Noname manuscript No.
(will be inserted by the editor)

How Different is the Integrated Healthcare Timetabling Competition 2024 Model? (Draft)

Jeffrey H. Kingston

Received: date / Accepted: date

Abstract IHTC2024, the Integrated Healthcare Timetabling Competition 2024, is the first attempt to bring the complexities of real-world health-care optimization to general notice. While IHTC2024 represents a real departure, it does not follow that its model is fundamentally different from other timetabling models. This paper studies this issue by attempting to convert the IHTC2024 model to XESTT, an existing employee scheduling model. It will turn out that conversion is not feasible, but the attempt will give insight into how IHTC2024 differs from other problems, and how XESTT could be further developed.

Keywords Automated timetabling · IHTC2024 · XESTT

1 Introduction

The Integrated Healthcare Timetabling Competition 2024 (IHTC2024) is the first attempt to bring the complexities of real-world health-care optimization to general notice. The innovation is that several different timetabling problems must be solved together. In IHTC2024, there are three: surgical case planning, patient admission scheduling, and nurse-to-room assignment. Nurse rostering is assumed to have been completed before the IHTC2024 problem begins.

For some time this author has been concerned at the way that every new timetabling problem in the employee scheduling area seems to be expressed using its own separate model. This is not inevitable. One can look to XHSTT in high school timetabling, XESTT in nurse rostering, and RobinX in sports scheduling, to see examples of models that include everything that a wide variety of real-world instances need for their expression.

J. Kingston (`http://jeffreykingston.id.au`)
School of Information Technologies, The University of Sydney, Australia
E-mail: jeff@it.usyd.edu.au

To understand the issues clearly here it is necessary to distinguish two kinds of models. A *high-level model* expresses a problem in terms familiar to end users. A *low-level model* expresses a problem in a form ready for solving.

For example, many timetabling papers define a problem informally and then express it formally as an integer programming problem. The informal description is a high-level model, integer programming is a low-level model. But a high-level model need not be informal. The IHTC2024 model, for example, is a high-level model expressed formally.

Given these two levels, one can envisage two strategies for handling a new problem like IHTC2024. A new high-level model is needed to capture the end user's requirements, but after that we can either start from scratch and write a fresh solve platform for the high-level model, or we can devise a conversion to an existing low-level model with an existing solve platform.

Provided the conversion is natural (does not introduce any unnecessary complications), it is undeniable that the second strategy is superior. It requires less programming, and it links the new problem with a solve platform which is likely to be more mature, and thus more capable, than anything that can be developed in a limited time for the new high-level model. For example, converting every high-level model to an integer programming model and using an IP solver to solve it would be an excellent strategy if only the conversions were natural and IP solvers were up to the solving task.

The second strategy has another advantage: it provides insight into the nature of the problems being studied. For example, a paper that converts a problem to an integer program might comment that, thus expressed, the problem appears to be bin packing with some side constraints. However, integer programming is able to express many problems that are quite unrelated to timetabling, and some conversions from timetabling to integer programming are not very natural (we are thinking of constraints, common in nurse rostering, which limit the number of consecutive busy days in a nurse's timetable), so as a vehicle for understanding timetabling problems its value is limited.

This paper investigates these issues by attempting to convert instances in the IHTC2024 model into instances in the XESTT model, a low-level employee scheduling model with a well-developed solve platform. It will turn out that conversion is not feasible, but the attempt will give insight into how IHTC2024 differs from other employee scheduling problems, and how XESTT could be further developed.

2 A brief overview of the IHTC2024 model

In this section we present a brief overview of the IHTC2024 model. A full description is available on the IHTC2024 web site [].

remainder still to do

3 A brief overview of the XESTT model

In this section we present a brief overview of the XESTT model. A complete technical description is available online, accessible from the front page of the HSEval web site [].

remainder still to do

4 Converting from IHTC2024 to XESTT

This section attempts to convert IHTC2024 to XESTT. Here, 'IHTC2024' will often mean the IHTC2024 documentation [], and 'XESTT' will often mean the XESTT documentation []. When we speak of defining something, we mean that we are adding it to the XESTT instance which is the result of the conversion of an IHTC2024 instance. Paragraphs which identify a problem with the conversion are marked by an asterisk.

We use the term *surgery* for anything related to the surgery that the patient has on his first day, and *recovery* for anything related to the rest of his stay.

Because this is a paper and pencil exercise, there is no way to prove that the conversion is complete. The author has read the IHTC2024 documentation diligently and has done his best to ensure that nothing is left out. If there are omissions, that will not change the thrust of the findings.

4.1 Times

The scheduling period consists of D consecutive days. To allow for patients who are admitted towards the end of the scheduling period, the conversion calls these days *cycle days* ('cycle' is a synonym for 'scheduling period'), and it adds E consecutive days after them, called *post-cycle days*. E is the maximum, over all patients p, of p's length of stay, or 0 if there are no patients. IHTC2024 requires D to be 14, 21, or 28, but the conversion handles any positive integer.

Each day (cycle or post-cycle) contains three shifts, called *early*, *late*, and *night*. So the conversion defines three times for each day. For example, if the first day is `1Mon`, then the three times on that day are `1Mon1`, `1Mon2`, and `1Mon3`, representing the times of the early, late, and night shifts on that day. This arrangement is used by XESTT for nurse rostering, and has been adopted because it is familiar, despite the fact that IHTC2024 does not specify which day of the week the first day lies on.

IHTC2024 does not say that a patient cannot be in an operating theatre and a room at the same time. One gathers that what happens in operating theatres is so uncertain that it is necessary to plan for a recovery that could begin at any time on the day of admission. However that may be, we express this situation by adding one extra time to each day (for example, we add `1Mon0` to `1Mon`), and assigning this time to each surgery that occurs on that day. This extra time precedes the other times on its day. We call it a *surgery time*, and

we call the other times *recovery times*. There are surgery times on post-cycle days, for uniformity, but hard constraints prevent any surgery events from being assigned to them.

4.2 Resources

Five resource types are defined: `Patient`, `Surgeon`, `OperatingTheatre`, `Nurse`, and `Room`. One resource of type `Patient` is defined for each patient, one resource of type `Surgeon` is defined for each surgeon, one resource of type `OperatingTheatre` for each operating theatre, one resource of type `Nurse` for each nurse, and one resource of type `Room` for each room.

Each resource has a number of attributes: rooms have a number of beds; nurses have a skill level, a roster, and a maximum workload per shift; and so on. XESTT does not support resource attributes directly, but there are ways to express them using resource groups, constraints, and so on.

In most timetabling problems, each resource is accompanied by an avoid clashes constraint, saying that it cannot attend two events at the same time. These are not always appropriate here, due to the uncertainty about exactly what surgeons and operating theatres are doing at any given time, and to the fact that a nurse can supervise a whole roomful of patients simultaneously.

4.3 Events

We define two kinds of events.

A *surgery event* represents one patient's surgery. The event contains one unpreassigned surgery time, one preassigned patient, one preassigned surgeon, and one unpreassigned operating theatre. Assigning a time and an operating theatre determines the patient's admission date and operating theatre—items (i) and (iv) of the solution in IHTC2024.

A *recovery event* represents one patient's recovery during one shift. The event contains one unpreassigned recovery time, one preassigned patient, one unpreassigned room, and one unpreassigned nurse. The number of recovery events for a given patient is equal to the number of shifts that that patient needs for recovery: three times his length of stay.

A solver does not need to assign times to recovery events; their times are determined by structural order events constraints (see below). A mature solve platform (such as the author's KHE platform) would understand this and would not give a solver the opportunity to assign these times, instead assigning them automatically when the surgery event is assigned.

Assigning a room determines the patient's room—solution item (ii).

Assigning a nurse determines the nurse to room assignment—item (iii). This same nurse must appear in every recovery event with the same time and room. This is awkward and is discussed further below.

Occupants (patients already in the hospital when the scheduling period begins) have recovery events but not surgery events, and their recovery events

have preassigned times. Patients who stay after the end of the scheduling period have recovery events which are assigned post-cycle times.

4.4 Structural constraints

IHTC2024 sometimes says that certain things are fixed and cannot be changed. In the conversion, some of these things are not fixed in that absolute sense, but a hard constraint of large weight is violated if they are changed. We regard this as near enough in practice. Constraints of this kind in the converted XESTT instance are presented in this section. We call them *structural constraints*, to distinguish them from *converted constraints*, which are the converted forms of the constraints (**H1**, **S1** and so on) explicitly given and named in IHTC2024.

A surgery event cannot be assigned a recovery time, nor a surgery time in the post-cycle period. This is enforced by a structural prefer times constraint, containing the surgery event and the set of all times which are both cycle times and surgery times. This does not require the event to be assigned a time, but it does constrain which time is assigned if it is assigned at all.

Each recovery event is linked to its patient's previous recovery event by a structural hard order events constraint, which forces it to be assigned the recovery time following the previous event's time. The first recovery event is linked to its patient's surgery event, again by a structural order events constraint. The effect of these constraints is to determine the times of all of a patient's recovery events, when his surgery event is assigned a time. Post-cycle recovery times are available for use by recovery events.

The same room must be assigned in each recovery event for a given patient. This is easily implemented by a structural avoid split assignments constraint containing the room event resources of the patient's recovery events.

The recovery events of occupants have preassigned times and hence no need for order events constraints. Patients who stay after the end of the scheduling period have recovery events which are assigned post-cycle times, but no constraints apply in that region so no costs are incurred.

A nurse can only attend recovery events at times in her (pre-determined) roster. This is enforced by a structural avoid unavailable times constraint containing the nurse and the set of all times that are not in her roster. The roster will ensure that she does not work more than one shift per day.

The author jumped to the conclusion that a nurse can supervise at most one room during any one shift. However this is not stated anywhere, the wording of **S4** suggests the opposite, and the solution example in Appendix 1 makes it definite that a nurse can supervise multiple rooms. So we do not need a constraint limiting her to a single room per shift.

IHTC2024 says 'Some operating theatres might be unavailable on specific days, indicated by a maximum capacity of 0 minutes on those days.' We could add a hard avoid unavailable times constraint to enforce this, but the constraint that enforces operating theatre capacities is already hard (**H4**), so we don't need anything further here.

IHTC2024 says 'Rooms are characterized by their capacity, expressed in terms of the number of beds.' But the author can find no statement saying that the number of patients in a room at one time cannot exceed the number of beds. Presumably the intention is that a solution is invalid if this limit is exceeded, in which case we need a structural constraint enforcing it. One limit busy times constraint for each time and each room, whose maximum limit is the number of beds, does the job.

***** For each recovery time t and each room r, each recovery event containing t and r must contain the same nurse. The basic idea, sometimes called 'resource constancy', is expressed by the XESTT avoid split assignments constraint, but that constraint applies to a given fixed set of events. Here we choose the events based on the unpreassigned values t and r, which is a problem for XESTT.

An approach better suited to this requirement is to have one recovery event for each recovery time t and each room r, containing one preassigned time t, one preassigned room r, one unpreassigned nurse, and a number of unpreassigned patients equal to the number of beds. Then there is only one nurse assignment for each time and room, and no possibility of inconsistency. This is arguably more natural than what we have now. However, it does lead to other problems. For example, XESTT offers no way of summing the workloads of the patients assigned to this event. And although it is possible to ensure that the recovery events for a given patient do not clash (use an avoid clashes constraint), are correct in quantity (use a limit busy times constraint), and are consecutive in time (use a limit active intervals constraint), a solve platform would struggle to deduce from these constraints that there is only one independent time assignment to make (the surgery time) for each patient.

4.5 Converted constraints

In this section we convert the constraints listed explictly in IHTC2024 into XESTT constraints that we have previously called converted constraints. Each XESTT constraint can be hard or soft, and can have any weight, so there are no problems assigning suitable weights.

*** H1 (no gender mix)**. Genders are easily expressed in XESTT: define a `Male` resource group containing the male patients, and a `Female` resource group containing the female patients. But the constraint itself is a problem, since it requires gathering the recovery events with a given time and room and comparing the genders of their patients.

The alternative kind of recovery event given above, where there is one event for each time and room, is a better starting point, since the relevant patients are all present in that event. XESTT has a limit resources constraint which is able to count the number of patients in the event which lie in a given resource group. So one constraint could count the number of male patients and another could could count the number of female patients, but XESTT offers no way to report a cost when both of these numbers are non-zero.

There is one possible way out. Define a large number of resource groups, each containing one male patient and one female patient, and define one limit resources constraint for each resource group, with maximum limit 1. Then a cost will be incurred every time a recovery event contains a male and a female patient. However, this is too verbose to deserve serious consideration.

H2 (compatible rooms). For each patient p, define a resource group g holding p's compatible rooms. Add a prefer resources constraint containing g, applicable to the room event resources of p's recovery events.

*** S1 (age groups)**. This is reminiscent of **H1**, although here there are more than two classes of patient, and an integer deviation is needed.

S2 (minimum skill level). For each skill level, define a resource group containing all nurses which have that skill level or higher. For each recovery event, define a prefer resources constraint that applies to the nurse event resource of that event and requires a nurse from the skill level resource group corresponding to the skill level from the appropriate position in the patient's minimum skill level vector.

As described, this produces a cost when the nurse has an unsufficient skill level, but the cost is not proportional to the degree of insufficiency. This can be fixed by having one prefer resources constraint for each skill level, with weight appropriate to that level.

S3 (continuity of care). Define one avoid split assignments constraint for each patient, referencing the nurse event resources of the events to which the patient is preassigned. There may be a problem calculating the exact cost desired, since the constraint assigns a cost whenever the number of distinct resources assigned to the nominated event resources exceeds 1, even though (as IHTC2024 states) it is clear that the best possible number is 3.

S4 (maximum nurse workload). For each nurse and each recovery time in that nurse's roster, define a limit workload constraint for that nurse at that time whose maximum limit is the nurse's maximum workload. The workload of each nurse event resource w is the workload of the patient preassigned to the event containing w.

H3 (surgeon overtime). For each surgeon and each surgery time, add one limit workload constraint for that surgeon at that time whose maximum limit is the surgeon's maximum daily surgery time. The workload of each surgeon event resource x is the surgery time of the patient preassigned to the surgery event containing x.

H4 (operating theatre overtime). For each operating theatre and each surgery time, add one limit workload constraint for that operating theatre at that time whose maximum limit is the operating theatre's maximum capacity. The workload of each operating theatre event resource y is the surgery time of the patient preassigned to the event containing y.

S5 (open operating theatres). These are handled as for **H4**, except that the maximum limit is 0 and the cost function is a step function, so that a single fixed cost is paid for any workload greater than 0.

*** S6 (surgeon transfer)**. There is a problem here. The basic idea is implemented by an avoid split assignments constraint specifying operating

theatre event resources, but that constraint applies to a fixed set of events, whereas here we need to apply it to the set of events to which a given resource (the surgeon) is assigned on a given day.

H5 (mandatory vs. optional patients). For each mandatory patient, add a hard assign time constraint requiring the patient's surgery event to be assigned a time. For each optional patient, do not add such a constraint.

H6 (admission day). For each patient, add a hard prefer times constraint requiring the time assigned to the patient's surgery event to be a surgery time on or after the patient's release date. If the patient is a mandatory patient, this constraint also requires the time to be on or before the patient's due date.

S7 (admission delay). For each patient p, add one soft prefer times constraint for each day d after p's release date, preferring surgery times on or before d. If p is admitted k days after his release date, k of these constraints will be violated, producing the appropriate cost.

Alternatively, for each patient, add a dummy event preassigned to that patient's release date, and an order events constraint linking the dummy event to the patient's surgery event with maximum distance 0. However this produces a deviation of four times the number of days difference, which may not be easy to convert into the correct cost.

S8 (unscheduled patients). For each surgery event without a hard assign time constraint from **H6**, add a soft assign time constraint.

5 Discussion

Our conversion has four problems, indicated in the preceding presentation by asterisks. They all seem insurmountable and any one of them is enough to prevent an exact conversion from IHTC2024 to XESTT. Nevertheless, there are several useful lessons to be learned.

The dogma that timetabling is about assigning (or preassigning) times and resources to events lies at the foundation of the conversion presented here. It is disconcerting to find that there is more than one way to express IHTC2024 using events (Section 4.4); it weakens the claim of any one of those ways to be natural and inevitable.

There can be slight differences in how a deviation is calculated. Is it the number of distinct nurses, or that number minus 3? Is it the number of days separating two events, or the number of times? Such details can trip up a conversion despite being of no real importance.

Constraints could be made much more general at very little implementation cost if each was expressed using three independent parts: first a selection of a set of events or event resources; then a deviation function which takes the set and produces an integer (cardinality, say, or number of distinct assignments); and finally a cost function applied to the deviation. In XESTT the third step is independent but the first two are not.

Allowing resources to have attributes would be useful, but hard to carry through. Declarations of the attributes could be added easily enough, but then expressions which access attributes would be needed within constraints.