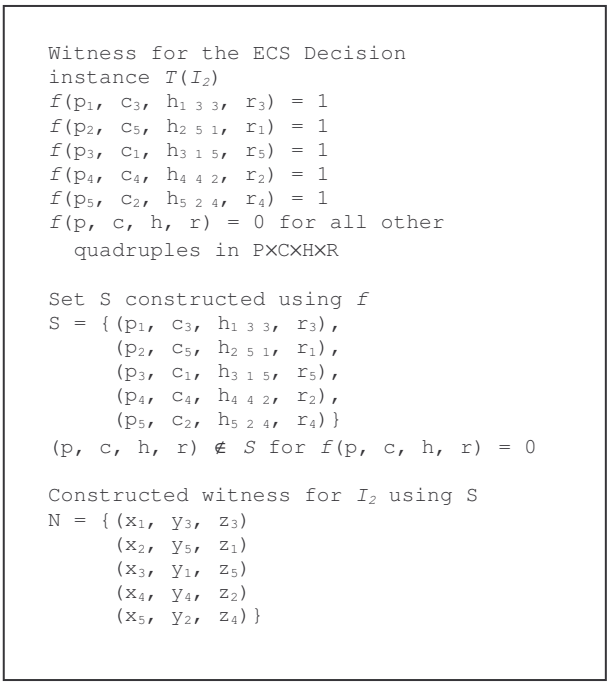

Figure 32 Construction of *N* from *f*

4.5 The Computational Complexity of the PTPA

Section 4.3 discusses a transformation from a known NP-Complete problem (3DM) to ECS Decision and section 4.4 proved that that transformation preserves answers. The last thing that must be shown to prove the transformation is a PTPA is that the transformation is indeed polynomial. Specifically, it must be shown that the construction of ECS Decision from the 3DM is polynomial in the size of the 3DM inputs.

This transformation function, T, involves the expression of numbers and as such the lengths of those numbers are important in computing the computational complexity. Let $k = L_{en}(q)$ represent the length of the number q, note k is $O(log(q))$. The largest number expressed in this algorithm is q, so k represents an upper bound on the lengths of all the numbers 1 through q since the largest number has the greatest length.

The PTPA has a computational complexity of $O((3+3qk)+(m+3km)+3q*(m+3km)+3q+2q^2)$ where k is $O(log(q))$ and $m = |M|$ which includes:

- $O((1+q*k) + (1+q*k) + (1+q*k)) = O(3*(1+qk)) = O(3+3qk)) = O(q*log(q))$ for creating the elements of the sets P, C, and R,
- $O(m*(1+3k)) = O(m*log(q))$ for creating the elements of the set H,
- $O(q*m*(1+3k) + q*m*(1+3k) + q*m*(1+3k)) = O(3q*m*(1+3k)) = O(q*m*log(q))$ for the $A(p)$, $A(c)$, and $A(r)$ availability sets,
- $O(q)$ for the construction of the $L(p)$, $S(c)$ and $U(r)$ bounding numbers,
- $O(q^2)$ for the construction of the $WTT(p, c)$ function,
- $O(q^2)$ for the construction of the $RS(c, r)$ function.

Thus the transformation from 3DM to ECS Decision is polynomial in the size of the inputs of 3DM which includes among other things three sets of size q, a set of size m and the number q of length $k = L_{en}(q)$, it is therefore shown to be a PTPA. This is the final step in the proof that ECD Decision Problem is indeed NP-Complete after noting that Appendix B provides the proof that ECD Decision Problem is in NP.

4.6 ECS Optimization Problem is in NP-Hard

This chapter contains most of the proof that the ECS Decision Problem is NP-Complete, The demonstration that ECS Decision is in NP is in Appendix B. Since the ECS Decision problem is in NP-Complete then the ECS Optimization problem is NP-Hard (this means it is at least as hard as an NP-Complete problem). In particular, if a polynomial algorithm is found which solves the ECS Optimization problem this polynomial algorithm can be used to solve the corresponding ECS Decision problem with a small amount of additional work. Similar to how the BCS Optimization Problem was used the solve the BCS Decision Problem section 3.4, count up all the sections scheduled in the optimal timetable, $\Sigma_{ijkl}(f^*(p_i,c_j, h_k,r_l))$, and also sum up all the desired number of

sections, $\Sigma_j(S(c_j))$, and then compare the two values to see they are equal. The reverse may not be true. If a polynomial algorithm is found which solves (*yes*/*no*) the ECS Decision problem there is no obvious way that that algorithm can then be used to solve the BCS Optimization Problem.

5. CONCLUSION

Now that it is known ECS Decision problem is NP-Complete and the ECS Optimization problem is no easier, trying to develop an exact, polynomial time algorithm would probably not be productive. Currently there are no known solutions to NP-Complete problems which run in time polynomial in the size of the input. There are a number of approaches that can be taken when confronted with an NP-Complete problem and specifically the ECS Optimization problem.

A popular approach is the use of Heuristic algorithms. Genetic Algorithms are the most well known of these heuristic approaches to course scheduling problems, although lesser known heuristics such as Tabu Search and Simulated Annealing also offer potentially viable solutions. The drawbacks to heuristic algorithms are that the runtime is not guaranteed to be polynomial nor are the results guaranteed to be optimal. These approaches have worked reasonably well (with regard to both runtime and results) for some NP-Complete problems.

A different approach would be to find a polynomial time algorithm that will solve a special case of ECS Optimization problem that meets all constraints and needs of a particular university, one example of this would be to if the courses were universally available (i.e. there are no restrictions on when courses could be taught). Another approach is to create an efficient polynomial time algorithm that finds a feasible solution for the whole problem but one that is not necessarily guaranteed to be optimal but which is ‘good enough’ to satisfy the needs of a university. These approaches have been left as future work.

Of course, if one must have an exact solution the entire ECS Optimization problem then using an exponential time algorithm is currently the only option.

BIBLIOGRAPHY

Agrawal, M., N. Kayal, and N. Saxena. "Primes is in P." Annals of Mathematics 160, No. 2 (2004): 781–793.

Cook, S. "The Complexity of Theorem-Proving Procedures." Proceeding of the 3rd Annual ACM Symposium on Theory of Computing (1971): 151-158

Cormen, T., C. Leiserson, R. Rivest, and C. Stein. Introduction to Algorithms. Cambridge: The MIT Press, 2001

Dinkel, J., J. Mote, M. Venkataramanan. "An Efficient Decision Support System for Academic Course Scheduling." Operations Research, Vol. 37, No. 6 (1989): 853-864

Dyer, J., and J. Mulvey. "An Integrated Optimization/Information System for Academic Departmental Planning." Management Science, Vol. 22, No. 12, (1976): 1332-1341

Edmonds J., and R. Karp, "Theoretical Improvements in the Algorithmic Efficiency for Network Flow Problems." Journal of ACM Vol. 19 (1972) 248-264

Evan, S., A. Itai, and A. Shamir. "On the Complexity of Timetable and Multicommodity Flow Problems." Siam Journal of Computing, Vol. 5 (1976): 691-703.

Garey, M., and M. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. New York: W. H. Freeman, 1976.

Karp R. "Reducibility among combinatorial problems." in Complexity of Computer Computations, ed. R. Miller and J Thatcher, Plenum Press, New York, 1972: 85-103.

Schaerf, A. "A Survey of Automated Timetabling." Artificial Intelligence Review. Vol. 13, No. 2, (1999): 87-127.

Schaerf, A., and L. Di Gaspero. 2001. "Local Search Techniques for Educational Timetabling Problem." Proceeding of the 6th International Symposium on Operational Research in Slovenia (SOR-01), Preddvor, Slovenia, (2001):13-23.

APPENDIX A

PROBLEM CLASSES P, NP AND NP-COMPLETE

To understand the significance of the theoretical work presented in this thesis knowledge about the problem classes P, NP, and NP-Complete is needed. Computer scientists have developed these classes to group together problems that have similar requirements with regard to the time it takes to solve them. This appendix offers a discussion of these classes. A problem class is actually a set of problems, so it will be useful to also review the way in which the term *problem* is used. The concept of computational complexity of an algorithm is also needed to understand these classes. A discussion of this concept is provided herein. Then discussions of the classes P, NP, NP-Complete are given.

Introduction to Problem Classes

Before discussing problem classes, the meanings of both terms *problem* and *problem instance* need to be clarified. A *problem* consists of a list of parameters and a question (or goal). An example of a *problem* is shown below in Figure 33, the parameter of this *problem* is a connected graph and the question is, "Does there exist a circuit in G that traverses every edge in E exactly once?" Until an actual graph is given this question cannot be answered.

Instance: A connected Graph G(V, E).
Question: Does there exist a circuit in G that traverses every edge in E exactly once?

Figure 33 Euler Circuit Problem

A *problem instance* results when values are specified for the parameters. Figure 34 below shows an instance of an Euler Circuit Problem. There a specific graph is given and the question now about the given graph has an answer of *yes* or *no*. Indeed the answer for this instance is *no*.

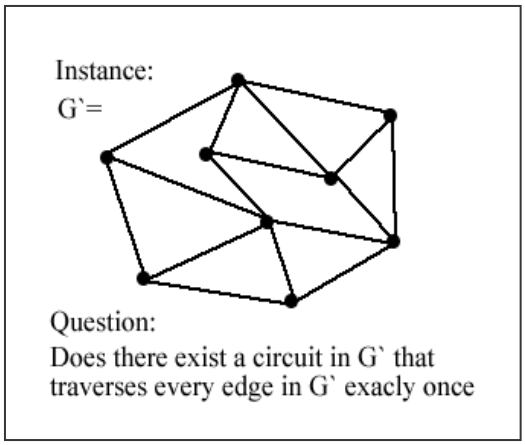

Figure 34 Euler Circuit Problem Instance

As a general rule algorithms are designed to solve a *problem* such that no matter what *problem instance* is input into the algorithm it will accurately answer the question, or solve for the goal. When discussing problem classes, computer scientists usually confine the discussion to decision problems. Decision problems are those problems with a question whose answer is *yes* or *no*. The Euler Circuit Problem given above in Figure 33 is a decision problem.

Computer scientists are often interested in optimization problems, which involve finding the 'best' solution from all feasible solutions. Optimization problems are not decision problems but many have decision problem counterparts. In Figure 35 variations on the problem of finding a path within a graph including a decision problem and an optimization problem are presented.

Decision Problem
Instance: Graph G(*V*, *E*) and vertices A and B within *G*
Question: Does there exist a path within *G* from vertex *A* to vertex *B*?

Non-Decision Problem
Instance: Graph *G*(*V, E*) and vertices *A* and *B* within *G*
Goal: Find a path within *G* from vertex *A* to vertex *B* or report one doesn't exist.

Optimization Problem (A specialized Non-Decision Problem)
Instance: Graph *G*(*V, E*) and vertices *A* and *B* within *G*
Goal: Find the shortest a path (i.e. that traverses the fewest edges) within G from vertex *A* to vertex *B* or report that no such path exists.

Figure 35 Variations on the Path in a Graph Problem

Computation Complexity of an Algorithm

The *computational complexity* of an algorithm for solving a problem, plainly stated, is the measure of how many basic operations that algorithm will require, in the worst case, and is measured as a function of the input size. The computational complexity of an algorithm is important because it gives information about the length of time the algorithm requires to execute and how this time increases as the size of the problem increases. Figure 36 illustrates how different computational complexity functions grow, in terms of number of basic operations, as the size of their input grows.

Computer Scientists usually define a *basic operation* as one transition of a Deterministic Turing Machine (DTM). A DTM is a language decider; given a language, *l*, and a string, *s*, a DTM can decide, *yes* or *no*, weather *s* is a valid string in *l*. *Input size* is the length of the string used to encode the problem input onto the tape of DTM.

When looking at an input size of 100, an algorithm that runs in logarithmic time, say $\log_2(n)$, takes approximately seven operations to complete, whereas a polynomial time algorithm, with say computational complexity of n^2, takes 10000 operations and an algorithm that runs in exponential time, say 2^n, takes approximately 1.2677E30

operations to complete. Figure 36 below shows the growth of several different problem complexities.

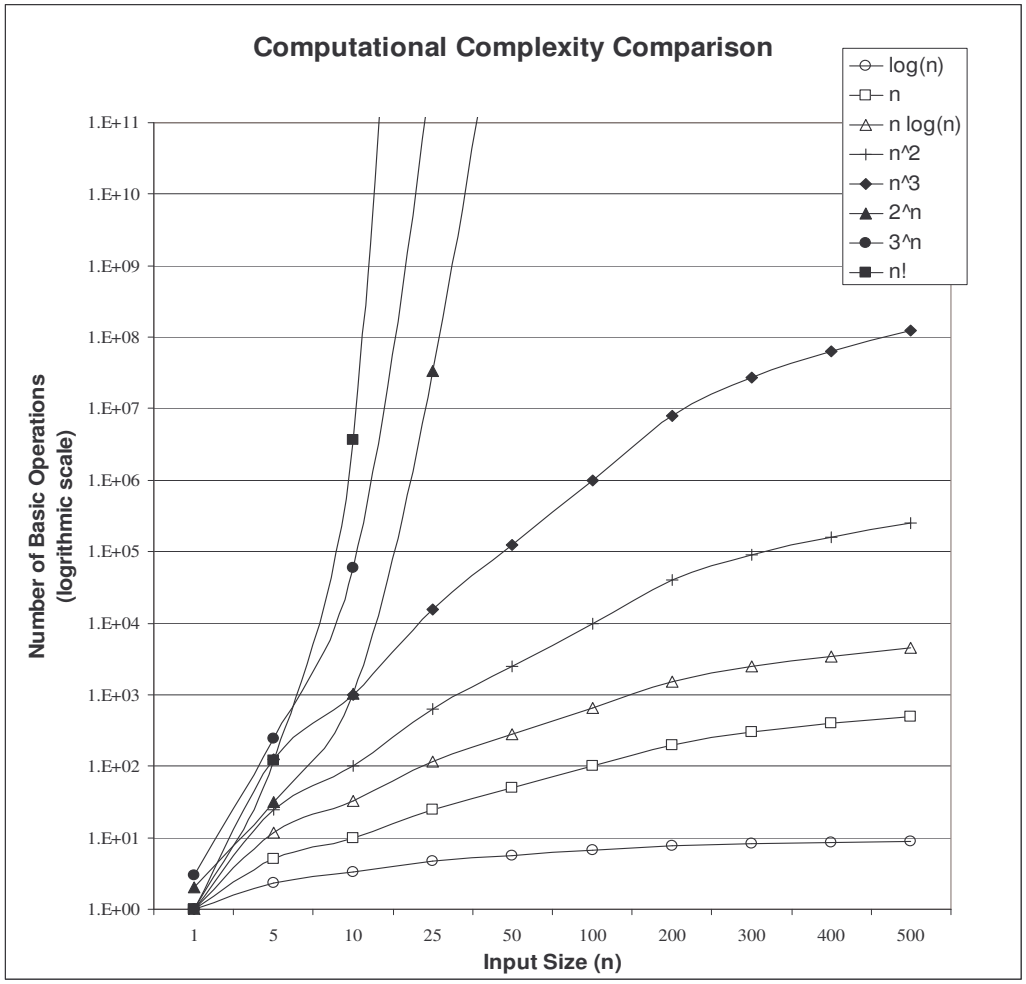

Figure 36 Input Size vs. Number of Operations for Various Complexities

The number of operations required by an algorithm can depend not only on the input size but also on other properties of the input. For example, an algorithm that has a graph as an input parameter may require more, or fewer, operations depending on the configuration of the graph, not just the number of vertices and edges in the graph. Thus the *worst case* for all configurations of the same size is typically discussed. Furthermore, the exact complexity is often difficult to compute and contains more information than is necessary. *Big O* notation is typically used to express an algorithms worse case runtime.

Big O is defined as follows: $f(n) = O(g(n))$ means there are two positive constants c and k such that $0 \leq f(n) \leq cg(n)$ for all $n \geq k$. For example if $f(n) = n^2 + 3n + 4$ it is said that $f(n)$ is $O(n^2)$ because $0 \leq n^2 + 3n + 4 \leq 2n^2$ for all $n > 10$ where $c = 2$. All algorithms in this thesis are discussed in terms of their worst case runtime using *Big O* notation.

The complexity of an algorithm is important both theoretically an in practice. Polynomial time algorithms, especially those with low exponents, are considered *good* whereas an exponential time algorithms are considered *bad* and logarithmic time algorithms are considered *very good*. Of course not all problems have polynomial time algorithms. For some problems it has been proven that they cannot be solved with a polynomial time algorithm, for many others it is not known if they can be solved with a polynomial time algorithm or not.

The Problem Class P

Computer scientists have developed a schema for characterizing *problems* by how time consuming or difficult they are to solve algorithmically. In doing this they have defined a number of problem classes, the first of interest to this thesis is the problem class P. P is an abbreviation for Polynomial. Stated simply, the class P consists of decision problems which can be solved with an algorithm whose runtime is $O(p(l))$, where $p()$ is some polynomial and l is the length of the input to the algorithm.

More formally, the set P is the class of all decision problems which can be solved by a DTM using a polynomial-time algorithm. This means, a problem $\pi \in$ P if and only if there is a reasonable scheme for encoding instances of π into strings, s, there is a DTM that is a decider for the set of all strings s representing encoded instances of π, and there is a polynomial $p()$ such that the DTM when started with string s on its tape halts within

$p(len(s))$ transitions of the DTM with the correct answer of *yes* or *no* on its tape. Here $len(s)$ means the number of tape cells used to write s on the DTM tape. Some examples of problems in P are shown below in Figure 37.

Spanning Tree (ST)
Instance: Graph $G(V, E)$ length $l(e) \in Z_0^+$ for each $e \in E$, and a positive integer K.
Question: Does there exist a spanning tree in G such that length of the tree $\leq K$?

Connected Graph
Instance: Graph G.
Question: Is G connected?

Composite Number (CN) [Discovered to be in P in 2002[1]]
Instance: Positive integer N.
Question: Are there positive integers $m < N$, $n > 1$ such that $N = m*n$

Euler Circuit (EC)
Instance: A connected Graph $G(V, E)$.
Question: Does there exist a circuit in G that traverses every edge in E exactly once?

Figure 37 Some Problems in P

The Problem Class NP

For some problems, there is no known polynomial-time algorithm but for some of these problems a Nondeterministic Polynomial-time algorithm exists. The class of decision problems for which here exists a Nondeterministic Polynomial time algorithmic solution is denoted by NP. Although nondeterministic algorithms are not useful in practice, their value is in characterizing problem theoretically. Simply put, a problem π is in NP if one can quickly (in polynomial time) test whether a guessed witness to any instance of π, that has an answer of *yes*, is indeed a witness (without worrying about how hard it might be to find the solution).

More formally, the previous section made use of a DTM which is a language decider; this section will use a Non-Deterministic Turing Machines (NDTM) which is a language acceptor. A NDTM is and accepter for a language, l, if there exists a path to the

[1] M. Agrawal, N. Kayal, and N. Saxena. "Primes is in P." Annals of Mathematics 160, No. 2 (2004): 781–793

halting state when started with a string in l on its tape and there does not exists a path to the halting state when starting with a string not in l on its tape. Formally a problem π is in NP if and only if there is a reasonable scheme for encoding instances, I, of π into a language, l, there is a Nondeterministic Turing Machine that accepts strings in l and will not accept strings not in l. Finally there is a polynomial $p()$ such that the string s is both guessed/written and accepted (path exists to the halting state) within $O(p(len(s)))$ transitions of the NDTM.

Spanning Tree (ST)
Instance: Graph $G = (V, E)$ length $l(e) \in Z_0^+$ for each $e \in E$, and a positive integer K.
Question: Does there exist a spanning tree such that length of the tree $\leq K$?

Composite Number (CN)
Instance: Positive integer N.
Question: Are there positive integers $m, n > 1$ such that $N = m*n$

Euler Circuit (EC)
Instance: A connected Graph $G(V, E)$.
Question: Does there exist a circuit in G that traverses every edge in E exactly once?

Hamiltonian Circuit (HC)
Instance: Graph $G = (V, E)$.
Question: Does G contain a Hamiltonian circuit?

Traveling Salesman (TS)
Instance: Set C of m cities, distance $d(c_i, c_j) \in Z^+$ for each pair of cities $c_i, c_j \in C$, positive integer K.
Question: Does there exist a tour of C having length K or less?

3-Dimensional Matching (3DM)
Instance: Set $M \subseteq X \times Y \times Z$, where X, Y and Z are disjoint sets having the same number q of elements.
Question: Does M contain a matching, i.e., a subset $M` \subseteq M$ such that $|M`| = q$ and no two elements of M' agree in any coordinate?

Figure 38 Some Problems in NP

Figure 41 provides some examples of problems that are in the class NP. Note that the first three problems above are also in P. Indeed it has been proven that all problems in P are also in NP. It is important to keep in mind that the reason a problem is in NP is different from the reason that same problem is in P.

Figure 39 below shows this relationship between P and NP. The major unanswered question in computer science is does P = NP? In other words, does each problem in NP have a polynomial time algorithm that solves it, correctly answering *yes* or *no* for all instances or to the contrary is there a problem in NP that is not in P? Computer scientists have devoted a lot of effort to this question but have not yet discovered the answer. A subset of NP, the class NP-Complete, which is discussed in the next section, may hold the key to answering this question.

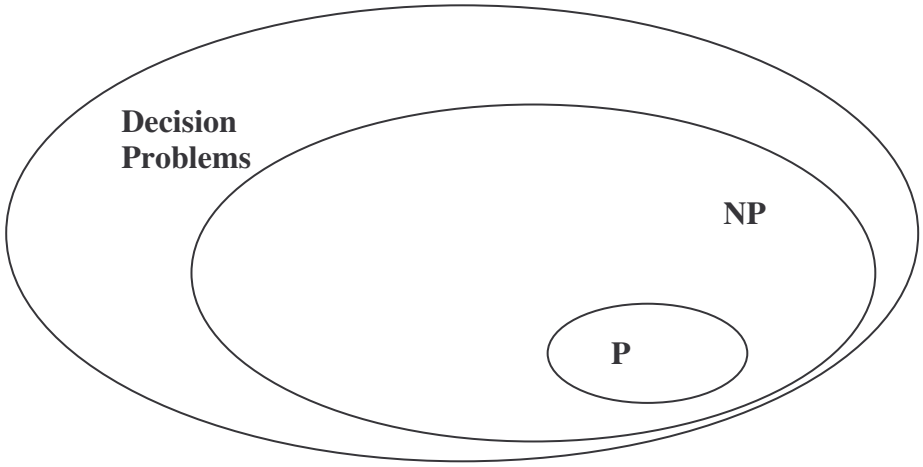

Figure 39 The Relationships between P and NP

The Problem Class NP-Complete

The class of NP-Complete problems is a subset of NP. NP-Complete problems are considered by computer scientist to be the *hardest* problems to solve in NP. This is because if a polynomial algorithm for *any one* of these problems is found it could be used to create a polynomial-time algorithm for *every* problem in NP. This is a very strong statement since NP includes a large number of problems that appear to be extremely time consuming to solve and there are many that have not even been discovered yet.

Formally stated a decision problem π is NP-Complete if and only if π is in NP and from each NP problem to π there exists a Polynomial Transformation that Preserves

Answers (PTPA). In his seminal computer science paper[2], Stephen Cook discovered the first NP-Complete problem, Satisfiability (SAT). He first proved that SAT belongs to NP (i.e. there is a NDTM that is language accepter for SAT in polynomial time in the size of SATs input). Then he provided a generic transformation T_{SAT} that for each pair, π and I^{π} (where $\pi \in$ NP and I^{π} is an instance of π), T_{SAT} maps the pair to an instance I^{SAT} of *SAT* in such a way that the construction of I^{π} to I^{SAT} is a PTPA from π to *SAT*.

Figure 40 is a graphical representation of Cooks powerful theorem. Each arrow represents a PTPA from a known NP problem to SAT. Of course there are many more NP problems than are shown in the diagram in Figure 40. This diagram helps to illustrate that if a polynomial time algorithm is discovered for SAT then it could be used to construct polynomial time algorithm for solving <u>any</u> problem in NP. For example if a polynomial time algorithm, say *f*(), is discovered for SAT then one could use Cook's[3] PTPA to transform an instance of 3DM, I^{3DM}, into an instance of SAT, I^{SAT} (T_{SAT}(3DM, I^{3DM}) = I^{SAT}) in polynomial time, then solve I^{SAT} with the discovered algorithm *f*() to get the correct answer of *yes* or *no* for I^{SAT}. The answer for I^{3DM} is guaranteed to be the same as the solution for I^{SAT} via the PTPA. It is known that two polynomial functions executed sequentially (e.g. $T_{SAT}() + f()$) result in an overall runtime this is still a polynomial. The thus if a polynomial solution to SAT were discovered then P = NP.

[2] Steven Cook, 151-158
[3] Ibid.

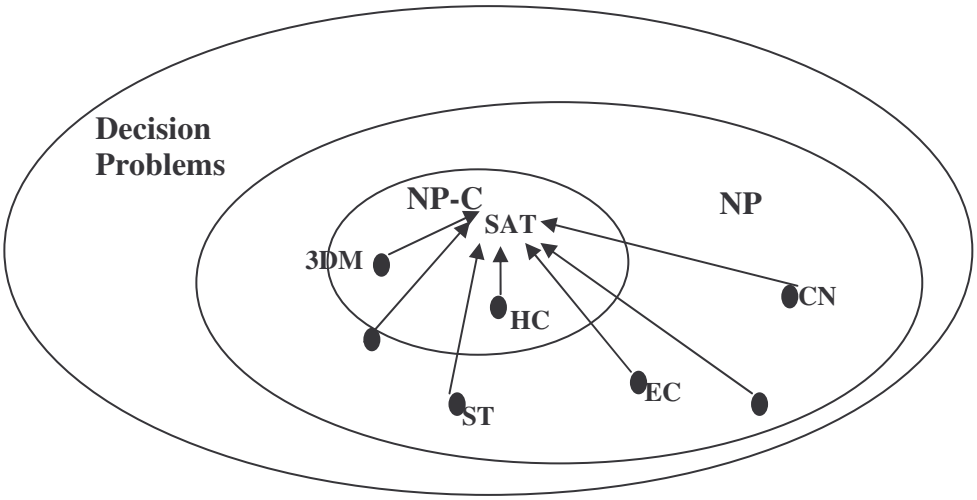

Figure 40 PTPA from all NP Problems to SAT

SAT was the first known NP-Complete problem but it is by no means the only one, there are now thousands of problems which have been proved to be NP-Complete (see Figure 41 for some examples). Because of the definition of NP-Complete, all NP-Complete problems have the same property as SAT in that if there is a polynomial algorithm for any one of these NP-Complete problems, then there is a polynomial algorithm for every problem in NP.

Hamiltonian Circuit (HC)
Instance: Graph $G = (V, E)$.
Question: Does G contain a Hamiltonian circuit?

Traveling Salesman (TS)
Instance: Set C of m cities, distance $d(c_i, c_j) \in Z^+$ for each pair of cities $c_i, c_j \in C$, positive integer K.
Question: Is there a tour of C having length K or less?

3-Dimensional Matching (3DM)
Instance: Set $M \subseteq X \times Y \times Z$, where X, Y and Z are disjoint sets having the same number q of elements.
Question: Does M contain a matching, i.e., a subset $M` \subseteq M$ such that $|M`| = q$ and no two elements of M' agree in any coordinate?

Satisfiability (SAT)
Instance: Set U of variables, collection C of clauses over U.
Question: Is there a satisfying truth assignment for C?

3-Satisfiability (3SAT)
Instance: Set U of variables, collection C of clauses over U such that each clause $c \in C$ has $|c| = 3$.
Question: Is there a satisfying truth assignment for C?

Vertex Cover (VC)
Instance: Graph $G = (V, E)$, positive integer $K \leq |V|$.
Question: Is there a vertex cover of size K or less for G?

Figure 41 Some NP-Complete Problems

To prove that an NP problem, Π, is NP-Complete it must be shown that from each NP problem there is a PTPA to Π. Proving this would be a daunting task if not for Cook's Theorem[4].

Because of Cook's powerful theorem, to show a new problem, $\Pi \in$ NP, is NP-Complete it suffices to show that there exists a PTPA from SAT to the new problem, Π (i.e. $T_{\Pi}(\text{SAT}, I^{SAT}) = I^{\Pi}$). Here is why, suppose there exists a PTPA from SAT to the new problem, Π, this means every problem instance of SAT, I^{SAT}, can be transformed to an instance of Π, I^{Π}, such that the answer is preserved. Now for any NP problem, π, there is a PTPA from π to SAT (the restriction of T_{SAT} to π). It maps instances, I^{π}, of π to

[4] Steven Cook, 155-158.

instances of SAT, i.e. $T_{SAT}(\pi, I^{\pi}) = I^{SAT}$, via Cook's Theorem. Combine the two PTPAs together, $T_{SAT}(\pi, I^{\pi}) = I^{SAT}$ and $T_{\Pi}(\text{SAT}, I^{SAT}) = I^{\Pi}$ and every problem instance, I^{π}, in NP results in an instance Π, $T_{\Pi}(\text{SAT}, T_{SAT}(\pi, I^{\pi})) = I^{\Pi}$. The results of the two transformations result in a PTPA from Π to π. It can be assured that the resulting PTPA is polynomial based on the transitive property of polynomial transformability, i.e. composing two (or more) polynomial time algorithms produces an overall runtime that is also polynomial. Since π was an arbitrary NP problem, it follows that if PTPA from SAT to Π exist then Π is NP-Complete.

Cook himself and later Computer Scientists, notable Karp, demonstrated that other problems in NP are NP-Complete by creating PTPAs from SAT to these other problems. Because of this work, showing a new problem Π is NP-Complete became easier because rather than show a new PTPS directly from SAT to the new problem Π, in NP, it suffices to show there exists a PTPA from some known NP-Complete problem to the new problem, Π. In composing the new PTPA with the known PTPA from SAT to the known problem one gets a PTPA from SAT to Π.

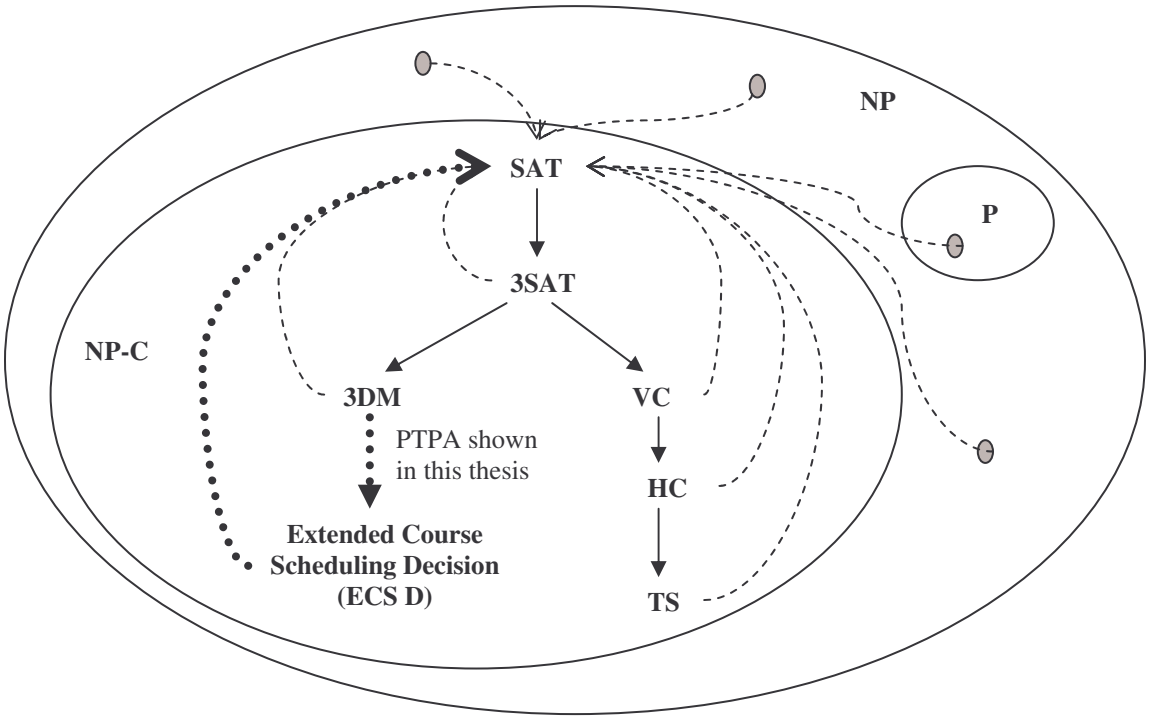

Figure 42 NP-Complete Hierarchical Tree

This complex relationship among NP-Complete problems is indicated in Figure 42 where each arrow represents a PTPA; the solid lines represent demonstrated PTPA presented in well known computer science papers, the dashed arrows are the PTPA provided by Cook's Theorem, and the two bold dotted arrows represent PTPAs which are shown to exit in this thesis. The bold dotted arrow from the Extended Course Scheduling Decision (ECSD) Problem to SAT is implied when it is shown that ECSD $\in$ NP. This proof appears in Appendix B. The second bold dotted arrow from 3DM to ECS represents the PTPA from 3DM to ECS that is proven to exist in Chapter 4.

APPENDIX B

DEMONSTRATION THAT ECS DECISION PROBLEM IS IN NP

To prove that ECS Decision Problem is in NP we must describe two things, first a reasonable scheme for encoding instances of ECS Decision Problem into strings, σ, and secondly a Non-Deterministic Turing Machine (NDTM), *M**, which is an accepter for the set of all strings, σ, representing encoded instances of ECS Decision Problem for which the answer is *yes*

Furthermore, we must demonstrate that there is a polynomial, *P*, having the property that for each string, σ, accepted by *M** there is an accepting computation for that string for which the number of transitions in the computation is bounded above by $P(\lambda(\sigma))$; where $\lambda(\sigma)$ (i.e. the length of the string, σ).

Encoding Scheme for ECS Decision Problem

From the definition of the ECS Decision Problem, Figure 5, the following instance variables must be encoded: the sets *H*, *P*, *C* and *R*; the collection of sets $A(p)$, $A(c)$ and $A(r)$; the collection of numbers $L(p)$, $S(c)$ and $U(r)$; the functions $WTT(p,c)$ and $RS(c,r)$. There is also the scheduling function $f(p, c, h, r)$ which is to be guessed and checked.

An instance of the ECS Decision Problem is provided below in Figure 43 (note this same instance is found in Figure 17, section 2.5) which shows the inputs as stated in the previous paragraph and it also shows a witness. This problem will be used for the encoding example for ease of understanding.

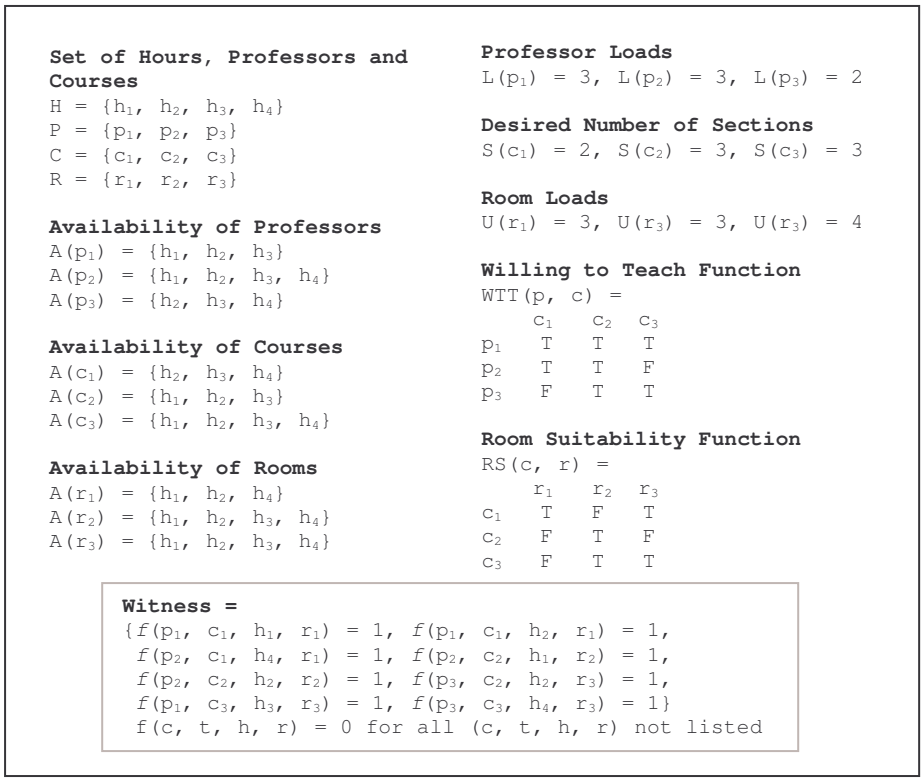

Figure 43 ECS Decision Instance

The machine *M**, which will be described below, has the following alphabet Γ = {**H, h, 1, 0, P, p, C, c, R, r, A, L, q, S, s, U, u, W, t, f, E, F, *, x, @, b, $, ^**} where **b** is the blank, note some of the alphabet is not used in input string. The encoding in the ECS Decision problem inputs from Figure 43 is shown in Figure 44. The various components are as follows:

Encoding of the Hours, *H* (the binary number identifying hour *h* is a fixed length of size $L_{en}(|H|)$):

Hh001h010h11h100

Encoding of the Professors, *P* (the binary number identifying professor *p* is a fixed length of size $L_{en}(|P|)$):

Pp01p10p11

Encoding of the Courses, C (the binary number identifying course c is a fixed length of size $L_{en}(|C|)$):

Cc01c10c11

Encoding of the Rooms, R (the binary number identifying room r is a fixed length of size $L_{en}(|R|)$):

Rr01r10r11

Encoding of the Professor Availability sets, $A(p)$:

Ap01h001h0100h011Ap10h001h010h011h100Ap11h010h011h100

Encoding of the Course Availability sets, $A(c)$:

Ac01h010h011h100Ac10h001h010h011Ac11h001h010h011h100

Encoding of the Room Availability sets, $A(r)$:

Ar01h001h010h100Ar10h001h010h011h100Ar11h001h010h011h100

Encoding of Professor Loads $L(p)$ (the binary number representing the number of sections that professor p can teach is a fixed length of size $L_{en}(M_{ax}(L(p)))$:

Lp01q11Lp10q10Lp11q10

Encoding of the Desired Number of Sections $S(c)$ (the binary number representing the number of sections that course c desired is a fixed length of size $L_{en}(M_{ax}(S(c)))$:

Sc01s10Sc11s11Sc11s11

Encoding of the Room Loads $U(r)$ (the binary number representing the number of sections that room r can accommodate is a fixed length of size $L_{en}(M_{ax}(U(r)))$:

Ur01u011Ur10u011Ur11u100

Encoding of the Willing to Teach function $WTT(p, c)$:

Wp01c01c10c11Wp10c01c10Wp11c10c11

Encoding of the Room Suitability function *RS*(*c*, *r*):

Ec01r01r11Ec10r10Ec11r10r11

Lastly there is the encoding of the blank scheduling function *f*(*p*, *c*, *h*, *r*) where both the x's and *'s are place holders for use by NDTM *M**. For brevity not all 108 possible quadruples (*p*, *c*, *h*, *r*) are shown below, the ellipsis '…' indicates those quadruples not shown:

Fp01*c01*h001r01*xFp01*c01*h001r10*xFp01*c01*h001r11*x

Fp01*c01*h010r01*xFp01*c01*h010r10*xFp01*c01*h010r11*x

Fp01*c01*h011r01*xFp01*c01*h011r10*xFp01*c01*h011r11*x

…

Fp11*c11*h011r01*xFp11c11*h011r10*xFp11*c11*h011r11*x

Fp11*c11*h100r01*xFp11c11*h100r10*xFp11*c11*h100r11*x

Below in Figure 44 is the string, σ, which is the encoding of the inputs for the ECS Decision Problem, found in Figure 43. Note the spaces have been added to aid the human eye but are not part of the actual encoded string.

```
H h001 h010 h011 h100 P p01 p10 p11 C c01 c10 c11 R r01 r10 r11
Ap01 h001 h010 h011 Ap10 h001 h010 h011 h100 Ap10 h010 h011 h100
Ac01 h010 h011 h100 Ac10 h001 h010 h011 Ac11 h001 h010 h011 h100
Ar01 h001 h010 h100 Ar10 h001 h010 h011 h100 Ar11 h001 h010 h011
h100 Lp01q11 Lp10q10 Lp1ql10 Sc01s10 Sc11s11 Sc11s11 Ur01u011
Ur10u011 Ur11u100 Wp01c01c10c11 Wp10c01c10 Wp11c10c11 Ec01r01r11
Ec10r10 Ec11r10r11 Fp01*c01*h001r01*x Fp01*c01*h001r10*x
Fp01*c01*h001r11*x Fp01*c01*h010r01*x Fp01*c01*h010r10*x
Fp01*c01*h010r11*x Fp01*c01*h011r01*x Fp01*c01*h011r10*x
Fp01*c01*h011r11*x
   …
Fp11*c11*h011r01*x Fp11*c11*h011r10*x Fp11*c11*h011r11*x
Fp11*c11*h100r01*x Fp11*c11*h100r10*x Fp11*c11*h100r11*x @
```

Figure 44 Encoded String, σ, for ECS Decision Instance of Figure 43

Complexity of the Encoding

Let $h = |H|$, $p = |P|$, $c = |C|$, $r = |R|$, $l = M_{ax}(L(p))$, $s = M_{ax}(S(c))$, and $u = M_{ax}(U(r))$. Note that the input contains sets of size h, p, c, r, and numbers whose lengths are bounded above by the following: $log(h)$, $log(p)$, $log(c)$, $log(r)$, $log(l)$, $log(s)$, and $log(u)$.

The encoding of H $= O(h*log(h))$

The encoding of P $= O(p*log(p))$.

The encoding of C $= O(c*log(c))$.

The encoding of R $= O(r*log(r))$.

The encoding of the $A(p)$ $= O(p*(log(p)+(h*log(h)))$.

The encoding of the $A(c)$ $= O(c*(log(c)+(h*log(h)))$.

The encoding of the $A(r)$ $= O(r*(log(r)+(h*log(h)))$.

The encoding of the $L(p)$ $= O(p*(log(p)+log(l)))$.

The encoding of the $S(c)$ $= O(c*(log(c)+log(s)))$.

The encoding of the $U(r)$ $= O(r*(log(r)+log(u)))$.

The encoding of $WTT(c, r)$ $= O(p*(log(p)+c*log(c)))$.

The encoding of $RS(c, r)$ $= O(c*(log(c)+r*log(r)))$.

The encoding of $f(p, c, h, r)$ $= O(p*log(p)*c*log(c)* h*log(h)* r*log(r))$.

As shown above, each portion of the string, σ, is polynomial in the size, or length, of the problem input. The total length of the encoding of string, σ, is the sum of all its portions, thus the total length of the encoded string, λ(σ) (or λ for brevity), is polynomial in the size of the problem inputs which includes sets of size h, p, c, r, and numbers whose lengths are bounded by $log(h)$, $log(p)$, $log(c)$, $log(r)$, $log(l)$, $log(s)$, and $log(u)$. This is considered a reasonable encoding scheme.

Top Level Description of NDTM

The NDTM *M** has six different phases, or submachines, labeled ***A*** through ***F*** in Figure 45 below. The first submachine, ***A***, is the non-deterministic phase of the NDTM which will guess a potential timetable. ***A*** is the only phase of the NDTM that is non-deterministic, all other phases are deterministic. The next four submachines, ***B***, ***C***, ***D*** and ***E***, deterministically check whether the potential timetable respects the various constraints, Availability, Physical, Academic, and Bounding respectively. In each, if the constraints are met control passes to the next submachine otherwise *M** hangs. Then the last submachine, ***F***, adds up the number of sections scheduled. If the number of section is equal to the total number of sections desired then the machine *M** halts in the accepting state, $, if not, submachine ***F*** hangs.

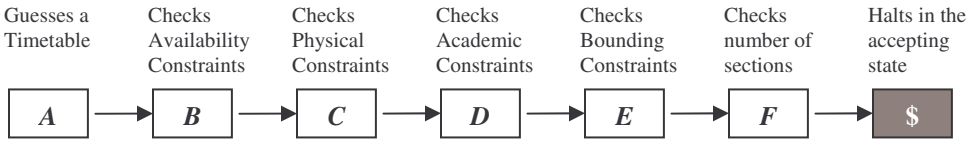

Figure 45 NDTM Flow

Submachine A

Submachine ***A*** is the guessing phase. It moves through the string, σ, looking for `x`'s. Whenever an `x` is found ***A*** will, non-deterministically, replace it with either a `t` or an `f`, until it comes to the end of string σ, as indicated by the `@`. This submachine can be thought of it as "guessing" a potential timetable, *f*. For those quadruples (*p*, *c*, *h*, *r*) for which the `x` is replaced by a `t`, $f(p, c, h, r) = 1$ (i.e. a section is scheduled), Whereas the if `x` for the quadruple (*p*, *c*, *h*, *r*) is replaced by a `f`, then $f(p, c, h, r) = 0$. For example, the first substring `Fp01c01h001r01x` represents $f(p_1, c_1, h_1, r_1)$. If ***A*** overwrites that particular `x` with a `t` that would mean $f(p_1, c_1, h_1, r_1) = 1$. On the other hand, if ***A***

overwrites that particular x with an f that would mean $f(p_1, c_1, h_1, r_1) = 0$. Please note that the only thing ***A*** is "guessing" is the result $f(p, c, h, r)$ for all valid inputs C×P×H×R. The number of transitions in each computation of submachine ***A*** is $O(\lambda)$.

As stated above, ***A*** is non-deterministic. Each encoded *yes* instance has a valid timetable, *f*, and therefore there is indeed a path through ***A*** which can write this valid timetable. There are other paths through ***A*** that write t's and f's that do not form a valid timetable for the encoded instance but they are of no consequence. For the example in Figure 43 there is a path through ***A*** such that after it is completed the tape string, σ, will be modified to be as shown in Figure 46.

```
H h001 h010 h011 h100 P p01 p10 p11 C c01 c10 c11 R r01 r10 r11
Ap01 h001 h010 h011 Ap10 h001 h010 h011 h100 Ap10 h010 h011
h100 Ac01 h010 h011 h100 Ac10 h001 h010 h011 Ac11 h001 h010
h011 h100 Ar01 h001 h010 h100 Ar10 h001 h010 h011 h100 Ar11
h001 h010 h011 h100 Lp01q11 Lp10q10 Lp11q10 Sc01s10 Sc11s11
Sc11s11 Ur01u011 Ur10u011 Ur11u100 Wp01c01c10c11 Wp10c01c10
Wp11c10c11 Ec01r01r11 Ec10r10 Ec11r10r11 Fp01*c01*h001r01*t
Fp01*c01*h001r10*f Fp01*c01*h001r11*f Fp01*c01*h010r01*t
Fp01*c01*h010r10*f Fp01*c01*h010r11*f Fp01*c01*h011r01*f
Fp01*c01*h011r10*f Fp01*c01*h011r11*f
   …
Fp11*c11*h011r01*f Fp11*c11*h011r10*f Fp11*c11*h011r11*f
Fp11*c11*h100r01*t Fp11*c11*h100r10*f Fp11*c11*h100r11*t @
```

Figure 46 String σ after Submachine *A*

Submachine B

Submachines ***B*** checks whether the guessed function *f*, respects the availability constraints. It performs three tasks for each quadruple (p, c, h, r) for which x has been replaced by an t (i.e. for each section in the potential timetable).

The first task is to check that the particular hour, h, in the quadruple is in the availability list for the particular professor, p. It does this by comparing the h in a section substring, F…t, to all the h's in the Ap substring for professor p. If the first h in

Ap substring does not match it will overwrite the h (not the actual binary identifier) with a ^. ***B*** will continue to compare the Ap's h's with F's h until a match is found at which point the ***B*** will overwrite the * immediately after the p in the section substring F with a ^ and the ^'s in the substring Ap will be overwritten with a h. Otherwise, if no match is found *M** hangs. This task for one section is $O(\lambda^*(h^*log(h)+p^*log(p))) = O(\lambda^2)$, because $(h^*log(h)+p^*log(p))$ is less than λ.

The second task is to check that the particular hour, h, in the quadruple is in the availability list for the particular course, c. It does this by comparing the h in a section substring, F…t, to all the h's in the Ac substring for course c. If the first h in Ac substring does not match it will overwrite the h (not the actual binary identifier) with a ^. ***B*** will continue to compare the Ac's h's with F's h until a match is found at which point the ***B*** will overwrite the * immediately after the c in the section substring F with a ^ and the ^'s in the substring Ac will be overwritten with a h. Otherwise, if no match is found *M** hangs. This task for one section is $O(\lambda^*(h^*log(h)+c^*log(c))) = O(\lambda^2)$, again because $(h^*log(h)+c^*log(c))$ is less than λ.

The third task is to check that the particular hour, h, in the quadruple is in the availability list for the particular room, r. It does this by comparing the h in a section substring, F…t, to all the h's in the Ar substring for room r. If the first h in Ar substring does not match it will overwrite the h (not the actual binary identifier) with a ^. ***B*** will continue to compare the Ar's h's with F's h until a match is found, at which point the ***B*** will overwrite the * immediately after the r in the section substring F with a ^ and the ^'s in the substring Ar will be overwritten with a h. Otherwise, if no match is found *M** hangs. This task for one section is $O(\lambda^*(h^*log(h)+r^*log(r))) = O(\lambda^2)$.

Each of quadruple that has been marked with a `t` by ***A*** has a computational complexity of $O(3\lambda^2) = O(\lambda^2)$ in ***B***. There are at most $p*c*h*r$ such quadruples which is less than the length of the string, σ, which contains all quadruples. Thus the computational complexity of submachine ***B*** on *yes* instances of the ECS Decision Problem is $O(\lambda^3)$.

After submachine ***B*** is complete the string, σ, will look like Figure 47.

```
H h001 h010 h011 h100 P p01 p10 p11 C c01 c10 c11 R r01 r10 r11
Ap01 h001 h010 h011 Ap10 h001 h010 h011 h100 Ap10 h010 h011
h100 Ac01 h010 h011 h100 Ac10 h001 h010 h011 Ac11 h001 h010
h011 h100 Ar01 h001 h010 h100 Ar10 h001 h010 h011 h100 Ar11
h001 h010 h011 h100 Lp01q11 Lp10q10 Lp11q10 Sc01s10 Sc11s11
Sc11s11 Ur01u011 Ur10u011 Ur11u100 Wp01c01c10c11 Wp10c01c10
Wp11c10c11 Ec01r01r11 Ec10r10 Ec11r10r11 Fp01^c01^h001r01^t
Fp01*c01*h001r10*f Fp01*c01*h001r11*f Fp01^c01^h010r01^t
Fp01*c01*h010r10*f Fp01*c01*h010r11*f Fp01*c01*h011r01*f
Fp01*c01*h011r10*f Fp01*c01*h011r11*f
   …
Fp11*c11*h011r01*f Fp11*c11*h011r10*f Fp11*c11*h011r11*f
Fp11^c11^h100r01^t Fp11*c11*h100r10*f Fp11^c11^h100r11^t @
```

Figure 47 String σ after Submachine *B*

Submachine C

Submachine C checks that the physical constraints are respected by the potential guessed timetable. The first physical constraint is that for each pair $(p, h) \in H \times P$ there is at most one pair $(c, r) \in C \times R$ for which $f(p, c, h, r) = 1$. In plain English this means professor, *p*, can only be scheduled to teach one section at any given hour, *h*. The second physical constraint is that each pair $(r, h) \in R \times H$ there is at most one pair $(p, c) \in P \times C$ for which $f(p, c, h, r) = 1$. In plain English this means room, *r*, can only have one section scheduled to be taught in it at any given hour, *h.*

Submachine ***C*** accomplishes this first task by checking that the particular hour, `h`, in the quadruple is in the availability list (`Ap`) for the particular professor, `p`. If a

matching `h` is found in `Ap` list then that `Ap h` is overwritten with a `^` thus indicating that particular hour is no longer available when later sections are checked. If there is no hour, `h`, in the professor's availability substring, `Ap`, matching the hour, `h,` in the section substring then ***C*** hangs otherwise it continues to the second task. The first task for one section is $O(\lambda * p * log(p)) = O(\lambda^2)$.

The second task of ***C*** compares that the particular hour, `h`, in the quadruple is in the availability list (`Ar`) for the particular room, `r`. If a matching `h` is found in `Ar` list then that `Ar h` is overwritten with a `^` thus indicating that particular hour is no longer available when later sections are checked. If there is no hour, `h`, in the room's availability substring, `Ar`, matching the hour, `h,` in the section substring then ***C*** hangs otherwise it continues the next section. The second task for one section is $O(\lambda * r * log(r)) = O(\lambda^2)$.

Each of the section has a computational complexity of $O(2\lambda^2) = O(\lambda^2)$. There are potentially there are $p*c*h*r$ quadruples which, again, is less then the length of the string, σ. Thus the computational complexity of submachine ***C*** on *yes* instances of the ECS Decision Problem is $O(\lambda^3)$.

After submachine ***C*** is complete the string, σ, will look like Figure 48.

```
H h001 h010 h011 h100 P p01 p10 p11 C c01 c10 c11 R r01 r10 r11
Ap01 ^001 ^010 ^011 Ap10 ^001 ^010 h011 ^100 Ap10 ^010 ^011
^100 Ac01 h010 h011 h100 Ac10 h001 h010 h011 Ac11 h001 h010
h011 h100 Ar01 ^001 ^010 ^100 Ar10 ^001 ^010 h011 h100 Ar11
h001 ^010 ^011 ^100 Lp01q11 Lp10q10 Lp11q10 Sc01s10 Sc11s11
Sc11s11 Ur01u011 Ur10u011 Ur11u100 Wp01c01c10c11 Wp10c01c10
Wp11c10c11 Ec01r01r11 Ec10r10 Ec11r10r11 Fp01^c01^h001r01^t
Fp01*c01*h001r10*f Fp01*c01*h001r11*f Fp01^c01^h010r01^t
Fp01*c01*h010r10*f Fp01*c01*h010r11*f Fp01*c01*h011r01*f
Fp01*c01*h011r10*f Fp01*c01*h011r11*f
   …
Fp11*c11*h011r01*f Fp11*c11*h011r10*f Fp11*c11*h011r11*f
Fp11^c11^h100r01^t Fp11*c11*h100r10*f Fp11^c11^h100r11^t @
```

Figure 48 String σ after Submachine *C*

Submachine D

Submachine ***D*** checks whether the guessed timetable adheres to the Academic Constraints. It performs two tasks on each quadruple which has been tagged with a `t`. The first task will check whether the particular course, `c`, is in the list of courses, `Wp`, that the particular professor, `p`, is willing to teach. And the second task will check whether the particular room, `r`, is in the list of rooms, `Ec`, that the particular course, `c`, is suitable to be taught in.

The first task of ***D*** compares the course, `c`, in a section substring, `F…t`, to all the `c`'s in the `Wp` substring for professor, `p`, corresponding the professor in the section substring `F…t`. If a matching `c` is found this means professor, `p`, is indeed willing to teach the course, `c`, as scheduled in the given section, `F`, and ***D*** continues to the next tasks. Otherwise it hangs. The first task for one section has a computational complexity of $O(\lambda*(p*log(p) + c*log(c)) = O(\lambda^2)$.

The second task of ***D*** compares the room, `r`, in a section substring, `F…t`, to all the `r`'s in the `Ec` substring for course, `c`, corresponding the course in the section substring `F…t`. If a matching `r` is found this means room, `r`, is indeed a suitable room for course,

c, as scheduled in the given section, F, and ***D*** continues to the next section. Otherwise it hangs. The second task for one section has a computational complexity of $O(\lambda*(c*log(c) + r*log(r)) = O(\lambda^2)$.

Each of the section has a computational complexity of $O(2\lambda^2) = O(\lambda^2)$. There are potentially $p*c*h*r$ quadruples which, again, is less than the length of the string, σ. Thus the computational complexity of submachine ***D*** on *yes* instances of the ECS Decision Problem is $O(\lambda^3)$. There is no change to the string σ, after ***D*** is complete.

Submachine E

Submachine ***E*** is designed to check whether the guessed witness respects the bounding constraints. ***E*** has three tasks the first is the count the number quadruples marked with a t for each professor and compares that count to the load bound of that professor. The second and third tasks check the course bound and room bound respectively.

The first task starts with the first professor loads substring Lp and counts the number of sections, F…t, scheduled for that professor, p. ***E*** then compares the number of sections counted to the professors load q, if the values match ***E*** continues to the next professor load until all professors have been completed. Otherwise ***E*** hangs. The computational complexity of this task it is $O(\lambda*(p*log(p)) = O(\lambda^2)$.

The second task starts with the first number of desired courses substring Sc and counts the number of sections, F…t, scheduled for that course, c. ***E*** then compares the number of sections counted to the desire number of sections s, if the values match ***E*** continues to the next course load until all courses have been completed. Otherwise ***E*** hangs. The computational complexity of this task it is $O(\lambda*(c*log(c)) = O(\lambda^2)$.

The last task starts with the first room loads substring `Ur` and counts the number of sections, `F…t`, scheduled for that room, `r`. ***E*** then compares the number of sections counted to the room load `u`, if the values match ***E*** continues to the next room load until all rooms have been completed. Otherwise ***E*** hangs. The computational complexity of this task it is $O(\lambda*(r*log(r)) = O(\lambda^2)$.

Thus the computational complexity of submachine ***E*** on *yes* instances of the ECS Decision Problem is $O(3\lambda^2) = O(\lambda^2)$. There is no change to the string σ, after ***E*** is complete.

Submachine F

The last submachine, ***F***, counts all the quadruples tagged with a `t` in the guessed timetable and compares it to the total number of sections desired (i.e. the sum of each courses desired number of sections). This submachine starts with the first number of desired courses substring `Sc` and sums up all the courses desired sections values `s`. It then goes through the timetable substring `F` counting all the `t`s. Finally it compares the two values, if they match it overwrites the end of sting character `@` with a `$` and halts in the accepting state. Otherwise it hangs. The computational complexity of submachine ***F*** on *yes* instances of the ECS Decision Problem is $O(2\lambda) = O(\lambda)$. There is no change to the string σ, after ***F*** is complete.

After submachine ***F*** is complete the string, σ, will look like Figure 48, and $M*$ will be halted in the accepting state. Note this will only be the case for *yes* instances of the ECS Decision Problem.

```
H h001 h010 h011 h100 P p01 p10 p11 C c01 c10 c11 R r01 r10 r11
Ap01 ^001 ^010 ^011 Ap10 ^001 ^010 h011 ^100 Ap10 ^010 ^011
^100 Ac01 h010 h011 h100 Ac10 h001 h010 h011 Ac11 h001 h010
h011 h100 Ar01 ^001 ^010 ^100 Ar10 ^001 ^010 h011 h100 Ar11
h001 ^010 ^011 ^100 Lp01q11 Lp10q10 Lp11q10 Sc01s10 Sc11s11
Sc11s11 Ur01u011 Ur10u011 Ur11u100 Wp01c01c10c11 Wp10c01c10
Wp11c10c11 Ec01r01r11 Ec10r10 Ec11r10r11 Fp01^c01^h001r01^t
Fp01*c01*h001r10*f Fp01*c01*h001r11*f Fp01^c01^h010r01^t
Fp01*c01*h010r10*f Fp01*c01*h010r11*f Fp01*c01*h011r01*f
Fp01*c01*h011r10*f Fp01*c01*h011r11*f
   …
Fp11*c11*h011r01*f Fp11*c11*h011r10*f Fp11*c11*h011r11*f
Fp11^c11^h100r01^t Fp11*c11*h100r10*f Fp11^c11^h100r11^t $
```

Figure 49 String σ after Submachine *F*

Computational Complexity of M* on yes instances of ECS Decision Problem

The overall complexity of M* is the sum of all the submachine complexities. Thus it is $O(\lambda+\lambda^3+\lambda^3+\lambda^3+\lambda^2+\lambda) = O(\lambda^3)$ which is polynomial in the size of the string σ, which is in turn polynomial in the size inputs, Therefore ECS Decision Problem is in NP.