New Integer Linear Programming Approaches for Course Timetabling

Natashia Boland[1*], Barry D. Hughes[1], Liam T.G. Merlot[2], and Peter J. Stuckey[3]

[1]Department of Mathematics and Statistics,
The University of Melbourne, Victoria 3010, Australia
natashia@ms.unimelb.edu.au, hughes@ms.unimelb.edu.au

[2]Quantm, 4/333 Flinders Lane,
Melbourne 3000, Australia
liam.merlot@quantm.net

[3]Department of Computer Science and Software Engineering,
The University of Melbourne, Victoria 3010, Australia
pjs@cs.mu.oz.au

[*]Corresponding author: fax +61 3 8344 4599

April 24, 2006

(To be submitted to *Computers and Operations Research*)

Abstract

The most complete form of academic timetabling problem is the population and course timetabling problem. In this problem, there may be multiple classes of each subject, and the decision on which students are to constitute each class is made in concert with the decision on the timetable for each class. In order to solve this problem, it is normally simplified or decomposed in some fashion. One simplification commonly used in practice is known as *blocking*: it is assumed that the classes can be partitioned into sets of classes (or *blocks*) that will be timetabled in parallel. This restricts clashing to occur only between classes in the same block, and essentially removes the timetabling aspect of the problem, which can be carried out once the blocks are constituted and the classes populated. The problem of constituting the blocks and populating the classes, known as the *course blocking and population problem* is nevertheless a challenging problem, and provides the focus of this paper. We demonstrate, using data provided by a local high school, that integer linear programming approaches can solve the problem in a matter of seconds. Key features include remodelling to remove symmetry caused by students with identical subject selection, and the observation that in practice, only integrality of the block composition variables needs to be enforced; the class population aspects of the model have strong integrality properties.

1 Introduction

Academic timetabling is a challenging problem faced by educational institutions of many types. Academic timetabling problems tend either to be large (such as the timetabling problems faced by universities) or small but tightly constrained (such as the timetabling problems faced by high schools). This paper will consider a quite general form of the problem, but the approach taken is more usually used in high schools.

The most complete form of academic timetabling problem is the *population and class timetabling problem* (PCTP). The PCTP requires the timetabler to determine the population of each class, for subjects with multiple classes, in addition to finding a session in the timetable for every lesson of every class.

A popular simplification for the population and class timetabling problem is known as *blocking*. Under blocking, it is assumed that the set of classes can be partitioned into subsets, with all classes in a subset timetabled in parallel. Each subset is known as a *block*. It is assumed that the blocks will induce a partition of the sessions available in the timetable; all classes in a block will be timetabled within the set of sessions corresponding to the block, and none of these sessions can be used for classes in other blocks. This process restricts clashes to occur within blocks, and in effect removes the timetabling aspect of the problem. It thus decomposes the problem into two independent problems: the class blocking and population problem (CBPP) and the block timetabling problem (BTP). The former determines the blocks (i.e. the partition of classes) and populates the classes; the latter determines the timetable, by partitioning the sessions in the timetable into disjoint sets, one for each block.

In this paper, we focus on the hardest aspect of the blocking approach: solving the CBPP. Our paper is structured as follows. We first review the literature on the PCTP and articulate the contribution of this paper, in Section 2. In Section 3, we present a formal definition of the CBPP as a feasibility problem, namely that of finding a clash-free solution. Two variants on an integer linear programming formulation are then presented. In the first variant, (binary) variables are included for every student. In the second, symmetry caused by students with identical subject selections is removed, by the use of general integer variables. We also present three alternative approaches for relaxing the feasibility problem as an optimization problem, with penalties on infeasibility. Now

the general integer student variable models essentially aggregate students; it is not obvious, *a priori*, that a solution to these models can be decomposed to feasible solutions for individual students. In Section 4 we prove that this is indeed possible, showing the correspondence between the general integer and binary models. Using data from a local high school that uses the blocking approach, we compare the performance of both the binary and general integer models, and all three infeasibility penalty models, reporting the results in Section 5. We also test enforcing integrality on only the block constitution variables, and find that, for these data sets, this is sufficient to yield fully integer solutions. The results demonstrate that under priority branching on the block constitution variables, one of the penalty models with general integer variables performs consistently better than the other variations, solving all instances in under 4 seconds.

2 Review of the Literature on the PCTP

Although many papers have been written on academic timetable problems, most have assumed that all classes have already been allocated, that is, that the students constituting the population of each class are already decided. Only a small proportion of the educational timetabling papers have investigated the complete PCTP variant of the timetabling problem (either for universities or schools). Typically researchers looking at the PCTP for university data sets used the population-timetabling decomposition and the papers looking at the PCTP for secondary school data sets tended to use a blocking simplification (with the notable exception of Willemen [18], who uses a population-timetabling decomposition for a school problem).

Of those that do consider the PCTP, the majority [1, 2, 3, 6, 7, 9, 10, 13, 11, 14, 18] decompose the problem into the two parts of populating the classes, and timetabling the classes, often with some form of iteration between the two subproblems. The techniques applied are usually heuristics, meta-heuristics and constraint programming, often used in combination. Tabu search is the most popular of the metaheuristics, used by Alvarez-Valdes et al. [1, 2], Hertz [9], Hertz and Robert [10, 13], and, in conjunction with constraint programming, by Willemen [18]. Constraint programming is also relatively popular; in addition to Willemen [18], it is used by Ferland and Fleurent [7] in conjunction with sequential construction heuristics, and by Rudova and Matyska [14]. Other

techniques include the use of graph colouring heuristics, employed by Carter [6] in conjunction with other heuristics, and by Lewandowski [11], and hill-climbing, used by Aubin and Ferland [3].

There appears to be no work that provides exact solutions to the PCTP. Furthermore, it would appear to be quite difficult to compare the methods developed for PCTP. Whilst most of the papers listed above give the results of tests on real data, there has been no attempt to compare different methods on the same data sets, and no standard data set has yet been developed for testing.

Several papers in the literature have approached the CBPP [4, 15, 16, 17, 19]. All but one of these, (the exception is the paper of van Kesteren [17]), solve the problem by decomposing it along similar lines to the PCTP decomposition: the two aspects of the problem, constituting the blocks, and populating the classes, are treated separately, possibly with iteration between the two subproblems. Thus the solutions methods are at best heuristic in terms of solving the CBPP. Furthermore, with the exception of Banks et al.[4], who employ pure constraint programming, the techniques applied are generally heuristic, or some combination of constraint programming and heuristic: Tripathy's approach [16] is purely heuristic, while both Sampson et al. [15] and Wood and Whitaker [19] combine simulated annealing with constraint programming.

Again, it would appear to be quite difficult to compare the methods developed for CBPP: there has been no attempt to compare different methods on the same data sets, and no standard data set. For a more detailed review of this work, including information on the size of problem instances solved, we refer the interested reader to the thesis of Merlot [12].

As far as we are aware only work published that solves the CBPP without further decomposition is that of van Kesteren [17]. This paper develops a highly tailored combination of branch-and-bound and constraint programming, leading to a rather complex algorithm, which we believe is an exact method. Results for the CBPP are reported for two problems, one with with 97 students (apparently a hard problem) and another with 131 students (an easier problem). It is difficult to assess the effectiveness of his approach from the results in [17]: depending on the data and version of the algorithm used, solution times ranges from as little as 15 seconds for the easier problem, to times in the range 900 to over 7000 seconds for the harder problem; of course these times are for an old machine. Numbers of branch-and-bound nodes are not given, but the numbers of restarts

due to hitting a local minima in the search tree with no improvement for two seconds, are given. These were 4 for the easier problem and in the range 152–409 for the harder problem. The number of branch-and-bound nodes and/or search choice points could be expected to exceed these figures.

Whilst only van Kesteren [17] solves the CBPP without decomposition, two papers, those of Wood and Whitaker [19] and Sampson et al. [15], do present exact integer programming models. Neither actually *solve* the mathematical programming models they present, in both cases believing them to be intractable.

Wood and Whitaker [19] give a nonlinear integer programming model, with binary variables used to allocate students and teachers to classes and classes to blocks. The nonlinearity occurs in the clash constraints. Although standard linearization techniques *could* be applied, these would greatly increase the number of variables required: Wood and Whitaker [19] did not solve either their model, or any linearization of it[1]. Instead their model is decomposed into a series of subproblems that are solved heuristically.

Sampson et al. [15] consider a PCTP that arises in course timetabling for an MBA program. A special feature of their problem is that each class only requires one lesson[2]. In this context, one can view the role of a session as equivalent to that of a block. Thus the model of Sampson et al. [15] can be reinterpreted as a model for the CBPP. Sampson et al. [15] use binary variables[3] similar to those we will use to indicate constitution of each block. The other variables are equivalent to the binary individual student class allocation variables we will discuss in Section 3. Sampson et al. [15] also use a 'preference' value for each student's preference for each subjects in his/her program.

The major difference between the models of Wood and Whitaker [19] and Sampson et al. [15] is that the model used by Sampson et al. [15] does not explicitly allocate students to classes (but rather to blocks). This allows the constraints preventing student clashing to be linear. However, Sampson et

[1] '. . . it would be extremely difficult, if not impossible, to find a solution to a real timetabling problem with the above constraints' [19].

[2] This fact is never explicitly stated in the paper. However the model given appears to assume that each class has only one lesson.

[3] The paper is ambiguous about whether the class variables are integer or binary; they are defined as both in different parts of the paper. The constraints used force the variables to be binary, but the variable definitions themselves are given as general integer. We assume that these variables are binary.

al. [15] still feel the need to decompose the problem further[4], and solve it using a population-block constitution style decomposition.

The contribution we make here in this paper is to demonstrate that—with the right model—standard integer linear programming packages can be used to solve the CBPP *exactly*. The approach is far simpler to that of van Kesteren [17], and we obtain solutions on problem instances of similar size, in a matter of seconds, indeed, without the need for any branching at all. With an appropriate re-interpretation, the first variant of our model can be shown to be quite similar to that of Sampson et al. [15]. However we also investigate a second variant, in which symmetry is reduced, by aggregating variables for students with the same subject selection, and demonstrate the efficacy of priority branching. We note that in a somewhat different context, Willemen [18] also combines students who take identical programs of subjects. However when allocating students with the same program to classes, Willemen uses binary variables, forcing all students with the same subject selection into the same classes. As we show, our model imposes no such restriction; nor is any required.

We believe our paper also makes a secondary, but nevertheless useful, contribution to the more general area of "aggregated", or "implicit", models. Our general integer model does not explicitly represent the assignment of individual students to classes, but rather only represents the allocation of groups of students (those with identical subject selections) into groups of classes (blocks). Both students and class variables have been aggregated, and their representation in terms of individual students and classes is implicit in the model. Whilst the proof of this is somewhat long and technical, we believe it worth including, as we believe this general idea to have much wider application. Since carrying out the research we report in this paper, some of the authors have found the same idea can be applied to improve models for problems as diverse as cutting stock, cancer radiation treatment planning and labour force modelling. We thus believe that rigourously establishing the underlying theory is worthwhile.

[4] 'We considered general integer programming, but abandoned the attempt because of the combinatoric nature of the problem' [15].

3 Problem Definition and Integer Linear Programming Approaches

We begin by introducing the notation and terminology we will use, first formally defining the PCTP, and then discussing precisely the form of the CBPP and how it arises. We go on to give two variants on an integer linear programming model of the feasibility version of the CBPP. We then discuss three alternative approaches to solve this via optimization, with penalties on infeasibility.

3.1 Terminology, Notation and Problem Definition

There is considerable variation in terminology for academic timetabling, and some words used with technical meanings also have common meanings that may mislead the reader. It is therefore necessary to explain the concepts and terminology for course timetabling used throughout this paper. We do so with a particular focus on the CBPP. However the terminology is equally suitable for other course timetabling problems. The data set under investigation in this paper is a school data set, and consequently the language will assume that the problem is being defined for a school, even though the terminology may well be appropriate for a university or other institution.

Some notation is introduced only for the purpose of initial exposition of the problem; any notation used in subsequent sections is gathered together, including definitions of variables and parameters of the integer programming models, in Table 1.

The first definitions required make no specific mention of individual teachers or students, and represent some of the data supplied at the start of the timetabling process.

A school offers to its students a variety of *subjects*, represented by an indexed set $\mathbf{S} = \{1, 2, \ldots, S\}$. A subject $s \in \mathbf{S}$ is defined primarily by its content (for example, Geography) and its year level (so that Year 11 Geography and Year 12 Geography are treated as distinct subjects). Each distinct subject s is taught in one or more *classes*: a subject with sufficiently few students requires only one class, while one taken by all students in a given year level will require several classes. We assume that the number of classes of each subject is determined *a*

priori. The following parameters are defined

γ_s = the capacity (maximum students) for any one class of subject s;

μ_s = the number of classes of subject s.

The ith class of subject s is denoted by the ordered pair (s, i) and the set of classes is

$$\mathbf{C} = \{(s, i) \,:\, s \in \mathbf{S},\ 1 \leq i \leq \mu_s\}.$$

School timetables are based on *cycles* (weekly or fortnightly cycles are the most common) and setting the timetable for one cycle determines it for a longer period (a term, semester or year, depending on the practices of the school). The teacher and students assigned to a given class meet a number of times each timetable cycle for activities that will be called *lessons* (discussed more fully below). The parameter λ_s, defined to be the number of lessons for any one class of subject s per timetable cycle, is assumed to be given. Each lesson occupies one of a set of possible time intervals, called *sessions*, for example 8.50 a.m. to 9.30 a.m. on Tuesday of the first week in a fortnightly timetable cycle. The indexed set $\mathbf{A} = \{1, 2, \ldots, A\}$ is the set of sessions available per timetable cycle. Each lesson also has a location, referred to as a *room*, which may be a generic classroom, or a specialized location such as a laboratory. In this paper, we assume rooms are not a scarce resource, and ignore the room allocation aspect of timetabling.

Thus we define a *timetable* to be an assignment of each lesson to a session of the timetable, for all lessons of all classes of all subjects.

The students and teachers at the school are represented by the indexed sets $\mathbf{N} = \{1, 2, \ldots, N\}$ and $\mathbf{T} = \{1, 2, \ldots, T\}$, respectively. The set of all students taking subject $s \in \mathbf{S}$ is denoted by $\mathbf{N}_s$, while the set of all teachers who teach subject $s \in \mathbf{S}$ is denoted by $\mathbf{T}_s$.

Some students will have made identical subject selections, but there is typically considerable diversity. Each student at a school, $n \in \mathbf{N}$, will be taking a selection of subjects, which we will call a *program*.

It is useful to define an indexed set $\mathbf{P} = \{1, 2, \ldots, P\}$ of *(distinct) student programs*: each element of this set is a set of subjects and there is at least one student who takes precisely these subjects and no others. It is also helpful to

define

$$\begin{aligned} \nu_p &= \text{the number of students taking program } p, \\ \boldsymbol{\omega}_n &= \text{the program that student } n \text{ takes}, \\ \boldsymbol{\pi}_p &= \text{the program represented by } p. \end{aligned}$$

As teachers may take multiple classes of the same subject, the parameter

$$\theta_{ts} = \text{the number of classes of subject } s \text{ taken by teacher } t$$

is defined. From θ_{ts} for each $s \in \mathbf{S}$ we deduce

$$\boldsymbol{\tau}_t = \text{the set of subjects taught by teacher } t.$$

Note that it is not $\boldsymbol{\tau}_t$ alone, but rather the set $\boldsymbol{\tau}_t$ and the parameters θ_{ts} for all subjects $s \in \boldsymbol{\tau}_t$, that captures the teacher's assigned duties. We assume that this is defined *a priori* (such was the case in the high school we worked with). However, it is not difficult to extend the models in this paper to relax this requirement, for example, assuming that only $\boldsymbol{\tau}_t$ is given for each teacher t. Indeed, the models we give take the values of θ_{ts} to be upper bounds on the number of classes of each subject a teacher can teach.

In terms of the sets and parameters introduced, it is now possible to to address the issue of the membership of class (s, i), the ith class of subject s. When the timetable has been fully constructed, the class (s, i) will be associated with

$$\begin{aligned} \mathbf{Q}_s^i &= \text{the set of students in class } i \text{ of subject } s; \\ t_s^i &= \text{the teacher assigned to teach class } i \text{ of subject } s. \end{aligned}$$

When these quantities are known, one may say that the classes have been populated with teachers and students. For example, consider year 12 Geography (subject 43, say). The second year 12 Geography class may be defined as being taught by teacher Burstall (teacher 6) to students Blake, Blundell, Forster, Hagen and Miller (students 4, 27, 40, 46 and 87), or $t_{43}^2 = 6$ and $\mathbf{Q}_{43}^2 = \{4, 27, 40, 46, 87\}$. For this definition to be correct year 12 Geography must be an element of the programs of both the teacher and all students ($43 \in \boldsymbol{\tau}_t$ for $t = 6$ and $43 \in \boldsymbol{\omega}_n$ for all $n \in \{4, 27, 40, 46, 87\}$). In other words, the sets $\{\mathbf{Q}_s^i\}_{i=1,\ldots,\mu_s}$ partition the set $\mathbf{N}_s$.

Each class will meet in multiple sessions of a timetable. A *lesson* will be defined as each instance of class i of subject s meeting in the timetable, and can be thought of as an *event*, with the set of all events given by $\mathbf{V} = \{(s, i, j) : s \in \mathbf{S}, i = 1, ..., \mu_s, j = 1, ..., \lambda_s\}$. (Recall that we assume that the school has already determined how many lessons for each class of each subject there will be in the timetable, denoted by λ_s.) In the creation of a timetable, each lesson (event) $(s, i, j) \in \mathbf{V}$ must be timetabled. $l_s^{ij} \in \mathbf{A}$ is used to denote the session assigned to event $(s, i, j) \in \mathbf{V}$. From this we can derive the set of lessons, $\mathbf{L}_s^i$, for each class $i = 1, \ldots, \mu_s$ of each subject $s \in \mathbf{S}$: $\mathbf{L}_s^i = \{l_s^{ij} \ : \ j = 1, \ldots, \lambda_s\}$. For example the lessons of the second class of year 12 Geography (subject 43) will be held in sessions 6, 12, 15, 16, 26 and 33 or $\mathbf{L}_{43}^2 = \{6, 12, 15, 16, 26, 33\}$.

In order to produce a populated timetable for a PCTP it must be determined which students and which teacher will comprise each class of each subject (the classes need to be *populated*), and it needs to be determined in which sessions each lesson of each class of each subject will be held (the classes need to be *timetabled*). Consequently a successful outcome of the timetabling process will be:

- a set of class populations $\{t_s^i, \mathbf{Q}_s^i\}$ for each class $i \in 1, ..., \mu_s$ of each subject $s \in \mathbf{S}$; and
- the sessions $l_s^{ij} \in \mathbf{A}$ in which the lessons of each class will be held.

Having defined the inputs and outputs of a PCTP, we now discuss the blocking simplification, and describe the consequent decomposition into two problems: the block constitution and population problem, and the block timetabling problem. Note that it is the former of these two (the CBPP) which is the focus of this paper.

Blocking is the assumption that the classes can be partitioned into sets, called *blocks*, that will be timetabled together, with each session in the timetable devoted to classes in a single block, i.e. no classes in different blocks will have a lesson timetabled at the same time. This leads to a problem with two independent parts: the constitution of the blocks and population of classes has no bearing on the actual timetabling of the lessons; the two can be treated as independent problems. Thus blocking is essentially a heuristic idea, which restricts the solution space of the PCTP so that it can be decomposed. The two subproblems in the decomposition – the class blocking and population problem

(**CBPP**) and the block timetabling problem (**BTP**) – will be defined as follows.

The set of all blocks is defined as the set $\mathbf{B} = \{1, ..., B\}$. Each block $b \in \mathbf{B}$ is used to represent a set of classes. It is generally assumed when using this decomposition that all subjects $s \in \mathbf{S}$ require the same number of lessons which is defined to be $\lambda = \lambda_s$ for all $s \in \mathbf{S}$. Each block $b \in \mathbf{B}$ has associated with it a set of classes $\mathbf{C}_b \subseteq \mathbf{C}$. In other words, $\{\mathbf{C}_b\}_{b \in \mathbf{B}}$ partitions the set of all classes $\mathbf{C}$. All lessons of all classes allocated to the block will be allocated to an identical set of sessions in the timetable ($L^i_{s_1} = L^j_{s_2}$, for all $(s_1, i), (s_2, j) \in \mathbf{C}_b$, for all $b \in \mathbf{B}$). Thus no student or teacher may be in the population of two classes that are allocated to the same block. Subjects with more than one class may have multiple classes in each block.

To summarize, the input for the CBPP is the set $\mathbf{S}$ of subjects and the programs for all students ($\boldsymbol{\omega}_n$ for all $n \in \mathbf{N}$) and teachers ($\boldsymbol{\tau}_t$ for all $t \in \mathbf{T}$), together with all of the parameters associated with these sets ($\mu_s, \gamma_s, \lambda_s$ and θ_{ts}). The population of each class must be determined and each class of each subject must be allocated to exactly one block. The output will be a set of class populations $(t^i_s, \mathbf{Q}^i_s)$ for all $i \in 1, ..., \mu_s$, for all $s \in \mathbf{S}$ and the set of classes allocated to each block ($\mathbf{C}_b$ for all $b \in \mathbf{B}$). The CBPP can be seen as a constraint satisfaction problem where one is trying to satisfy the following constraints:

- **Student Allocation (SA):** All students must be allocated to a class for each subject in their program;

$$|\{i : (s, i) \in \mathbf{C}, n \in \mathbf{Q}^i_s\}| = 1, \quad \forall s \in \boldsymbol{\omega}_n, \forall n \in \mathbf{N}. \tag{1}$$

- **Student Capacity (CS):** The total number of students allocated to a class must not exceed the capacity of the class;

$$|\mathbf{Q}^i_s| \leq \gamma_s, \quad \forall (s, i) \in \mathbf{C}. \tag{2}$$

- **Teacher Allocation (TA):** All teachers must be allocated to the required number of classes for each subject in their program;

$$|\{i : (s, i) \in \mathbf{C}, t = t^i_s\}| = \theta_{ts}, \quad \forall s \in \boldsymbol{\tau}_t, \forall t \in \mathbf{T}. \tag{3}$$

- **Teacher Capacity (CT):** There must be a teacher allocated to teach each class of every subject. This constraint is enforced implicitly by the definition of the t^i_s variables;

$$t^i_s \in \mathbf{T}, \quad \forall (s, i) \in \mathbf{C}. \tag{4}$$

- **Class Block Allocation (CB):** every class of every subject must be allocated to a block;

$$|\{b : (s,i) \in \mathbf{C}_b\}| = 1, \quad \forall (s,i) \in \mathbf{C}. \tag{5}$$

  Note that this is equivalent to asking that the sets $\{\mathbf{C}_b\}_{b \in \mathbf{B}}$ partition the set $\mathbf{C}$.

- **Student Block Clash (SB):** only one class requiring the same student may be allocated to a block;

$$|\{(s,i) \in \mathbf{C}_b : n \in \mathbf{Q}_s^i\}| \leq 1, \quad \forall b \in \mathbf{B}, \forall n \in \mathbf{N}. \tag{6}$$

- **Teacher Block Clash (TB):** only one class requiring the same teacher may be allocated to a block;

$$|\{(s,i) \in \mathbf{C}_b : t_s^i = t\}| \leq 1, \quad \forall b \in \mathbf{B}, \forall t \in \mathbf{T}. \tag{7}$$

The above defines a feasible solution to the CBPP. As mentioned in Section 2, the CBPP can be further, heuristically, decomposed into two problems: the student population problem, and the class blocking problem. The former seeks sets $(t_s^i, \mathbf{Q}_s^i)$ while the block sets $\mathbf{C}_b$ remain fixed. The latter involves finding block sets $\mathbf{C}_b$ while the population sets $(t_s^i, \mathbf{Q}_s^i)$ remain fixed. Papers taking this approach were discussed in Section 2.

Once the CBPP has been solved, the blocks themselves need to be timetabled, that is, the *block timetabling problem* (BTP) needs to be solved, to create the sets $\mathbf{L}_b \subseteq \mathbf{A}$.

Again it should be noted that it is assumed for this model that all classes (and therefore blocks) require the same number of lessons λ. Consequently the constraints defining the BTP are:

- **Block Allocation (BA):** every class in a block must have lessons in the same set of sessions, and there must be the appropriate number of sessions;

$$|\{\mathbf{L}_b\}| = \lambda, \quad \forall b \in \mathbf{B}. \tag{8}$$

- **Block Clashing (BC):** no more than one block may be allocated to a session;

$$|\{b \in \mathbf{B} : a \in \mathbf{L}_b\}| \leq 1, \quad \forall a \in \mathbf{A}. \tag{9}$$

For practical application, additional constraints determined by the institution may be anticipated. For example a school may forbid a class to have more than one lesson in a day, and the available sessions for some teachers may be restricted. A solution to this problem will be the partitioning of the sessions into appropriate size sets (λ sessions in each set) for each block, and depending on the school's restrictions this can be either relatively trivial (if there are no restrictions, it can be modelled as a transportation problem), or more difficult.

The major strength of the blocking decomposition is that the timetable has a strong underlying structure and so minor changes to the finished timetable are very easy to make. The major weakness is that it restricts the solution space of the PCTP, which may eliminate many, or possibly all, feasible solutions for instances that are tightly constrained (such as when students or teachers have strong restrictions on the sessions for which they are available). As we will show, another strength of the blocking model is that it can be solved easily with standard integer linear programming packages.

In what follows, we present our approach to modelling and solving the CBPP. In doing so, we make several assumptions, for example, that any class can be assigned to any block, which is reasonable if subjects require the same number of lessons, but which may present problems otherwise. We call the result "pure blocking", and have found this representative of the core practices at the local high school, in particular in the upper year levels. However the *complete* high school timetabling problem, in practice, involves "partial blocking", subjects with different numbers of lessons, students in the same year level taking different numbers of subjects, etc. Whilst the blocking models we present here form the basis for a complete solution to the problem, much more needs to be done to obtain this solution. For further details, see the thesis of Merlot [12].

3.2 Two Integer Linear Programming Models for the CBPP

In this section we introduce a new integer linear programming model, with two variants, for the CBPP. In contrast to all previous work, each of the following extensions of simpler ideas is simultaneously incorporated in the new model.

(a) Class allocation variables are not binary variables, but are general integer variables, allowing more than one class of a subject to be allocated to the same block.

(b) Teacher allocation is considered in detail with no a *priori* restrictions.

(c) In the second variant, variables combining students with the same program of subjects are used, thus eliminating symmetry in the student allocation variables, but in a way that allows students with the same program to be allocated differently when necessary.

An important practical advantage of the model we will present is that, as we will demonstrate with the data from a local high school, relaxing integrality of the student or teacher allocation variables leads to all fully integer solution of all instances without the need for branching.

The model assumes that the number of blocks to be created is known *a priori*. In high school timetabling, this is not a strong requirement: generally students are occupied during every session of the timetable, so the number of blocks can be taken to be the number of subjects in the students' programmes. In other words, we assume the values of $|\boldsymbol{\omega}_n|$ are identical for all students $n \in \mathbf{N}$, and set B to this value. In university timetabling, it is more difficult to know B in advance; we discuss possible approaches to this in Section 6.

The model does not explicitly populate classes with students. Instead, it allocates classes of subjects to blocks, and allocates students and teachers to take specific subjects in specific blocks. In other words, *subjects* in blocks are populated, but specific classes are not populated explicitly. This is an important distinction, as it allows the model to be linear, in contrast to nonlinear models, such as that of Wood and Whitaker [19], without using a large number of additional variables or standard linearization techniques, which often lead to weak models. The key to this approach is that the solution can a posteriori be shown to induce feasible class populations. In other words, the detail of class population can safely be deduced from the subject-block population.

To constitute the blocks, we use general integer variables x_{sb} to represent the number of classes of subject s assigned to block b. Note we assume no *a priori* prescription on the blocks, assuming that any class can be assigned to any block. (If all classes require the same number of lessons, this is a reasonable assumption.) Since all classes of every subject must be allocated to a block, we require the following constraint:

$$\sum_{b \in \mathbf{B}} x_{sb} = \mu_s, \quad \forall s \in \mathbf{S}. \tag{10}$$

For teachers the variables used will be the binary variables, z_{tsb}, which are 1 if a class of subject $s \in \boldsymbol{\tau}_t$ is taken by teacher t in block b and 0 otherwise. To ensure each teacher is allocated to the set of classes they have been assigned to teach, is not required to teach two classes at the one time (no clashes), and to ensure every class receives a teacher, we require the following constraints, where $\mathbf{T}_s$ is the set of teachers that teach subject $s \in \mathbf{S}$ ($\mathbf{T}_s = \{t \in \mathbf{T} \ : \ s \in \boldsymbol{\tau}_t\}$).

- **Teacher Subject Allocation (ST):** every teacher must be allocated a block for every class of every subject in their program;

$$\sum_{b \in \mathbf{B}} z_{tsb} = \theta_{ts}, \quad \forall t \in \mathbf{T}, \quad \forall s \in \boldsymbol{\tau}_t. \tag{11}$$

- **Teacher Block Allocation (BT):** each teacher must have no more than one subject allocated to any block;

$$\sum_{s \in \boldsymbol{\tau}_t} z_{tsb} \leq 1, \quad \forall t \in \mathbf{T}, \quad \forall b \in \mathbf{B}. \tag{12}$$

- **Teacher Block Capacity (TBC):** every class of every subject must have a teacher;

$$\sum_{t \in \mathbf{T}_s} z_{tsb} = x_{sb}, \quad \forall s \in \mathbf{S}, \quad \forall b \in \mathbf{B}. \tag{13}$$

In the first variant of our pure blocking model, which we call the *separate student* model (since students are modelled separately, rather than in groups), binary variable v_{nsb} is equal to 1 if student n takes subject s in block b and 0 otherwise. To ensure every student is assigned to block for every subject in their program, and a subject in every block, and that the class capacities are respected, we require the following constraints.

- **Program Subject Allocation (SP):** every student must be allocated a block for every subject in their program;

$$\sum_{b \in \mathbf{B}} v_{nsb} = 1, \quad \forall s \in \boldsymbol{\omega}_n, \quad \forall n \in \mathbf{N}. \tag{14}$$

- **Program Block Allocation (BP):** every student taking a program must have a subject allocated to every block (the student must always be occupied);

$$\sum_{s \in \boldsymbol{\omega}_n} v_{nsb} = 1, \quad \forall n \in \mathbf{N}, \quad \forall b \in \mathbf{B}. \tag{15}$$

- **Block Subject Capacity (BSC):** the total number of students taking a subject in a block must be no more than the total capacity of the classes of that subject in the block;

$$\sum_{n \in \mathbf{N}_s} v_{nsb} \leq \gamma_s x_{sb}, \quad \forall s \in \mathbf{S}, \quad \forall b \in \mathbf{B}. \tag{16}$$

It is not hard to see the latter constraint ensures that if multiple classes of the same subject $s$are assigned to a block b, the student population of the subject in the block induced by the v_{nsb}variables for each $n \in \mathbf{N}_s$ can be (arbitrarily) partitioned into x_{sb}classes, each having capacity no more than γ_s.

The separate student variant of our pure blocking model is thus given by the variables x_{sb}, z_{tsb} and v_{nsb}, together with the constraints (10), (11), (12), (13), (14), (15) and (16). With the appropriate re-interpretation, this model is quite similar to that of Sampson et al. [15], except that their x_{sb} variables were taken to be binary, and of course, their model was never implemented or tested.

A potential problem with this model is that many students may have an identical programme of study, that is, an identical subject selection. The v_{nsb} variables for such students will play an identical role in the model, which will then suffer from symmetry.

We have observed that it is not in fact necessary to model individual students explicitly. Students having the same programme can be grouped together; the activity of these students can be represented using (general integer) variables that depend only on the program taken, not on the individual student. To do this, we define the set of distinct programs taken by students by $\mathbf{P} = \{1, ..., P\}$, with each $p \in \mathbf{P}$ representing program $\boldsymbol{\pi}_p$. The sets $\mathbf{N}_p = \{n \in \mathbf{N} \ : \ \boldsymbol{\omega}_n = \boldsymbol{\pi}_p\}$ partition the set of all students $\mathbf{N}$ into sets of students all taking an identical program $\boldsymbol{\pi}_p$. Note that

$$\{\boldsymbol{\pi}_p \ : \ p \in \mathbf{P}\} = \{\boldsymbol{\omega}_n \ : \ n \in \mathbf{N}\},$$

and define

$$\nu_p = |\{n \in \mathbf{N} \ : \ \boldsymbol{\omega}_n = \boldsymbol{\pi}_p\}|$$

to be the number of students taking program p. The set $\mathbf{P}_s = \{p \in \mathbf{P} \ : \ s \in \boldsymbol{\pi}_p\}$ is defined to be the set of student programs that include subject $s \in \mathbf{S}$.

Now integer variables y_{psb} are used to represent the number of students taking program p that are assigned to a class of subject s in block b, for all

programs $p \in \mathbf{P}$, for each subject $s \in \boldsymbol{\pi}_p$ and each block $b \in \mathbf{B}$. Use of these variables has clear advantages over the use of the binary variables for every student: the number of variables is reduced, but also the symmetry in the model is greatly reduced, as the binary variables for students taking identical programs play an identical role in the model. Symmetry of this type can significantly degrade the performance of the branch-and-bound method for solving the integer linear program, so reducing the symmetry in this way is a significant advantage.

We require the y_{psb} variables to satisfy the following constraints, which can be derived by substituting

$$y_{psb} = \sum_{n \in \mathbf{N}_p} v_{nsb}, \quad \forall p \in \mathbf{P}, \quad \forall s \in \boldsymbol{\pi}_p, \quad \forall b \in \mathbf{B} \tag{17}$$

in constraints (14) and (15), summed over all $n \in \mathbf{N}_p$, and in constraint (16), where it can be observed that the set $\mathbf{N}_s$ is partitioned by the sets $\mathbf{N}_p$ with $s \in \boldsymbol{\pi}_p$.

- **Program Subject Allocation (SP):** all students taking a particular program must take each subject of their program in some block;

$$\sum_{b \in \mathbf{B}} y_{psb} = \nu_p, \quad \forall p \in \mathbf{P}, \quad \forall s \in \boldsymbol{\pi}_p. \tag{18}$$

- **Program Block Allocation (BP):** every student taking must take some subject of their program in a given block;

$$\sum_{s \in \boldsymbol{\pi}_p} y_{psb} = \nu_p, \quad \forall p \in \mathbf{P}, \quad \forall b \in \mathbf{B}. \tag{19}$$

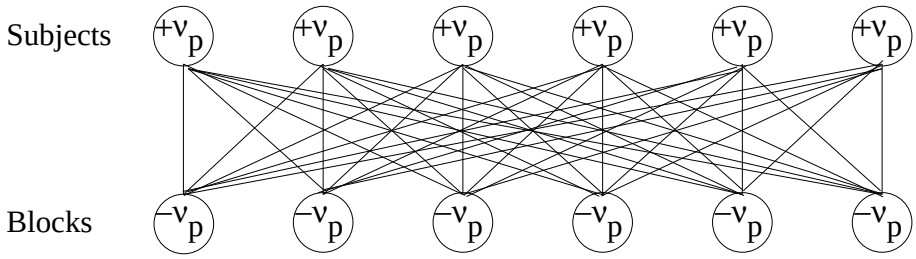
- **Block Subject Capacity (BSC):** the total number of students taking a subject in a block must be no more than the total capacity of the classes of that subject in the block;

$$\sum_{p \in \mathbf{P}_s} y_{psb} \leq \gamma_s x_{sb}, \quad \forall s \in \mathbf{S}, \quad \forall b \in \mathbf{B}. \tag{20}$$

The *combined* student variant of our pure blocking model is thus given by the variables x_{sb}, z_{tsb} and y_{psb}, together with the constraints (10), (11), (12), (13), (18), (19) and (20).

It is far from obvious that the solution resulting from the use of these y_{psb} variables can always be decomposed into a feasible assignment of individual students to classes in blocks. It is shown in Section 4 that this is indeed possible.

Figure 1: **The Network Structure of the Constraints.** This diagram shows the balanced transportation problem induced by the constraints (18) and (19) on the y_{psb} variables, for each program p. In this example, there are six blocks (and six subjects in every program). There is a supply node for each subject in the program with supply ν_p, and a demand node for each block, with demand ν_p.



Before proceeding to discuss optimization models for solving the pure blocking feasibility problem, we note that this model has some nice underlying structure. For each program p, constraints (18) and (19) define a special case of a balanced transportation problem. The corresponding network structure is illustrated in Figure 1, with a supply node for each subject in the program, and a demand node for each block. (Recall the standing assumption that there are an identical number of subjects in every program, and that this number is precisely the number of blocks.) The transportation problems for each program are tied together via the constraint (20), which can be seen to be a kind of variable upper bound flow constraint. A similar network structure underlies the constraints (11) and (12) on the z_{tsb} variables.

The strong underlying network structure of these variables has some benefits. When the allocation of classes to blocks is fixed, (the x_{sb} variables are determined), the problem is still a non-standard network flow problem as the upper bounds on the flows are summed over many arcs. However the problem solves very quickly, and still has the benefits of the underlying network structure. As we demonstrate in our numerical results, in all of our instances, (under all model variants), once the x_{sb} variables are determined, no further branching is required.

So far, our CBPP has been formulated as constraint satisfaction problem. However, in practice, this problem may not be feasible. In the next section, we discuss three alternative optimization models, in which some aspect of the problem is relaxed, and penalties imposed on infeasibility.

3.3 Optimization Models for Minimizing Infeasibility

Infeasibility in a CBPP model can arise from several "causes". Perhaps more classes of a particular subject are needed, or perhaps the teacher assignment's need to be altered. Perhaps some students should change their subject allocation.

We consider three alternative ways of relaxing the constraints, and minimizing infeasibility (or maximizing feasibility). We discuss these in terms of the combined student variant, but the same alternatives exist (and are implemented and tested) for the separate student variant, in the obvious way. In all cases, we relax the ST (11) constraints, and the number of teacher assignments to classes is maximized.

In the first model, the SP (18) and BP (19) constraints are relaxed, and the number of student assignments to classes is maximized (see model (PBILP1) below). In the case that the problem is not in fact feasible, there is a risk with this approach that the model may assign many students to classes for *parts* of their programs rather than assigning *complete* programs for fewer students.

In the second, we encourage the model to ensure as many students as possible are assigned blocks for all subjects in their programs. To do this, we introduce new variables, w_p, that represent the number of students taking program p that have had all of their subjects completely assigned. The model seeks to maximize the total number of student programs that have been completely assigned (see model (PBILP2) below).

A third alternative (model (PBILP3) below) is allow class capacities to be exceeded. Consequently the variables u_{sb} are introduced to model the extra capacity required for each subject and block combination.

It is not clear which of these alternatives might be most useful for the schools in the case of infeasibility, as all the real data sets we obtained from the local high school were in fact feasible. In practice, all three may provide useful information to the timetabler, when infeasibility occurs.

The first Pure Blocking Integer Linear Program (PBILP1) is given below,

where α and β are the respective "rewards" for assigning a student to a class in a block for one of the subjects in his/her program, and for giving a class a teacher:

$$\mathbf{max}\{\sum_{p\in\mathbf{P}}\sum_{s\in\boldsymbol{\pi}_p}\sum_{b\in\mathbf{B}}\alpha y_{psb}+\sum_{t\in\mathbf{T}}\sum_{s\in\boldsymbol{\tau}_t}\sum_{b\in\mathbf{B}}\beta z_{tsb}\} \tag{21}$$

subject to constraints (10), (12), (20), and

$$\sum_{b\in\mathbf{B}} y_{psb} \leq \nu_p, \quad \forall p\in\mathbf{P}, \quad \forall s\in\boldsymbol{\pi}_p; \tag{22}$$

$$\sum_{s\in\boldsymbol{\pi}_p} y_{psb} \leq \nu_p, \quad \forall p\in\mathbf{P}, \quad \forall b\in\mathbf{B}; \tag{23}$$

$$\sum_{b\in\mathbf{B}} z_{tsb} \leq \theta_{ts}, \quad \forall t\in\mathbf{T}, \quad \forall s\in\boldsymbol{\tau}_t; \tag{24}$$

$$\sum_{t\in\mathbf{T}_s} z_{tsb} \leq x_{sb}, \quad \forall s\in\mathbf{S}, \quad \forall b\in\mathbf{B}; \tag{25}$$

$$x_{sb} \in \mathbb{Z}_+, \quad \forall s\in\mathbf{S}, \quad \forall b\in\mathbf{B}; \tag{26}$$

$$y_{psb} \in \mathbb{Z}_+, \quad \forall p\in\mathbf{P}, \quad \forall s\in\boldsymbol{\pi}_p, \quad \forall b\in\mathbf{B}; \tag{27}$$

$$z_{tsb} \in \{0,1\}, \quad \forall t\in\mathbf{T}, \quad \forall s\in\boldsymbol{\tau}_t, \quad \forall b\in\mathbf{B}. \tag{28}$$

The second Pure Blocking Integer Linear Program (PBILP2) is given below, where ω is the "reward" for assigning a student's complete program of subjects to blocks, and β is as defined earlier:

$$\mathbf{max}\left\{\sum_{p\in\mathbf{P}}\omega w_p+\sum_{t\in\mathbf{T}}\sum_{s\in\boldsymbol{\tau}_t}\sum_{b\in\mathbf{B}}\beta z_{tsb}\right\} \tag{29}$$

subject to constraints (10), (12), (20), (24), (25), (26), (27), (28) and

$$\sum_{b\in\mathbf{B}} y_{psb} = w_p, \quad \forall p\in\mathbf{P}, \quad \forall s\in\boldsymbol{\pi}_p; \tag{30}$$

$$\sum_{s\in\boldsymbol{\pi}_p} y_{psb} = w_p, \quad \forall p\in\mathbf{P}, \quad \forall b\in\mathbf{B}; \tag{31}$$

$$w_p \leq \nu_p, \quad \forall p\in\mathbf{P}; \tag{32}$$

$$w_p \in \mathbb{Z}_+, \quad \forall p\in\mathbf{P}. \tag{33}$$

The third Pure Blocking Integer Linear Program (PBILP3) is given below, where γ is the "penalty" per student per class for violating the class capacity constraint:

$$\mathbf{max}\left\{\sum_{t\in\mathbf{T}}\sum_{s\in\boldsymbol{\tau}_t}\sum_{b\in\mathbf{B}}\beta z_{tsb}-\sum_{s\in\mathbf{S}}\sum_{b\in\mathbf{B}}\gamma u_{sb}\right\} \tag{34}$$

subject to constraints (10), (12), (18), (19), (24), (25), (26), (27), (28), and

$$\sum_{p \in \mathbf{P}_s} y_{psb} \leq \gamma_s x_{sb} + u_{sb}, \quad \forall s \in \mathbf{S}, \quad \forall b \in \mathbf{B}; \tag{35}$$

$$u_{sb} \in \mathbb{Z}_+, \quad \forall s \in \mathbf{S}, \quad \forall b \in \mathbf{B}. \tag{36}$$

At this point, we have defined all notation and variables required to formulate our models; we summarize these in Table 1. Choices of α, β, γ and ω used for the data sets tested are discussed in Section 5.

We now proceed to prove that the combined student variant "makes sense" for all three of the PBILP models.

Table 1: **Table of Symbols**

Symbol	Description
$\mathbf{S} = \{1, ..., S\}$	set of subjects (with elements s)
$\mathbf{B} = \{1, ..., B\}$	set of blocks (with elements b)
$\mu_s \in \mathbb{Z}_+$	number of classes there will be of subject s
$\gamma_s \in \mathbb{Z}_+$	maximum capacity of a class of subject s
$\mathbf{C} = \{(s, i) : s \in \mathbf{S},\ 1 \leq i \leq \mu_s\}$	set of all classes
$\mathbf{N} = \{1, 2, \ldots, N\}$	set of all students
$\boldsymbol{\omega}_n \subset \mathbf{S}$	set of subjects in programme of student $n \in \mathbf{N}$
$\mathbf{N}_s = \{n \in \mathbf{N} \ : \ s \in \boldsymbol{\omega}_n\}$	set of students taking subject s
$\mathbf{P} = \{1, ..., P\}$	set of discrete programs (with elements p)
$\boldsymbol{\pi}_p \subset \mathbf{S}$	program of subjects represented by p
$\nu_p \in \mathbb{Z}_+$	number of students taking program p
$\mathbf{N}_p = \{n \in \mathbf{N} \ : \ \boldsymbol{\omega}_n = \boldsymbol{\pi}_p\}$	set of all students taking program $\boldsymbol{\pi}_p$
$\mathbf{P}_s = \{p \in \mathbf{P} \ : \ s \in \boldsymbol{\pi}_p\}$	set of student programs that include subject $s \in \mathbf{S}$
$\mathbf{T} = \{1, ..., T\}$	set of teachers (with elements t)
$\boldsymbol{\tau}_t \subset \mathbf{S}$	program taught by a teacher (set of subjects)
$\theta_{ts} \in \mathbb{Z}_+$	number of classes of subject s taught by teacher t
$\mathbf{T}_s = \{t \in \mathbf{T} \ : \ s \in \boldsymbol{\tau}_t\}$	set of teachers that teach subject $s \in \mathbf{S}$
x_{sb}	number of classes of subject s allocated to block b
z_{tsb}	indicates teacher t teaches subject s in block b
v_{nsb}	indicates that student $n$takes subject s in block b
y_{psb}	number of students with program p taking subject s in block b
w_p	number of students taking program p that have every subject in their program allocated to a block
u_{sb}	number of extra students beyond capacity taking subject s in block b
α	"reward" for assigning a student to a class in a block for one of the subjects in his/her program
β	"reward" for giving a class a teacher
ω	"reward" for assigning a student's complete program of subjects to blocks
γ	"penalty" per student per class for violating the class capacity constraint

4 Validity of Combined Student Variables in PBILP

For the PBILP models to be able to use general integer variables for students with the same program (y_{psb}), we must prove that the allocation of subjects to blocks for all students taking this program can be decomposed into allocations of subjects to blocks for individual students, (v_{nsb} variables), and that doing so does not "disturb" the objective function. In particular, it will be proved that the PBILP models are valid, in a sense defined as follows.

For all three combined student PBILP models, it is claimed that if y_{psb} for each $p \in \mathbf{P}$, $s \in \mathbf{S}$ and $b \in \mathbf{B}$ is the student allocation part of any feasible solution to the model, then there exist values for v_{nsb} for every $n \in \mathbf{N}$, each subject $s \in \boldsymbol{\omega}_n$ and each block $b \in \mathbf{B}$ such that

$$\sum_{n \in \mathbf{N}_p} v_{nsb} = y_{psb}, \quad \forall p \in \mathbf{P}, \quad \forall s \in \boldsymbol{\pi}_p, \quad \forall b \in \mathbf{B}, \tag{37}$$

$$\sum_{s \in \boldsymbol{\omega}_n} v_{nsb} \leq 1, \quad \forall n \in \mathbf{N}, \quad \forall b \in \mathbf{B}, \quad \text{and} \tag{38}$$

$$\sum_{b \in \mathbf{B}} v_{nsb} \leq 1, \quad \forall n \in \mathbf{N}, \quad \forall s \in \boldsymbol{\omega}_n. \tag{39}$$

Thus for all models, it is assured that student clashing is avoided (no student takes more than one subject in any block) and every student takes each subject in his/her program no more than once. As is now shown, the total number of student subject allocations to blocks is identical to that recorded by the program variables. Using (37, interchanging the order of summation and noting that $\boldsymbol{\omega}_n = \boldsymbol{\pi}_p \forall n \in \mathbf{N}_p$, we have

$$\sum_{p \in \mathbf{P}} \sum_{s \in \boldsymbol{\pi}_p} \sum_{b \in \mathbf{B}} y_{psb} = \sum_{p \in \mathbf{P}} \sum_{s \in \boldsymbol{\pi}_p} \sum_{b \in \mathbf{B}} \Big(\sum_{n \in \mathbf{N}_p} v_{nsb} \Big) = \sum_{p \in \mathbf{P}} \sum_{n \in \mathbf{N}_p} \sum_{s \in \boldsymbol{\omega}_n} \sum_{b \in \mathbf{B}} v_{nsb}. \tag{40}$$

Since, as noted earlier, the sets $\mathbf{N}_p$ for $p \in \mathbf{P}$ partition the set of all students $\mathbf{N}$, we conclude that

$$\sum_{p \in \mathbf{P}} \sum_{s \in \boldsymbol{\pi}_p} \sum_{b \in \mathbf{B}} y_{psb} = \sum_{n \in \mathbf{N}} \sum_{s \in \boldsymbol{\omega}_n} \sum_{b \in \mathbf{B}} v_{nsb}. \tag{41}$$

This shows that in PBILP1, maximizing the sum of the program allocation variables is equivalent to maximizing the number of individual student subject to block allocations that can be made.

Furthermore, if y_{psb} for each $p \in \mathbf{P}$, $s \in \mathbf{S}$ and $b \in \mathbf{B}$, and w_p for each $p \in \mathbf{P}$, provide the student allocation part of any feasible solution to PBILP2, it will

be shown that for each program $p \in \mathbf{P}$, values can be found for the student variables v_{nsb} so that exactly w_p individual students can be assigned to take all their subjects in some block, i.e., there exists a subset of students taking program p, $\mathbf{W}_p \subseteq \mathbf{N}_p$, with $|\mathbf{W}_p| = w_p$, such that for every student $n \in \mathbf{W}_p$ in this set,

$$\begin{aligned} \sum_{s \in \boldsymbol{\omega}_n} v_{nsb} &= 1, \quad \forall b \in \mathbf{B}, \quad \text{and} \\ \sum_{b \in \mathbf{B}} v_{nsb} &= 1, \quad \forall s \in \boldsymbol{\omega}_n. \end{aligned}$$

This shows that in PBILP2, maximizing the sum of the w_p variables is equivalent to maximizing the number of individual students that can have their programs completely allocated to blocks. The same can be said for PBILP3, except that in this case, w_p is replaced by ν_p, so this shows that every individual student can be given a complete allocation of their subjects. That the excess capacity in each subject and block is identical is obvious from the discussion around constraint (17).

The proof of these claims involves first showing that the feasible solution of any balanced transportation problem with equal supplies and demands at every node can be decomposed into perfect matchings on the underlying bipartite graph, where the number of perfect matchings in the decomposition is exactly the supply (demand). Now w_p and y_{psb} variables (ν_p parameters) that provide the student allocation part of any feasible solution to PBILP2 (PBILP3) have the property that for each program, $p \in \mathbf{P}$, the y_{psb} variables induce a feasible solution for a balanced transportation problem with supply and demand at every node given by w_p (ν_p). Thus the v_{nsb} variables are set to values that indicate the perfect matchings in the decomposition, for the first w_p (ν_p) students, and the rest are set to zero. This shows the second claim. To prove the first, it will be shown that dummy subject and blocks can be introduced, and the y_{psb} variables augmented correspondingly, to induce a feasible solution to a balanced transportation problem with equal supplies and demands (equal to the largest supply or demand induced by the original y_{psb} variables in the underlying network). Again the variables v_{nsb} are set to indicate the perfect matchings found in a decomposition; these must then have components corresponding to dummy subjects or blocks removed.

4.1 Proof of Transportation Problem Solution Decomposition

In this section it will be proved that any feasible solution of a balanced transportation problem with equal supplies and demands at every node can be decomposed into perfect matchings on the underlying bipartite graph, where the number of perfect matchings in the decomposition is exactly the supply (demand). The *notation introduced and used in this section is local only to this section and the next.* The transportation problem is defined by two sets: A, the set of supply nodes, and B, the set of demand nodes, with equal cardinality, $|A| = |B| = \mu$. The complete bipartite graph on A and B is denoted by $G_{(A,B)}$. The supply at every node in A and the demand at every node in B is given by $\eta \in \mathbb{Z}_+, \eta \geq 1$. In what follows, β is used to represent feasible solutions to the transportation problem, and ζ to indicate the perfect matchings in the decomposition. For brevity we write $\mathbb{Z}_+^\mu = \{1, 2, \ldots, \mu\}$.

The main result in this section is given in Theorem 2, and is proved by mathematical induction, using the following hypothesis.

Hypothesis 1 *$H(\eta)$: If $\beta \in \mathbb{Z}_+^\mu \times \mathbb{Z}_+^\mu$, satisfies*

$$\sum_{i \in A} \beta_{ij} = \eta, \quad \forall j \in B \tag{42}$$

and

$$\sum_{j \in B} \beta_{ij} = \eta, \quad \forall i \in A \tag{43}$$

then there exists a decomposition $\zeta_{ijk} \in \{0, 1\}$ for each $k \in \{1, \ldots, \eta\}$, $i \in A$, $j \in B$ satisfying

$$\sum_{i \in A} \zeta_{ijk} = 1, \quad \forall j \in B, \quad \forall k \in \{1, \ldots, \eta\}, \tag{44}$$

$$\sum_{j \in B} \zeta_{ijk} = 1, \quad \forall i \in A, \quad \forall k \in \{1, \ldots, \eta\}, \tag{45}$$

$$\sum_{k=1}^{\eta} \zeta_{ijk} = \beta_{ij}, \quad \forall i \in A, \forall j \in B. \tag{46}$$

We recall several definitions from graph theory. For any graph G the set of nodes which have an arc connected to a specific node x on a graph is called the *neighboring set* $N_G(x)$ of node x. Similarly, the set of nodes which have an arc to any node in a subset $R \subseteq G$ of nodes on a graph is called the neighboring set

$N_G(R)$ of subset R. For any graph G and any vector $\beta \geq 0$ with a component for every arc in G, the *support graph* of β in G is the subgraph of G induced by non-zero elements of β, denoted by $G(\beta)$.

In order to prove Theorem 2, it is first necessary to establish several subsidiary results, concerning the support graph of a feasible solution to the transportation problem. It is first shown that the neighbouring set of any set of supply nodes in this support graph has cardinality at least as big as as that of the set of supply nodes. This is a necessary and sufficient condition for the support graph to contain a perfect matching. Then it is demonstrated that if a vector indicating the perfect matching is "stripped off" from the transportation problem solution, a feasible solution to a similar transportation problem is produced, with supply (demand) one less. Applying induction is then relatively straightforward.

Lemma 1 *Let A and B be two sets of equal cardinality, let $\eta \in \mathbb{Z}$, with $\eta \geq 1$, and let $\beta \in \mathbb{Z}_+^{\mu} \times \mathbb{Z}_+^{\mu}$ satisfy constraints (42) and (43). Let $G_{(A,B)}(\beta)$ be the support graph of β. For all subsets $R \subseteq A$,*

$$|N_{G_{(A,B)}(\beta)}(R)| \geq |R|, \quad \forall R \subseteq A. \tag{47}$$

Proof of Lemma 1

For brevity write $N'(R) = N_{G_{(A,B)}(\beta)}(R)$ to denote the neighbouring set of R in the support graph of β.

Let R be a subset of A. By summing equation (42) over all $i \in R$ the following equation is obtained

$$\sum_{i \in R} \sum_{j \in B} \beta_{ij} = \sum_{i \in R} \eta = |R|\eta. \tag{48}$$

The neighboring set $N'(R)$ of R represents the nodes $j \in B$ where $\beta_{ij} > 0$ for all $i \in R$, and so

$$\sum_{i \in R} \sum_{j \in N'(R)} \beta_{ij} = \sum_{i \in R} \sum_{j \in B} \beta_{ij} = |R|\eta. \tag{49}$$

By equation (43), and since R is a subset of A, and all $\beta \geq 0$

$$\sum_{i \in R} \beta_{ij} \leq \sum_{i \in A} \beta_{ij} = \eta, \quad \forall j \in B. \tag{50}$$

Summing these equations over all $j \in N'(R)$ gives

$$\sum_{j \in N'(R)} \sum_{i \in R} \beta_{ij} \leq \sum_{j \in N'(R)} \eta = |N'(R)|\eta. \tag{51}$$

Combining (49) and (51) and exchanging the order of summation,

$$|R|\eta \leq |N'(R)|\eta. \tag{52}$$

As $\eta > 0$, it is deduced that

$$|R| \leq |N'(R)|, \tag{53}$$

as required and the proof is complete. □

It will now be shown that there must exist a perfect matching in the support graph of a feasible solution to the transportation problem.

Formally, a *matching* is defined as a subset of the arcs on a graph, such that no node has more than one arc incident to it in the subset. If an arc (i, j) is in a matching, node i is said to be matched to node j. For a matching M on a bipartite graph with bipartition (A, B), if for all $i \in A$ there exists an arc $(i, j) \in M$, for some $j \in B$, then the matching is said to *saturate* A. Similarly, if for all $j \in B$ there exists an arc $(i, j) \in M$, for some $i \in A$, then the matching is said to *saturate* B. If a matching saturates both A and B then the matching is said to be *perfect*.

Bondy and Murty [5] (p. 72), state and prove the following theorem of Hall [8], from which it may be deduced that the support graph of a feasible solution to the balanced transportation problem with identical supplies and demands contains a perfect matching.

Theorem 1 [Bondy and Murty [5]] *Let G be a bipartite graph with bipartition (X, Y). Then G contains a matching that saturates every vertex (node) in X if and only if:*

$$|N_G(S)| \geq |S|, \quad \forall S \subseteq X \tag{54}$$

Theorem 1 is required to prove the following result.

Lemma 2 *Given two sets A and B with equal cardinality, $|A| = |B|$, and given $\eta \in \mathbb{Z}, \eta \geq 1$, if $\beta \in \mathbb{Z}_+^{\mu} \times \mathbb{Z}_+^{\mu}$ satisfies equations (42) and (43) then there exists a perfect matching in the support graph $G_{(A,B)}(\beta)$ of β.*

Proof of Lemma 2

Firstly note that $G_{(A,B)}(\beta)$ is a bipartite graph with bipartition (A, B). Now the definitions of A, B, η and β in Lemma 2 satisfy the conditions of Lemma 1, and thus for any subset $R \subseteq A$ the neighboring set is at least as large as the

original subset. Thus, by Theorem 1, $G_{(A,B)}(\beta)$ must contain a matching that saturates every node in A. As there are μ nodes in A, there must be μ arcs in the matching, one incident to each node in A. However $|B| = \mu$ also, and no node in B can be incident to more than one arc in any matching. Thus the matching also saturates B. Hence the matching is perfect. □

Since the support graph of a feasible solution to the balanced transportation problem with identical supplies and demands contains a perfect matching, a vector indicating the perfect matching can be "stripped off", leaving a feasible solution to a balanced transportation problem having supply (demand) reduced by one. This is established formally in the following two corollaries to Lemma 2.

Corollary 1 *Under the conditions of Lemma 2, if there exists* $\beta \in \mathbb{Z}_+^{|A|} \times \mathbb{Z}_+^{|B|}$ *satisfying equations (42) and (43), then there must exist a binary vector* $\xi \in \{0,1\}^{|A|} \times \{0,1\}^{|B|}$ *with the following properties:*

(i) $\xi_{ij} > 0$ *only if* $\beta_{ij} > 0$,

(ii) $\sum_{i \in A} \xi_{ij} = 1$ *for all* $j \in B$, *and*

(iii) $\sum_{j \in B} \xi_{ij} = 1$ *for all* $i \in A$.

Proof of Corollary 1

Let M be the perfect matching in $G_{(A,B)}(\beta)$ that must exist by Lemma 2. Let ξ be the incidence vector of M, i.e., let $\xi_{ij} = 1$ if $(i,j) \in M$ and $\xi_{ij} = 0$ otherwise. Now ξ is binary, so if $\xi_{ij} > 0$, then $\xi_{ij} = 1$. Thus $(i,j) \in M$, by the definition of ξ, and M is a subset of the edges in $G_{(A,B)}(\beta)$, so, by the definition of a support graph, $\beta_{ij} > 0$, and we have established property (i). Since M is a perfect matching, M saturates both A and B. As M saturates B, there is exactly one arc in M incident to each node in B. Property (ii) follows. Similarly, as M saturates A, there is exactly one arc in M incident to each node in A. Property (iii) follows. □

Corollary 2 *Under the conditions of Lemma 2, if there exists* $\beta \in \mathbb{Z}_+^{|A|} \times \mathbb{Z}_+^{|B|}$ *satisfying equations (42) and (43), then* β' *defined by*

$$\beta'_{ij} = \beta_{ij} - \xi_{ij}, \quad \forall i \in A, \quad \forall j \in B, \tag{55}$$

where $\xi \in \{0,1\}^{|A|} \times \{0,1\}^{|B|}$ *is the vector guaranteed to exist by Corollary 1, then* $\beta' \in \mathbb{Z}_+^{|A|} \times \mathbb{Z}_+^{|B|}$ *and furthermore* β' *satisfies the equations*

$$\sum_{i \in A} \beta'_{ij} = \eta - 1, \quad \forall j \in B \tag{56}$$

and

$$\sum_{j\in B} \beta'_{ij} = \eta - 1, \quad \forall i \in A. \tag{57}$$

Proof of Corollary 2

By Corollary 1, ξ satisfies the three properties listed in the statement of that corollary. Now $\beta' \in \mathbb{Z}_+^{|A|} \times \mathbb{Z}_+^{|B|}$, and, in particular, $\beta' \geq 0$, since β is integer, ξ is binary, and by property (1) of Corollary 1. Also

$$\sum_{i\in A} \beta'_{ij} = \sum_{i\in A} (\beta_{ij} - \xi_{ij}) = \sum_{i\in A} \beta_{ij} - \sum_{i\in A} \xi_{ij} = \eta - 1$$

by (42) and property (2) of Corollary 1. This proves that (56) holds. Similarly,

$$\sum_{j\in B} \beta'_{ij} = \sum_{j\in B} (\beta_{ij} - \xi_{ij}) = \sum_{j\in B} \beta_{ij} - \sum_{j\in B} \xi_{ij} = \eta - 1$$

by (43) and property (3) of Corollary 1, proving that (57) holds. □

The main theorem is now presented, which shows that a feasible solution to a balanced transportation problem with identical supplies and demands can be decomposed into a sum of incidence vectors of perfect matchings in the support graph of the transportation problem solution.

Theorem 2 *Given two sets A and B of equal cardinality, $|A| = |B| = \mu$, hypothesis $H(\eta)$ holds for all $\eta \in \mathbb{Z}$ with $\eta \geq 1$.*

Proof of Theorem 2

The result will be proved by mathematical induction on η. The case $\eta = 1$ is straightforward. Suppose β satisfies the conditions of $H(1)$. Let $\zeta_{ij1} = \beta_{ij}$ for all $i \in A$ and all $j \in B$. Then (44), (45) and (46) all hold trivially, and $H(1)$ holds.

Now suppose that for some $\eta \in \mathbb{Z}$, $\eta > 1$, $H(\eta - 1)$ holds. Furthermore, suppose β satisfies the conditions of $H(\eta)$. Then, by Corollary 1, there exists a vector $\xi \in \{0,1\}^{|A|} \times \{0,1\}^{|B|}$ having the three properties given in the statement of that corollary. Let $\zeta_{ij\eta} = \xi_{ij}$ for all $i \in A$ and $j \in B$. Then properties (2) and (3) of Corollary 1 ensure that (44) and (45) hold for ζ_{ijk} with $k = \eta$.

Set $\beta'_{ij} = \beta_{ij} - \xi_{ij}$ for all $i \in A$ and $j \in B$. Now β' has been constructed here precisely as in the statement of Corollary 2, and so meets the conditions of the inductive hypothesis $H(\eta - 1)$. Thus there exists ζ_{ijk} for all $i \in A$, $j \in B$

and $k = \{1, \ldots, \eta - 1\}$ satisfying

$$\sum_{i \in A} \zeta_{ijk} = 1, \quad \forall j \in B, \quad \forall k = 1, \ldots, \eta - 1,$$

and

$$\sum_{j \in B} \zeta_{ijk} = 1, \quad \forall i \in A, \quad \forall k = 1, \ldots, \eta - 1.$$

Since it has already been established that (44) and (45) hold for ζ_{ijk} with $k = \eta$, it has been shown that ζ satisfies (44) and (45).

Now, by the inductive hypothesis,

$$\sum_{k=1}^{\eta-1} \zeta_{ijk} = \beta'_{ij}, \quad \forall i \in A, \quad \forall j \in B.$$

So

$$\sum_{k=1}^{\eta} \zeta_{ijk} = \sum_{k=1}^{\eta-1} \zeta_{ijk} + \zeta_{ij\eta} = \beta'_{ij} + \xi_{ij} = \beta_{ij}, \quad \forall i \in A, \quad \forall j \in B,$$

and it has been shown that (46) also holds.

Thus the hypothesis $H(\eta)$ holds. The result follows by the principle of mathematical induction. □

4.2 Application of Proof to PBILP

The results of the previous section are now applied to show that program variables from the PBILP models can be decomposed into student variables, having the desired properties. The proof begins with the PBILP2 and PBILP3 models, and subsequently addresses the PBILP1 model. The general result is given in Theorem 3.

Theorem 3 *Suppose y_{psb} for each $p \in \mathbf{P}$, $s \in \mathbf{S}$ and $b \in \mathbf{B}$, and w_p for each $p \in \mathbf{P}$ satisfy equations (27) (30), (31), (32) and (33). Then for every program $p \in \mathbf{P}$ with $w_p \geq 1$, there exist values for the student variables v_{nsb}, for $n \in \mathbf{N}_p$, $s \in \boldsymbol{\omega}_n = \boldsymbol{\pi}_p$, and $b \in \mathbf{B}$, so that*

$$\sum_{n \in \mathbf{N}_p} v_{nsb} = y_{psb}, \quad \forall s \in \boldsymbol{\pi}_p, \quad \forall b \in \mathbf{B}, \tag{58}$$

$$\sum_{s \in \boldsymbol{\omega}_n} v_{nsb} \leq 1, \quad \forall b \in \mathbf{B}, \quad \forall n \in \mathbf{N}_p, \tag{59}$$

$$\sum_{b \in \mathbf{B}} v_{nsb} \leq 1, \quad \forall s \in \boldsymbol{\omega}_n \quad \forall n \in \mathbf{N}_p. \tag{60}$$

Further, there exists a subset of students taking program p, $\mathbf{W}_p \subseteq \mathbf{N}_p$, with $|\mathbf{W}_p| = w_p$, such that for every student $n \in \mathbf{W}_p$ in this set,

$$\sum_{s \in \boldsymbol{\omega}_n} v_{nsb} = 1, \quad \forall b \in \mathbf{B}, \tag{61}$$

$$\sum_{b \in \mathbf{B}} v_{nsb} = 1, \quad \forall s \in \boldsymbol{\omega}_n. \tag{62}$$

Proof of Theorem 3

Let $p \in \mathbf{P}$ be a program with $w_p \geq 1$. Observe that by our standing assumptions, $\boldsymbol{\pi}_p$ and $\mathbf{B}$ are sets of equal cardinality. Define β by $\beta_{sb} = y_{psb}$ for each $s \in \boldsymbol{\pi}_p$ and $b \in \mathbf{B}$. Then $\beta \in \mathbb{Z}^{|\boldsymbol{\pi}_p|} \times \mathbb{Z}^{|\mathbf{B}|}$ by (27) and (33), and β satisfies the conditions of the hypothesis H(w_p) by (30) and (31). By Theorem 2, there exist $\zeta_{sbk} \in \{0, 1\}$ for each $s \in \boldsymbol{\pi}_p$, $b \in \mathbf{B}$ and $k \in \{1, \ldots, w_p\}$ such that

$$\sum_{s \in \boldsymbol{\pi}_p} \zeta_{sbk} = 1, \quad \forall b \in \mathbf{B}, \quad \forall k \in \{1, \ldots, w_p\}, \tag{63}$$

$$\sum_{b \in \mathbf{B}} \zeta_{sbk} = 1, \quad \forall s \in \boldsymbol{\pi}_p, \quad \forall k \in \{1, \ldots, w_p\}, \tag{64}$$

$$\sum_{k=1}^{w_p} \zeta_{sbk} = \beta_{sb}, \quad \forall s \in \boldsymbol{\pi}_p, \quad \forall b \in \mathbf{B}. \tag{65}$$

Now we may take $\mathbf{W}_p$ to simply be the first w_p students in the set $\mathbf{N}_p$; this is possible by constraint (32). Let $\sigma(n)$ be a unique integer in $\{1, \ldots w_p\}$ identified with each $n \in \mathbf{W}_p$, and set

$$v_{nsb} = \begin{cases} \zeta_{sb\sigma(n)}, & \text{if } n \in \mathbf{W}_p; \\ 0, & \text{otherwise} \end{cases}$$

for each $n \in \mathbf{N}_p$, $s \in \boldsymbol{\pi}_p$ and $b \in \mathbf{B}$. (Recall that $\boldsymbol{\omega}_n = \boldsymbol{\pi}_p$ for all $n \in \mathbf{N}_p$.) It is now obvious that equation (58) follows by equation (65) and the definitions of β and v. Equations (61) and (62) follow from equations (63) and (64) respectively, and the definition of v. Constraints (59) and (60) also follow from (63) and (64) respectively, and the definition of v. □

Theorem 3 demonstrates that the PBILP2 is valid. It is a straightforward corollary (the ν_p parameters take the role of the the w_p variables) that the PBILP3 model is valid also. The variables used in the models combine students with an identical program into one set of variables, and record the number of students taking each program that have their program fully assigned to blocks.

Theorem 3 demonstrates that these variables can be decomposed into variables for individual students, with the correct number of students having fully assigned programs. The proof also suggests how the decomposition can be performed in practice. The timetabler could solve a sequence of perfect matching problems on the supporting graph of the program variables (for example, by using the Hungarian algorithm, which finds perfect matchings in polynomial time).

It remains to show that the PBILP1 model is also valid, in the sense discussed earlier, i.e. that a decomposition of the program variables into student variables satisfying equation (58) and constraints (59) and (60) exists in this case also. However the proof of Theorem 3 relies on equality between the "flow" out of each subject and the "flow" into each block, which may not occur in the PBILP1 model. Fortunately, it is possible to adapt the solution of the PBILP1 model so as to ensure equality, with the use of dummy flows. The decomposition can be performed, dummy components removed, and the result follows. The general result is given in Theorem 4.

Theorem 4 *Suppose y_{psb} for each $p \in \mathbf{P}$, $s \in \mathbf{S}$ and $b \in \mathbf{B}$ satisfies equations (22), (23), and (27). Then for every program $p \in \mathbf{P}$ there exist values for the student variables v_{nsb}, for $n \in \mathbf{N}_p$, $s \in \boldsymbol{\pi}_p = \boldsymbol{\omega}_n$, and $b \in \mathbf{B}$, so that*

$$\sum_{n \in \mathbf{N}_p} v_{nsb} = y_{psb}, \quad \forall s \in \boldsymbol{\pi}_p, \quad \forall b \in \mathbf{B}, \tag{66}$$

$$\sum_{s \in \boldsymbol{\omega}_n} v_{nsb} \leq 1, \quad \forall b \in \mathbf{B}, \quad \forall n \in \mathbf{N}_p, \tag{67}$$

$$\sum_{b \in \mathbf{B}} v_{nsb} \leq 1, \quad \forall s \in \boldsymbol{\omega}_n, \quad \forall n \in \mathbf{N}_p. \tag{68}$$

Proof of Theorem 4

Let $p \in \mathbf{P}$, and suppose y_{psb} for each $s \in \boldsymbol{\pi}_p$ and $b \in \mathbf{B}$ satisfies equations (22), (23), and (27). Consider the transportation problem defined by the complete bipartite graph $G^p_{(\boldsymbol{\pi}_p, \mathbf{B})}$, with supply $\nu_p - \sum_{b \in \mathbf{B}} y_{psb}$ for each $s \in \boldsymbol{\pi}_p$ and demand $\nu_p - \sum_{s \in \boldsymbol{\pi}_p} y_{psb}$ for each $b \in \mathbf{B}$. This transportation problem picks up the "shortfall" from a complete allocation for the students in this program, and is

clearly balanced, as

$$\sum_{s\in\boldsymbol{\pi}_p}(\nu_p-\sum_{b\in\mathbf{B}}y_{psb})=|\boldsymbol{\pi}_p|\nu_p-\sum_{s\in\boldsymbol{\pi}_p}\sum_{b\in\mathbf{B}}y_{psb}$$
$$=|\mathbf{B}|\nu_p-\sum_{b\in\mathbf{B}}\sum_{s\in\boldsymbol{\pi}_p}y_{psb},$$

since $|\boldsymbol{\pi}_p|=|\mathbf{B}|$, and by exchange of summation order, and thus

$$\sum_{s\in\boldsymbol{\pi}_p}(\nu_p-\sum_{b\in\mathbf{B}}y_{psb})=\sum_{b\in\mathbf{B}}(\nu_p-\sum_{s\in\boldsymbol{\pi}_p}y_{psb}).$$

Furthermore, since the bipartite graph is complete, and all supplies and demands are in $\mathbb{Z}_+$ (by (22), (23) and (27)), the balanced transportation problem is feasible, and an integer solution can be found in polynomial time. Let $\delta_p\in\mathbb{Z}_+^{|\boldsymbol{\pi}_p|}\times\mathbb{Z}_+^{|\mathbf{B}|}$ denote such a solution; this can be viewed as a "dummy flow", which we will add to the y_{psb} variables to define a complete allocation: define $y'_p\in\mathbb{Z}_+^{|\boldsymbol{\pi}_p|}\times\mathbb{Z}_+^{|\mathbf{B}|}$ by

$$y'_{psb}=y_{psb}+\delta_{psb} \tag{69}$$

for all $s\in\boldsymbol{\pi}_p$ and $b\in\mathbf{B}$. Now it is obvious, since δ_p exactly meets the demand at each node of $G^p_{(\boldsymbol{\pi}_p,\mathbf{B})}$ in $\mathbf{B}$, that

$$\sum_{s\in\boldsymbol{\pi}_p}y'_{psb}=\sum_{s\in\boldsymbol{\pi}_p}y_{psb}+\sum_{s\in\boldsymbol{\pi}_p}\delta_{psb}=\sum_{s\in\boldsymbol{\pi}_p}y_{psb}+(\nu_p-\sum_{s\in\boldsymbol{\pi}_p}y_{psb})=\nu_p,\quad\forall b\in\mathbf{B} \tag{70}$$

and it is similarly obvious, since δ_p exactly achieves the supply at each node of $G^p_{(\boldsymbol{\pi}_p,\mathbf{B})}$ in $\boldsymbol{\pi}_p$, that

$$\sum_{b\in\mathbf{B}}y'_{psb}=\sum_{b\in\mathbf{B}}y_{psb}+\sum_{b\in\mathbf{B}}\delta_{psb}=\sum_{b\in\mathbf{B}}y_{psb}+(\nu_p-\sum_{b\in\mathbf{B}}y_{psb})=\nu_p,\quad\forall s\in\boldsymbol{\pi}_p. \tag{71}$$

Thus y'_{psb} satisfies the conditions of hypothesis $H(\nu_p)$ and Theorem 2 can be applied to deduce that there exists $v'_{nsb}\in\{0,1\}$ for each $n\in\mathbf{N}_p$, (recall that $|\mathbf{N}_p|=\nu_p$), $s\in\boldsymbol{\pi}_p$ and $b\in\mathbf{B}$ such that

$$\sum_{s\in\boldsymbol{\pi}_p}v'_{nsb}=1,\quad\forall n\in\mathbf{N}_p,\quad\forall b\in\mathbf{B}, \tag{72}$$

$$\sum_{b\in\mathbf{B}}v'_{nsb}=1,\quad\forall n\in\mathbf{N}_p,\quad\forall s\in\boldsymbol{\pi}_p, \tag{73}$$

$$\sum_{n\in\mathbf{N}_p}v'_{nsb}=y'_{psb},\quad\forall s\in p,\quad\forall b\in\mathbf{B}. \tag{74}$$

Consider a subject $s \in \boldsymbol{\pi}_p$ and a block $b \in \mathbf{B}$. By equations (69) and (74)

$$\sum_{n \in \mathbf{N}_p} v'_{nsb} = y'_{psb} = y_{psb} + \delta_{psb}.$$

Define Δ_{psb} to be any subset of $\{n \in \mathbf{N}_p \ : \ v'_{nsb} = 1\}$ having cardinality $|\Delta_{psb}| = \delta_{psb}$; the above equation (together with the integrality and non-negativity conditions that follow from the definitions of y and δ), ensures that this set is well defined. Now define

$$v_{nsb} = \begin{cases} v'_{nsb}, & n \in \mathbf{N}_p \setminus \Delta_{psb} \\ 0 & n \in \Delta_{psb} \end{cases}$$

for each $n \in \mathbf{N}_p$. Obviously

$$v_{nsb} \leq v'_{nsb}, \quad \forall n \in \mathbf{N}_p, \tag{75}$$

and furthermore, since $v_{nsb} = v'_{nsb}$ except for $n \in \Delta_{psb}$, in which case $v_{nsb} = 1$ and $v'_{nsb} = 0$, and since $|\Delta_{psb}| = \delta_{psb}$,

$$\sum_{n \in \mathbf{N}_p} v_{nsb} = \sum_{n \in \mathbf{N}_p} v'_{nsb} - \delta_{psb} = y_{psb}. \tag{76}$$

Now since (75) holds for every subject $s \in \boldsymbol{\pi}_p$ and every block $b \in \mathbf{B}$, and from equations (72) and (73), constraints (67) and (68) follow. Similarly, since equation (76) holds for every subject $s \in \boldsymbol{\pi}_p$ and every block $b \in \mathbf{B}$ equation (66) must also hold. Since program $p \in \mathbf{P}$ was chosen arbitrarily, the theorem follows. □

It has now been confirmed that the use of general integer variables to model the activities of all students taking a distinct program is valid, i.e. can be decomposed into activities for individual students, satisfying reasonable conditions.

5 Results

The PBILP model is useful for solving the CBPP faced by each year level of an Australian secondary school, the senior campus of Xavier College, Melbourne, which we refer to for brevity as Xavier College. We use real data for the year 2002; minor changes to numbers of students and other matters occur from year to year, but the 2002 data is representative of the typical complexity of the Xavier College problem, and of the analogous problem at other schools of comparable size and academic profile.

Table 2: **Xavier College individual year level data sets.**

Year	Subjects	Classes	Students	Programs	Teachers	Blocks
9	15	40	220	58	25	3
10	16	41	234	97	26	3
11	33	96	227	180	53	6
12	32	90	211	186	68	6

Xavier College has four year levels and in each of the year levels there is a subset of subjects (the elective subjects) from which each student will select a specified number to study. In year 12 a student chooses up to 6 of the 32 elective subjects. For each of these subjects the class populations need to be determined, and the classes must be allocated to blocks (a CBPP needs to be solved). Importantly each elective subject requires the same number of lessons, satisfying the assumption necessary for using the PBILP model. Therefore it is appropriate to solve the CBPP for this set of subjects for each year level of the school (separately) using the PBILP model.

The details for each data set from each year level are shown in Table 2. Students have much greater freedom in choosing electives in the upper year levels (11 and 12) than in the lower year levels (9 and 10). In the lower year levels, each student has three electives (from a set of 15 in year 9, and 16 in year 10). In the upper year levels, there are six electives (chosen from 32 in year 11, and 33 in year 12). Consequently the problem for the two upper year levels is more complex than that for the lower two year levels. The data sets for each year level were taken from an existing timetable for the school, and consequently it was known that a maximal feasible solution was possible (all students allocated to a class for all of the subjects in their programs and all classes allocated to blocks).

Upon inspection of the data sets, it is clear that the major benefit of having integer variables for each program, compared to binary variables for each student, comes in years 9 and 10. There the smaller number of subjects per student (each only has to choose three subjects) results in a smaller range of individual programs (58 and 97 programs for 220 and 234 students in years 9 and 10 respectively). The integer variables do not produce the same economy

Table 3: **Numbers of variables and constraints in Xavier College data for the three PBILP models with and without combined variables.**

	Combined variables		Separate variables	
Year	Variables	Constraints	Variables	Constraints
	PBILP1			
9	651	556	2188	556
10	1008	801	2241	801
11	7032	2966	8724	2966
12	7314	3367	8430	3367
	PBILP2			
9	709	556	2339	1534
10	1105	801	2475	1623
11	7212	2966	8951	3530
12	7500	3367	8467	3739
	PBILP3			
9	696	1534	2163	1534
10	1056	1623	2289	1623
11	7320	3530	8922	3530
12	7506	3739	8622	3739

for the upper year levels (180 and 186 programs for 227 and 211 students in years 11 and 12 respectively). This is illustrated more clearly in Table 3 where the number of variables and constraints for each of the models is presented. The model with combined integer variables always uses less variables and constraints, but the effect is less marked for the upper year levels. However any reduction should reduce the search space, and the reductions in problem size should reduce LP solve time.

The models presented in Section 3 were implemented in the modeling language AMPL, and solved using the ILOG package Cplex 8.0 on a XP900 Alphastation (667Mhz CPU). In setting the reward/penalty values for use in the PBILP model objective functions, it was deemed more important for a class to have a teacher than for a student to not be allowed to take a subject from their program, and consequently forced to change one of their subjects (the school's

policy for infeasible solutions). It was deemed equally important for class capacity to be exceeded by one student as for one student to have one subject unallocated. It was thus decided that for this problem appropriate values for α, β, γ and ω were $\alpha = 1$, $\beta = 100$, $\gamma = 1$ and $\omega = 1$. As was stated previously, because these data sets were known to have a feasible solution, any positive values for these parameters should produce a feasible (i.e., optimal) solution.

As expected, a feasible solution to the CBPP problem was found for all models on all data sets, with every student and teacher allocated to a class for all subjects in their program. The times taken by Cplex to find the optimal solutions for all the combined student PBILP models are presented in Table 4 (the 'Combined' columns). The data sets with a smaller number of subjects and blocks (years 9 and 10) were noticeably faster for Cplex to find the optimal solution than the later year levels with more subjects and blocks. Of particular note is that the third model is noticeably faster on three of the four data sets.

The separate student PBILP models were also implemented in AMPL and tested on the same data sets. The results are presented in Table 4 under the columns labelled 'Separate'. The results are somewhat unclear, and unexpected. Each approach (combined integer variables versus separate binary variables) produces the fastest solution in half the cases. Although it is clear that the introduction of the general integer variables should dramatically reduce the symmetry in the model, we do not see dramatic improvements in the run times for Cplex. We suggest three reasons for this.

(i) Cutting planes for general integer problems have been less well explored in the literature, whereas a great deal of effort has gone into developing good cutting planes for binary problems. Of the 7 types of cutting planes automatically generated and reported by Cplex for the PBILP models, only two apply in general integer problems (mixed integer rounding cuts and Gomory cuts). For each year level, Cplex generated more clique constraints (which are typically strong) for individual variables than for combined variables (this is especially marked in years 9 and 10). However the types of cut listed by Cplex are not the only cuts used by this program[5]. It is unknown how many other types of cut are made or their number and the impact this would have on the separate

[5]For example, for the year 11 data for PBILP1 with separate binary variables, Cplex reports 664 cuts at the root node. However, it only lists 426 cuts under the 7 reported types of cut for all nodes.

Table 4: **Results for Xavier College data for the three PBILP models with and without combined variables.** For each of the four year levels, we present the run time in seconds and the number of branch and bound nodes with or without the use of combined student variables. The fastest run times for a given year level are shown in boldface type.

	Combined		Separate	
Year	Nodes	Time	Nodes	Time
	PBILP1			
9	7	0.3	12	1.7
10	15	0.9	20	3.6
11	580	83.3	347	79.0
12	610	118.4	244	56.1
	PBILP2			
9	14	0.3	8	1.0
10	20	0.6	0	**0.3**
11	610	117.9	150	54.0
12	300	**38.6**	130	**38.6**
	PBILP3			
9	10	**0.2**	0	**0.2**
10	42	0.5	20	1.0
11	460	**42.1**	8050	344.9
12	1478	66.0	240	44.5

or combined variables.

(ii) Cplex has some inbuilt capacity for dealing with symmetry and for automatically reducing the negative effects of symmetry (Robert Bixby, personal communication, 2004).

(iii) As we will see below, fully integer solutions can be obtained on these data sets without requiring the y_{psb}, or v_{nsb} variables, respectively, to be integer. The symmetry reduction is only important for branch-and-bound; if not much branching is required on these variables, the effect of symmetry reduction will be marginal.

In other words, it is conjectured that Cplex does a much better job of solving binary integer models, as opposed to general integer models, even when the former are relatively poor. This may be less marked for other solvers, and

as the field of general integer programming matures, this difference may well disappear. Furthermore, when little branching is required on the student variables, little difference will be observed. This may be less marked for other data sets. Furthermore, as we see below, the combined variables have a considerable advantage when integrality is relaxed, as they induce smaller LPs.

As we discussed in Section 3, in the PBILP models the variables for student allocation (y_{psb}) and teacher allocation (z_{tsb}) are integer variables, but are based on a network structure; if the x_{sb} variables are determined, then the remaining problem consists of separate network flow problems, with joint capacity constraints (like multi-commodity flow). This observation motivated us to see if priority branching on the x_{sb} variables would speed up solution times. As a first step, we relaxed the integrality of the y_{psb} (v_{nsb}) and z_{tsb} variables, and solved each of the data sets. The results are presented in Table 5, and should be compared to the equivalent results in Table 4. No single student or teacher allocation variable has a non-integer value in any of the solutions generated by any model on all eight data sets.

There are three interesting points in these results. The first is that the relaxed problems almost always require less time and fewer branch and bound nodes in order to be solved to optimality than the integer or binary problems. This would be expected as the problems are relaxations. Interestingly, the second model (with combined variables) is noticeably faster than the others, and by far the superior model. Another important result is that the separate variables for each student do not seem to benefit as much from the relaxation as the combined variables. All models with combined variables provide faster solutions than the models with separate variables on all data sets. This is quite understandable: there is essentially no difference in their integer structure, but the combined variable model has smaller LPs, that will be faster to solve.

6 Conclusion

We have demonstrated that, with the models presented in this paper, standard integer programming techniques can solve pure blocking CBPPs generated by a local high school, with the best model and priority branching combination solving for every year level in under 4 seconds, without the need for branching.

In real high school timetabling, the assumptions of pure blocking do not

Table 5: **Results for Xavier College data for the three PBILP models with and without combined variables, when the integrality of the student and teacher allocation variables (y_{psb} v_{nsb}, z_{tsb}) is relaxed.** For each of the four year levels, we present the run time in seconds and the number of branch and bound nodes with or without the use of combined student variables. The fastest run times for a given year level are shown in boldface type.

	Combined		Separate	
Year	Nodes	Time	Nodes	Time
	PBILP1			
9	0	0.1	0	0.3
10	18	0.5	0	0.9
11	380	85.2	300	57.6
12	93	34.9	90	30.4
	PBILP2			
9	0	**0.0**	0	0.2
10	0	**0.1**	0	0.4
11	0	**2.8**	0	7.1
12	0	**3.9**	0	4.1
	PBILP3			
9	0	**0.0**	0	0.1
10	0	0.3	20	0.7
11	420	41.6	720	47.9
12	90	21.4	90	30.3

hold. However the models we have presented here can readily be modified, and applied to elements of a more complex problem. As is shown by Merlot [12], in conjunction with a suite of other, related, models, all the complexities of real world timetabling can be accommodated.

In future work, we would like to consider relaxing some of the pure blocking assumptions, and adapting these ideas for use in much larger, university timetabling problems. In such problems, the number of blocks may not be known in advance, and the symmetry of blocks may introduce further difficulties. However column generation with respect to blocks may offer the solution

to both these problems, and is a direction of further research.

Acknowledgement

The authors thank Xavier College for the supply of the data used to illustrate the methods developed in this paper.

References

[1] R. Alvarez-Valdes, E. Crespo and J.M. Tamarit. Assigning Students to Course Sections using Tabu Search. Annals of Operations Research 2000; 96; 1–16.

[2] R. Alvarez-Valdes, E. Crespo and J.M. Tamarit. Design and Implementation of a Course Scheduling System using Tabu Search. European Journal of Operational Research 2002; 137; 512–523.

[3] J. Aubin and J.A. Ferland. A Large Scale Timetabling Problem. Computers and Operations Research 1989; 16; 67–77.

[4] D. Banks, P. van Beek, and A. Meisels. A Heuristic Incremental Modelling Approach to Course Timetabling. Proceedings of the 12th Canadian Conference on Artificial Intelligence, 1998, pp. 16–29,

[5] J. Bondy and U. Murty. Graph Theory with Applications. London: Macmillan, 1977.

[6] M. Carter. A Comprehensive Course Timetabling and Student Scheduling System at the University of Waterloo. In: Burke E.; Erben W. (eds.): Practice and Theory of Automated Timetabling III Third International Conference, PATAT 2000, Konstanz, Germany, August 16–18, 2000. Selected Papers. Lecture Notes in Computer Science 2079, Springer-Verlag, Berlin Heidelberg New York, 64–82, 2001.

[7] J.A. Ferland and C. Fleurent. SAPHIR: A Decision Support System for Course Scheduling. Interfaces 1994; 24(2); 105–115.

[8] P. Hall. On Representation of Subsets. Journal of the London Mathematical Society 1935; 10; 26–30.

[9] A. Hertz. Tabu Search for Large Scale Timetabling Problems. European Journal of Operational Research 1991; 54; 39–47.

[10] A. Hertz and V. Robert. Constructing a Course Schedule by Solving a Series of Assignment Type Problems. European Journal of Operational Research 1998; 108; 585–603.

[11] G. Lewandowski. Practical Implementations and Applications of Graph Coloring. PhD Thesis, University of Wisconsin–Madison, USA, 1994.

[12] L.T.G. Merlot. Techniques for Academic Timetabling. PhD Thesis, University of Melbourne, Australia, 2005.

[13] V. Robert and A. Hertz. How to Decompose Constrained Course Scheduling Problems into Easier Assignment Type Subproblems. In: Burke, E.; Ross, P. (eds.): Practice and Theory of Automated Timetabling, First International Conference, Edinburgh, U.K., August/September 1995. Selected Papers. Lecture Notes in Computer Science 1153, Springer-Verlag, Berlin Heidelberg New York, 364–373, 1996.

[14] H. Rudova and L. Matyska. Constraint-Based Timetabling with Student Schedules. In: Burke E.; Erben W. (eds.): Proceedings of Practice and Theory of Automated Timetabling III Third International Conference, PATAT 2000, Konstanz, Germany, August 16–18, 109–123, 2000.

[15] S.E. Sampson, J.R. Freeland, and E.N. Weiss. Class Scheduling to Maximize Participant Satisfaction. Interfaces 1995; 25(3); 30–41.

[16] A. Tripathy. Computerised Decision Aid for Timetabling — A Case Analysis. Discrete Applied Mathematics 1992; 35; 313–323.

[17] B. van Kesteren. The Clustering Problem in Dutch High Schools: Changing Metrics in Search Space. Masters Thesis, Leiden University, The Netherlands, 1999.

[18] R.J. Willemen. School Timetable Construction: Algorithms and Complexity. PhD Thesis, Technische Universiteit Eindhoven, The Netherlands, 2002.

[19] J. Wood and D. Whitaker. Student Centered School Timetabling. Journal of the Operational Research Society 1998; 49; 1146–1152.