International Journal of Innovative Computing, Information and Control
Volume **17**, Number **2**, April **2021**

ICIC International ©2021 ISSN 1349-4198
pp. **613–638**

STEMMING THE EDUCATIONAL TIMETABLE PROBLEMS

Tuga Mauritsius, Faisal Binsar and Nilo Legowo

Information Systems Management Department
BINUS Graduate Program – Master of Information Systems Management
Bina Nusantara University
JL. K. H. Syahdan No. 9, Kemanggisan, Palmerah, Jakarta 11480, Indonesia
{ tmauritsus; nlegowo }@binus.edu; faisal.binsar@binus.ac.id

Received September 2020; revised January 2021

Abstract. *Educational Timetabling Problem (ETP) is a combinatorial problem, which is known to be NP-complete and is defined as the assignment of a set of academic transactions to resources (timeslots and rooms) subject to a set of constraints. Many attempts have been reported to solve timetabling problems. They are, however, lack of generality, where one solver is designed to solve one specific problem only. The focus of this paper is to propose a framework that is general enough to be used in any ETP. It is shown that in general, any ETP can be approached using the framework. In this context, two main fundamental phases are deeply explored in the paper. The first one deals with preprocessing steps, where the raw data is handled to output the stemmed form of an ETP. Having represented in this way, a method designed to solve one problem can be reused to solve another problem. The second one, as part of the framework, the paper also recommends an enhanced simulated annealing-based search in finding the solution for an ETP. The proposed framework is tested over some popular instances found in the literature, including real instances collected from an Indonesian college. The framework is applied to all tested instances and the qualities of the solutions are very competitive, validating the significant contribution of the proposed framework.*

Keywords: Heuristic, Metaheuristic, Simulated annealing, Local search, Timetabling, Scheduling

1. **Introduction.** A special type of scheduling problem is known as Educational Timetabling Problem (ETP) [1,2]. This problem is usually faced by educational institutions such as primary and secondary schools and tertiary educational institutions such as universities and colleges. An ETP coordinates large numbers of academic activities involving many parties, students, and teachers and must satisfy several requirements. Because of these, most of the timetabling problems are known to be NP-complete, meaning that there is no algorithm to solve most of the problems efficiently.

Since the first publication by Gotlieb in 1963 [3], in which computer programming was used to solve one of the problems found in an educational institution, researchers in ETP have been rising significantly. There have been a huge number of research papers, conferences, and workshops specially dedicated to ETP [4].

In general, ETPs are classified based on the type of educational institution for which the methods are developed. Under this perspective, ETPs are divided into three sub-problems.

- **Class-Teacher Timetabling Problems (CTTP)** or **School Timetabling** is the problem of assigning teachers and classes to timeslots. The inputs contain a set of classes (a fixed number of students who are taking almost the same programs) and a set of teachers. The assignment of teachers to classes presumably has already been

DOI: 10.24507/ijicic.17.02.613

made in advance. It is not part of the problem. The problem is to assign each class to a timeslot, such that no teacher is responsible for more than one class in the same timeslot. In addition, no class should be taught by more than one teacher in any given timeslot [5].

- **University Course Timetabling Problem (UCTP).** The inputs of this problem comprise a set of courses, a set of rooms, and a set of timeslots. The objective is to assign courses to rooms and timeslots avoiding some constraints. The assignment of instructors to courses and students to courses are not parts of the problem. Unlike in CTTP, in general, a UCTP does not recognize classes, i.e., a fixed group of students who are taking the same courses and sharing the same rooms most of the time. Thus, the number of students enrolled in each course can vary, and students may attend courses in different buildings or even on different campuses [6,7].
- **Examination Timetabling Problem (ExTP).** ExTP is the assignment of exams to timeslots and rooms, and the main requirement is that no students nor invigilators should be assigned to more than one room at the same time. ExTP differs from UCTP in two critical aspects. First, there is no repetition for an exam, and second, a room can be used by two or more exams at the same time. There are two well-known variants of ExTPs which are the un-capacitated and capacitated ones. In the un-capacitated version, the number of students and exams in any time slot is unlimited. Meanwhile, the capacitated version imposes limitations on the number of students assigned to every timeslot [8].

Despite the huge number of publications in this area, it is unusual to find a paper that deals with all the subproblems. Almost all authors tend to focus on a few specific ones [6-17]. The model and solver that were developed for one problem are usually fitted for the problem only.

The issue of unifying the ETPs has not been widely discussed. There is still no attempt devoted to deeply analyzing the ETP problems from a broader perspective and revealing the core elements of any ETP. Therefore, the main purpose of this paper is to present a framework based on the authors' experience and analysis that at its core any ETP contains the same basic elements, which are usually hiding behind the various input formats. The representation of an ETP in its basic elements is called the stemmed form. The benefit of this representation is that an approach that is deployed to solve one ETP can be reused to solve another type of ETP.

Despite the fact that there has been an enormous number of publications, many ETPs are still unsolved to optimality. Therefore, the second objective of the paper, as part of the framework, is to propose a general approach to solve the problem. The solution strategies are developed on top of the output of stemming process. We present strategies to solve two types of solutions – feasible and optimal, using some heuristics. In feasible solution finding, a Constructive Heuristic (CH) is used taking the advantages provided by the stemmed form of the problem. Meanwhile, in optimal solution finding, we deploy enhanced Simulated Annealing (SA)-based approach, which we call Explorative and Exploitative Simulated Annealing (E2SA). In this approach, a generic SA algorithm is equipped with three different neighborhood structures which will improve the exploratory and exploitative capability of the algorithm. Implementing the proposed framework, we manage to solve many kinds of ETP, real and fabricated, giving very competitive solutions compared to the state-of-the-art results.

The paper is organized as follows. Section 2 reviews the ETPs in more detail and presents some related works. Section 3 presents the methodology and the steps of the research. In Section 4 the proposed framework is presented in detail, and in Section

5 computational results of the proposed approach on some well-known ETP instances available in the literature are reported. Finally, Section 6 concludes the paper.

2. **ETPs and the Related Works.** Any educational institution has an obligation to manage all academic activities efficiently and effectively. The involvement of all stakeholders is key to achieve the institutional mission [18]. ETP is the problem of assigning a set of objects called events to a set of other objects called resources in such a way that no constraint is violated or kept to a minimum. This task is a part of the institutional tasks to ensure that all the activities can be realized in the best convenient and effective way for all parties.

Besides CTTP, UCTP, and ExTP, from a different perspective, ETP can also be categorized as room-dependent and room-independent. In room-dependent timetabling, an event requires a specific room to run [11,19,20]. In the case of room-independent timetabling, there is no issue related to room availability [14,15,21]. This variant includes online class timetable, class-teacher timetable, and examination timetable. It is also known as the graph coloring problem of timetabling, as the problem can be reformulated as graph coloring problem [21] where nodes represent events, edges between two nodes represent a relation between them. Color represents timeslot. One can color as many nodes as possible with the same color provided that there is no edge between them. The requirement that two adjacent nodes should be assigned to different colors can be analogously applied to a timetabling problem, where two conflicting events cannot be assigned to the same timeslot.

Constraints play a crucial role in ETPs. If there is no constraint, then there will be no ETP. Any additional constraint posed to the problem will increase the complexity of the problem [22,23]. Modeling an ETP is very much related to model the constraints. We recognize two types of constraints which are hard and soft.

1) *Hard Constraints* – are those requirements that must be satisfied. Any violation of this type of constraint is not acceptable. In this paper, we introduce two types of hard constraints which are Static Hard Constraint (SHC) and Dynamic Hard Constraint (DHC). SHCs are requirements that must be fulfilled by a specific event. This constraint has no impact on other events. Meanwhile, DHCs are constraints that may impact two or more events. SHC may include
   - SHC1: An event should be assigned to one of the given resources that provide all the required features.
   - SHC2: Instructor or student availability.
   - SHC3: Time and room availability.

   Examples of DHC can be
   - DHC1: Competition related constraints. Two events are in competition relation if their domains share one or more resources. If an event is using a specific resource, then the same resource must be made unavailable for its competitors. This prevents a resource from being double-booked.
   - DHC2: Distinct timeslot constraint. Two or more events should be scheduled at different timeslots due to student conflicts or instructor conflicts.
   - DHC3: Distinct day constraint. Two or more events should be assigned to different days.
   - DHC4: Parallel constraint. Two or more events should be scheduled at the same timeslot.
   - DHC5: Sequential constraint. An event should be scheduled before/after another event.

2) *Soft Constraints (SCs)* – are those constraints that are preferably fulfilled. These constraints can be violated, but the number of violations should be kept as low as possible. Examples of this type of constraints may include
   - (SC1) A student should not be assigned to the last timeslot of the day.
   - (SC2) A student should not have more than two events in a row.
   - (SC3) A student should not have only one event in a day.
   - (SC4) Conflicting events should be spaced out as evenly as possible.

Based on these two types of constraints, we can divide ETPs also into two categories. They are

1) *Feasibility problem* – the problem of finding a solution with no hard constraint violation, that is a feasible solution.
2) *Optimization problem* – the problem of finding a feasible solution with the minimum number of soft constraint violations.

There have been many ways to model an ETP found in the literature. In the case of UCTP, Babaei et al. [19] proposed a mathematical based formulation for the problem. Bettinelli et al. [11] discussed several models proposed for the curriculum-based timetabling problem that was used in the second International Timetabling Competition. The first one was proposed by Burke et al. [24], and the latter was proposed by Lach and Lübbecke [20]. The integer linear programming approach was used to describe the problem.

Many other researchers choose to model the ETP in the form of a mathematical model including IP and MIP [12,25,26]. In general, the problem is subsequently passed to some special tool to find the solution.

Mostly, the above-mentioned formulations are created tightly fitted to the problem in consideration. The models are only used to describe the specific problem precisely. They cannot be applied to a distinct type of ETP. The generalization level is very low. Besides, the formulation brings no ease for solution-seeking. To solve the problem represented in this way, some specific tool such as C-plex is used. For a moderate size problem, it may take days to solve.

In the effort to solve the ETPs, there has been an enormous number of publications, ranging from deterministic approaches to stochastic approaches. We review some methods devoted to solving the post-enrollment course timetabling problems, in particular the ones posted by Socha et al. [27] and the Post Enrollment Track of the 2007 International Timetabling Competition [28] (ITC-2007 PE). These two datasets will be used as the main references to benchmark the performance of our approach.

Solvers on Socha's Dataset. Obit et al. [17] used PSO based algorithm to solve the problem. Aziz et al. [10] utilized a variable neighborhood search in combination with an adaptive guided approach to choose the neighborhood. A hybrid swarm-based approach was used by Fong et al. [29]. In 2014 Fong et al. [30] presented another approach based on hybridization of some heuristics, on the same dataset. Abdullah et al. [31] used another hybrid metaheuristic combining electromagnetic-like mechanism and the great deluge algorithm. Ceschia et al. [32] used an SA-based algorithm with various parameter settings. Abdullah et al. [33] used a randomized improvement algorithm with multi neighborhood structures. In Rezaeipanah et al.'s paper [34], a hybrid approach between Genetic Algorithm (GA) and Local Search (LS) is presented. This approach is named (IPGALS) standing for "Improved Parallel Genetic Algorithm and Local Search" where LS is used to strengthen the Genetic Algorithm (GA). Finally, the first results on the dataset were published by Socha et al. [27]. They utilized random restart local search and MAX-MIN ant system.

Solvers on ITC-2007 PE Dataset. Ceschia et al. [32] used a metaheuristic approach based on SA. The result is that a carefully engineered and tuned SA-based solver performs exceptionally well on multiple instances. Lewis and Thompson [7] carried out a mathematical analysis on the PE UCTP problem and presented a metaheuristic-based two-stage algorithm to solve it.

In 2017, Goh et al. [35] combined various LS algorithms in a two-stage procedure. In the first stage, Tabu Search which is equipped with Sampling and Perturbation (TSSP) is used to produce a feasible solution. In the second stage, the authors propose an SA repair variant, called Simulated Annealing with Reheating (SAR), to improve the quality of the solution. In 2019 the same group of authors [36] used a two-stage problem-solving approach; namely finding a feasible solution in the first stage, followed by optimization of SC in the second stage. The researchers used a variant from SA which they called Simulated Annealing with Improved Reheating and Learning (SAIRL). Finally, Rezaeipanah et al. [34], used the same hybrid approach as described previously to solve the ITC-2007 PE instances.

3. **The Method.** Figure 1 depicts the procedures used in this research. Drew on a series of literature reviews, we found that most of the papers in ETP address a very specific problem. The technique developed for such a problem, cannot be applied to another problem. This is the main problem that we are addressing in this paper. Meanwhile, based on some observations and experiences in solving many types of ETPs we come up with a hypothesis that of the many variants of ETP, they actually have the same core containing three elements: set of events, resources, and constraints.

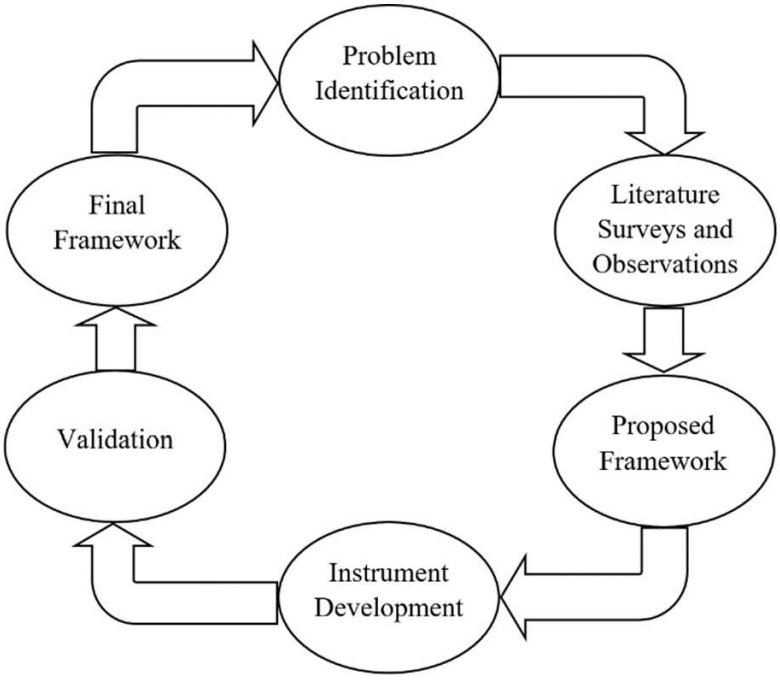


FIGURE 1. Research procedures

Furthermore, the author goes on to develop some computer programs which will be used as an instrument to prove the hypothesis. A cycle of problem identification, coding, testing, and evaluation are carried out to develop the instrument. The instrument is applied to many types of ETPs through a series of computational experiments. At the end of the process, we propose a framework that leads to a more effective way of handling

and solving the problem. A conclusion is drawn from this work that in most cases an ETP consists of the same recurring elements: events, resources, and constraints. The main task to be performed in an ETP is to assign resources to events taking consideration of the constraints posed to them. The set of constraints is the main element that distinguishes one ETP from another. The benefit of the framework becomes evidence where a solver designed for one ETP can be applied to others with a minimum adjustment.

4. **The Proposed Framework.** In Figure 2, we depict the proposed framework, where the ETP construction process is considered as a 4 layers mechanism. It started with the uppermost layer where a timetable requirement is created, triggered by student enrollment and/or instructor assignment in any courses. Layer 2 concerns with input reading, followed by input handling in Layer 3, whereby the detailed implementation formats are created. The output of Layer 3 is the instance's presentation in a unique form which will be called the stemmed form. Our main objective of the paper is to show that at the end, an ETP can be transformed into this form. Finally, in the last layer, the timetabling construction process is performed. Layer 1 to Layer 3 is called the Stemming Phase, and the last layer is the Solution Finding Phase. In Subsection 4.4 we present a solution finding approach that may be applied to any ETP instance once it has been transformed into the stemmed form.

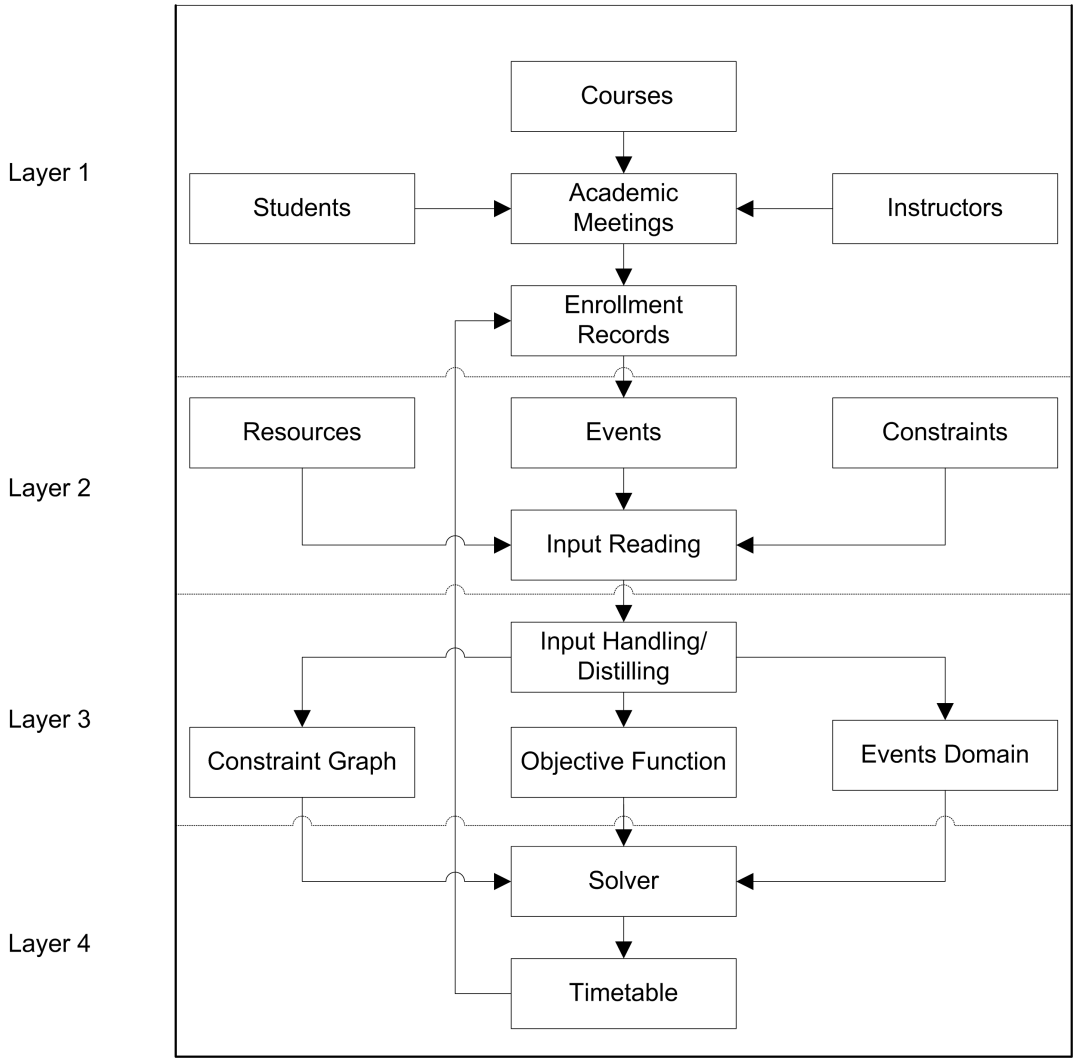


FIGURE 2. The high-level view of the proposed framework composed of four layers

4.1. **Layer 1: Events creation.** Every educational institution has several sets of curricula, containing courses or subjects to be offered to participants or students. A course may induce one or more academic transactions whereas an academic transaction belongs to one and only one course. It is these transactions that compose the set of main elements of an ETP. This set may include course sessions, tutorials, lab activities, and exams. Apart from exams, all academic transactions related to a course are assumed to be repeated, say on a weekly basis. In this case, the timetable is constructed for one week only. In the case of exam timetabling, there is usually no repetition; however, its construction process is identical to any other ETPs with the possibly required a longer time period. An academic transaction, together with its assigned instructor(s) and student(s) that need to be scheduled regularly, will be called an event. A unique event should then be identified from those three elements: an academic transaction, a set of instructors, and a set of students. Hereafter the symbol V will be used to denote the set of events.

4.2. **Layer 2: Input reading (resource and constraint).** This layer concerns how an ETP's instance is read into the application. We divide the inputs into three main elements: events, resources, and constraints.

4.2.1. *Events reading.* Most of the ETP instances found in the literature describe the set of events just as scalars representing the number of events. The event's attributes are usually presented as scalars or matrices. In many cases, the set of events is not presented directly. These sets have to be extracted from other elements. For example, in CB-CT, the set of events has to be derived from a set of courses.

4.2.2. *Resources reading.* An ETP must contain the set of resources composed of time and places set. This set may be presented in many different forms. The set of resources must be defined beforehand. The format is in general problem-dependent. In some problems, it may contain the definition of days, time slot, period, rooms, and all the information related to it. There is no single way to express these elements.

4.2.3. *Constraints reading.* Basically, any constraint restricts one or more events to be assigned to some specific resources. Constraint expression is also problem-dependent. It may be expressed directly and indirectly in many distinct ways. For example, the requirement that two events must not be scheduled in the same timeslots may be presented indirectly in the form of student enrolment. If two events are enrolled by the same student, then they must be scheduled in two different timeslots.

Another example of representing constraint is as follows: each event is commonly associated with features representing requirements for the specific room and/or timeslot to carry out the event effectively. An event may require a large room to accommodate the number of students enrolled or an event may require specific teaching support equipment in the room (data projector, computers, etc.). On the other hand, each resource will have its own characteristics or features. All inputs related to the feature requirements and feature provisions are assumed to be available prior to the timetabling construction.

Students' enrollment and lecturers' or instructors' assignments play an important role on the formation of the constraints. Many constraints are related to them. Curriculum, the institutional policy, as well as resources, also contribute to the formation of constraints. Timetabling becomes the task of assigning all events to resources such that the features required by an event can be satisfied by the chosen resource. A good timetable must or should have every event gets assigned to its best suitable resource.

4.3. **Layer 3: Input handling and refining.** This layer performs a series of transformations to extract some ingredients from the inputs to identify, classify and store the room and time information as well as any object related to constraints in the stemmed form representation.

4.3.1. *Set of events and students.* Identifying and defining the set of events is a very crucial first step. In most cases, there is no need to extract the set of events from the input, as it has been presented explicitly. In some cases, some procedures are carried out to extract the events.

Students and instructors are crucial in the timetabling process, especially when dealing with constraints. Together with events, this set becomes one of the main sources of constraints. This set may be presented explicitly or implicitly.

We introduce a new term called GOSP (Group of Similar Participants) to represent a group of students or instructors who participate in the same set of events.

The enrollment matrix (STUDENT-EVENT matrix) and the teacher assignment matrix are the sources of information to construct the GOSP. In this approach, we combine these two matrices into a new matrix called the PARTICIPANT-EVENT matrix.

To identify a GOSP, for any participant, collect the set of events taken by that participant, collect all participants taking the same events, and put them in a group as that GOSP. Notice that not all subsets of V can be a GOSP. This way, any participants will be mapped to a single GOSP. A GOSP can then be considered as a compact form for representing participants of an event. It is a mutually independent partition of the set of participants. A GOSP will be associated with weights or size which represents the number of participants. The creation of GOSP will shrink the size of the instance without changing the SHC. The benefit of using this representation is that it reduces the computational resources when it comes to solution evaluation task, and thus reduces the computational resources for the whole process. In the whole timetabling process it is more beneficial to utilize the GOSP-EVENTS matrix rather than the STUDENT-EVENT matrix.

4.3.2. *Set of resources.* In general, we assume that any ETP includes a set of timeslots or periods and rooms. In the case of a room-independent timetable, we create a virtual room, where all events or exams can use it at any time. Providing that no conflict occurs, any number of events or exams can be put in that room. However, in the case of the examination timetable, the (virtual) room capacity may be posed to the problem. This is to prevent examinees outnumber the (virtual) room's capacity.

In the following, we introduce an approach for handling the set of rooms and timeslots. Let the set of timeslots be denoted by W and the set of rooms be denoted by R. Let $B = R \times W$ be a Cartesian product between the set of rooms and the weekly timeslots. The members of this set are labelled by a number from 0 to $|R \times W|$. By a simple calculation, it is easy to find which room and timeslot a resource $b \in B$ refers to. To illustrate the idea, suppose that $W = \{0, 1, 2, \ldots, 44\}$ and $R = \{0, 1, \ldots, 9\}$ then $B = \{0, 1, \ldots, 449\}$. Let $b = 17$, and this number refers to resource 17 which is Room 0, and Timeslot 17.

4.3.3. *Constraints handling.* In this step, we describe the way in modeling the constraints. It is important to notice that there is no unique way to represent the constraints in the input file. Distinct timetable problems may have a very specific way to express the constraints. Sometimes extra efforts are needed to recognize and extract the constraints from their various input formats.

a) ***Hard Constraints Handling.*** Basically, any hard constraint posed to an event restricts the resources that could be used by that event. As pointed out in Section 2, there are two types of hard constraint, the first one called static hard constraint; defined as

those that posed to events independently and cannot be changed during the timetabling construction process, the second one called DHC defined as constraints that affect the available resources of an event depending on the assignment made to other related events. SHC restrict permanently the domain of some events by deleting some resources from its domain from the beginning and during the assignment process. The deletion of one or more values from the domain of an event does not depend on the assignment of other events. DHC temporarily turns on or off some values from the domain set of an event whenever one or more event(s) that have relation to that event have been assigned to a new resource.

i) ***SHC Handling.*** Any static hard constraint posed to an event will reduce the number of resources that could be used by that event. Therefore, we create a set D_e which contains all valid resources that could be used by the event e without violating any of the SHC. This set is referred to as *domain set* of event e. Any additional static hard constraint posed to an event will reduce the cardinality of D_e. For each event e, there is a specific set $D_e \subseteq B$, which contains all candidate resources that could be used by the event e without breaking any static constraint. In general, we also have $D_e \neq D_j$ for any two events e and j. Let $D = \{D_1, D_2, \ldots, D_{|V|}\}$. Hereafter D will be called the domain set of the instance.

ii) ***Dynamic Hard Constraints (DHC).*** To include this type of HC in the timetabling process, the so-called *Constraints Graph* is proposed in this paper. It is a weighted graph and incorporates both directed and undirected edges if necessary.

Let $V = \{v_1, v_2, \ldots, v_n\}$ be the set of events that have to be scheduled on a weekly basis. Let $G = (V, E)$ be a graph whose vertices are the set of V, and $(v_i, v_j) \in E(G)$ if and only if there is a relation between events v_i and v_j. The edges are weighed to indicate the type of relationship that the two corresponding events have. Suppose $(v_i, v_j) \in E(G)$ and let $w(v_i, v_j)$ represent the weight of the edge between v_i and v_j. As an example, $w(v_i, v_j) = 1$, if events v_i and v_j have to be assigned into distinct timeslots (DHC 2d).

The constraint graph is implemented as a two-dimensional array and will be used to control the assignment process. It acts as a source for consultation, when an event is being assigned or updated, supplying the information on whether a change to the solution can be carried out without breaking any hard constraints. This graph will play an important role to monitor the hard constraint violations during and after the assignment process. Any new event assignment has to be followed by a domain set adjustment.

b) ***SC Handling***. The objective function is used to quantify the quality of a solution. In timetabling problems, it usually measures the occurrence of SC violations. As there are many constraints involved, an objective function is usually composed of a weighted sum of some components, where the weight of each component reflects the importance of the corresponding constraint.

4.3.4. *Stemmed form of ETP instance.* The discussion from the beginning of this subsection attests that it seems unnecessary to make any distinction between ETPs, based solely on the type of institution it is dedicated to. At this point, an ETP is actually an assignment problem involving three components V, B, and C where V is a finite set of events, B is a finite set of resources and C is a set of constraints containing SHC, DHC, and SC. The constraints are represented in the following forms:

i) Domain set D, where $D = \{D_1, D_2, \ldots, D_{|V|}\}$, and D_i be set that contains all resources that could be used by event i. The resources are represented by integers where each integer refers to a specific room and timeslot.

ii) Constraint graph G of size $|V| \times |V|$, which is implemented as a two-dimensional array, where $G[i][j] = 1$, means that event i and event j are connected with weight 1,

which means they must be assigned to two distinct timeslots. Meanwhile, $G[i][j] = 0$, means that event i and event j have no connection, which means they can be assigned to any timeslots.

iii) GOSP-based objective function f.

Thus, an ETP can be viewed as a 4-tuple $< V, D, G, f >$. Finding a feasible solution can be mathematically formulated as finding an injection I from the vertices of a graph to their domain set. That is to find an $I : V \to B$ satisfying the following conditions:

- For each $v_i \in V,\ I(v_i) \in B_i$ (1)
- If $(v_i, v_j) \in Ed(G)$ and $q_{ij} = 1$ then $I(v_i) \bmod |W| \neq I(v_j) \bmod |W|$ (2)
- If $i, j \in Ed(G)$ and $q_{ij} = 4$ then $I(v_i) \bmod |W| < I(v_j) \bmod |W|$ (3)
- If $i, j \in Ed(G)$ and $q_{ij} = -4$ then $I(v_i) \bmod |W| > I(v_j) \bmod |W|$ (4)

Note that in this problem we assume that any kind of SHCs is represented by the set B and only DHC1, DHC2, and DHC3 are posed to the problem. Any additional SHC will change the set b_i of related event I and additional DHC will add more equation that is similar to that of Equations (2), (3), and (4).

If such an injection exists, then $I = \left\{I(v_0), I(v_1), \ldots, I\left(v_{|E|-1}\right)\right\}$ must be one of the feasible solutions to the problem for the following reasons:

1) I is an injection implying that all vertices must be mapped to their domain, meaning that all events must be assigned to their appropriate resource.
2) Likewise, as I is an injection, then there will be no two or more events to be assigned to the same resource. This guarantees that there will be no double-booked resources.
3) The condition in Equation (1) ensures that there is no violation of all SHC.
4) Equation (2) ensures there is no violation of the DHC that two conflicting events cannot be assigned to the same timeslot.
5) Equations (3) and (4) related to the order relation of two events that one event must be assigned to a timeslot before or after another event.

Note in the above formulation all SHC's are represented in the set B_i and the DHC's are represented by Equations (2), (3), (4). In this case, we assume that only two dynamic DHC's are imposed to the problem which are DHC1 and DHC2. Distinct ETP will induce distinct formulation. A solution is represented, as a one-dimensional array I. $I[2] = 3$, for example, means that event 2 is assigned to resource 3. Meanwhile $I[1] = -1$ denotes that event 1 is still unassigned.

The objective function is usually problem-dependent and model-dependent. It is worth to note that the utilization of GOSP can significantly reduce the computational burden for evaluating the quality of a solution. In the following equation, we present an example of an objective function. We assume that SC1, SC2, and SC3 are posed to the problem. Then the objective function is composed of the sum of all soft constraint violations over all GOSPs. That is, for a given solution I, the objective function f is defined as follows.

$$f(I) = \sum_{i=1}^{|H|} \sum_{j=1}^{|X|} \theta_{ij}(I) + \sigma_{ij}(I) + \rho_{ij}(I) \tag{5}$$

where

- H = the set of days,
- X = the set of GOSPs,
- $\theta_{ij}(I) = |GOSP_j|$, if $GOSP_j$ has to attend an event scheduled in the last timeslot of day i and $\theta_{ij}(I) = 0$ otherwise,

- $\sigma_{ij}(I) = \begin{cases} |GOSP_j|, & \text{if } GOSP_j \text{ has 3 consecutive events in day } i \\ 2|GOSP_j|, & \text{if } GOSP_j \text{ has 4 consecutive events in day } i \\ 3|GOSP_j|, & \text{if } GOSP_j \text{ has 5 consecutive events in day } i \\ 4|GOSP_j|, & \text{if } GOSP_j \text{ has 6 consecutive events in day } i \\ 5|GOSP_j|, & \text{if } GOSP_j \text{ has 7 consecutive events in day } i \\ 6|GOSP_j|, & \text{if } GOSP_j \text{ has 8 consecutive events in day } i \\ 7|GOSP_j|, & \text{if } GOSP_j \text{ has 9 consecutive events in day } i \end{cases}$
- $\rho_{ij}(I) = |GOSP_j|$, if $GOSP_j$ has only one event to attend in day i; $\rho_{ij}(I) = 0$ otherwise.

4.4. **Layer 4: Solution finder.** The outcome of the processes from Layer 1 up to Layer 3 mentioned before is an ETP instance that has been represented in stemmed form containing a set of events V, resource set B, domain set d, constraint graph G, and its corresponding objective function f. In the following, we will describe the approach for finding the solution (feasible and optimum) based on the stemmed form of the instance.

The timetabling solution finding process is carried out in four stages. The first two stages deal with the feasibility problems, i.e., the problems of finding a timetable with no hard constraints violations, and the last two stages deal with the optimization problems, i.e., the problem of minimizing the SC violations.

In this proposed method, the domain set and the constraint graph play as the inputs for the feasible solution finder and together with the feasible solution, become the inputs in the optimization process. We also introduce a new element called the adjusted domain set. Initially, the adjusted domain set is equal to the domain set. Any new event assignment will affect the domain set of an event and therefore it has to be adjusted.

The adjusted domain set of an event shows the remaining available resources that could be used by the event. By using this method, a feasible timetable can be constructed by assigning events one by one to its available resources. This is the essence of Stage 1 of the approach proposed in this paper and will be referred to as CH.

Stage 1. Constructive Heuristic (CH). CH is one of the graph coloring techniques that is not uncommon in finding a feasible solution in timetabling. It is the simplest way to construct a timetable by picking events one by one and a resource is assigned to it. CH selects events sequentially based on several criteria that reflect how difficult it is to schedule the event. Various techniques commonly used in CH are presented below. An event's degree states the number of events that cannot be scheduled in the same timeslot as that event.

- Largest Degree first (LD) – order the events descending according to their degree.
- Largest Weighted Degree first (LWD) – similar to LD, but the degree is weighed by the number of students involved in the conflict.
- Least Saturation Degree first (LSD) – the next event to be assigned is the one that has the minimum number of resources remaining.

Some of the sequential techniques mentioned above are quite effective in generating initial solutions [8,23,37,38]. It was also concluded that the ability of a technique to generate the initial solution will increase if it uses more than one CH. However, using one or more heuristic combinations does not guarantee that feasible solutions can be found. There is a situation that at some steps, no more resources left for an unsigned event. Therefore, we enhance our construction strategy with other techniques as described in [13] and [39] and will be briefly reviewed in the next discussion.

Stage 2. Relaxation Strategy. This stage is intended to handle the feasibility problem for some instances that Stage 1 fails to produce. The process is called a relaxation strategy and has been described in detail in [39].

This technique starts with running CH with an increasing number of periods starting with the given periods + 1, until a "feasible" solution is found. A "feasible" solution, in this case, is a solution with no hard constraint violations, possibly using some extra periods. Thus, it is infeasible in reality. This solution is then passed to the next step in order to eliminate the extra timeslot through an optimization process.

We develop SA based heuristic to minimize the number of events assigned to an extra timeslot. We use the number of students of the corresponding event as the objective function in our SA schema. The process is terminated if a global optimum solution is found or if the time limit allocated to this stage is reached. A solution with zero objective function value would be a real feasible solution.

Stage 3. Solution's Quality Improvement. This stage is aimed to optimize the SC violations that is to minimize the value of the objective function. An SA-based search technique is implemented in this stage.

Simulated Annealing (SA). SA is a stochastic search method based on the use of an LS. It was introduced by Kirkpatrick [40] in 1983, inspired by the annealing process of solids in physics. An analogy with this process, the SA method seeks to find a solution by moving from one solution to another. In any step, SA either moves to a better neighbor solution if it finds one, or to a worse solution with a certain probability. It includes some important parameters concerning the probability of accepting a deteriorating solution. If the parameters are tuned properly then Hajek [42] proved that the mechanism will find a global optimum solution with probability one.

A generic SA pseudocode for a problem P realizing the SA idea follows:

Pseudocode 1: A generic simulated annealing

1. **Input:** Incumbet solution I, *maxItSA*, *isoIt*
2. **For** $k = 1$ to *maxItSA*
3. Calculate t_k
4. **For** $j = 1$ to *isoIt*
5. Generate $\aleph(I)$ the neighborhood of I
6. Choose a solution S' from $\aleph(I)$
7. **If** $\Delta_{cost}\left(I, I^{'}\right) \leq 0$ then $I \leftarrow I'$
8. **Else**
9. **If** $rand[0,1] < \exp\left(-\frac{\Delta_{cost}(I,I')}{t_k}\right)$ then $I \leftarrow I'$
10. **Endfor**
11. **Endfor**

Note: $\Delta_{cost}\left(I, I^{'}\right) = g\left(I^{'}\right) - g(I)$

There are some parameters to be set, *maxItSA* called outer loop, *isoIt* called inner loop, and t_k the temperature for each outer loop. The *maxItSA* depends on the resources (running time) availability, and *isoIt* is set such that the particles reach their equilibrium state in particular temperature t_k.

The higher the temperature, the higher the probability of accepting a worse solution. As the search progresses, the probability of accepting a worse solution decreases. A cooling schedule in SA is a general term used to manage the temperature during the SA process.

It includes the choice of the initial temperature, the rate of decrease of the temperature and the number of trials at one temperature level. The whole SA process is proven to be very sensitive to the choice of cooling schedule. Despite the fact that many authors have investigated this aspect, it turned out that none of their recommendations was suitable for our problem at hand. We then had to carry out some preliminary tests to tune in our cooling schedule:

1) Initial temperature: The initial temperature is chosen such that the probability of accepting a worse solution is sufficiently high (±40%);
2) Cooling equation: We tested many cooling equations and found out that the best one is similar to the one used by Kostuch [41], i.e., $T = 1/((1/T) + \beta)$ where β is chosen between 0.001 and 0.0005;
3) Number of trials: The number of trials in each temperature level is set to $a.|V|$ where a is linearly increased. Initially, a is set to 10.

In order to save CPU time, some problem-specific knowledge is incorporated into the search process. For instance, the cost calculation is done using delta evaluation (Δ_{cost}). Therefore, given a new solution, its cost is not calculated from scratch. Instead, it incorporates some unchanged cost components from its predecessor. In addition, moving an event using a simple neighborhood from an artificial timeslot to another artificial timeslot will not change the cost. This also applies to swapping two events in the same artificial timeslot and to the Kempe chain neighborhood.

Note that any SA-based algorithm requires the definition of the neighborhood structure. A neighborhood structure defines a set of solutions that the algorithm can choose from to move to its next state. We use three types of neighborhood structures as described next.

Simple, Swap and Kempe Chain Neighborhood Structures. A Simple neighborhood structure contains solutions that can be obtained by simply changing the resource of one randomly chosen event. Since the domain set stores valid unused resource(s) for each event, by employing this set the neighborhood structure can be easily implemented. However, this neighborhood structure involves some bias as there might be events that do not have any valid resources left at one stage of the search. This might create a disconnected search space. Henceforth, this neighborhood structure will be expressed by N_1.

A Swap neighborhood is created when the resources of two events are exchanged. A simple checking procedure has to be carried out to make sure the new solution will be from the feasible area. This has to be done in particular to deal with the time and room related constraints. The neighborhood structure overcomes the disconnection of the search space that might occur in the simple neighborhood. Henceforward, this neighborhood structure will be expressed by N_2.

Kempe chain neighborhood structure operates over two selected timeslots. It swaps the timeslot of a subset of events in such a way that the feasibility is maintained. In the present paper, we extend this type of neighborhood structure to increase the exploration ability of the move. The modified one is called K-Pairwise Kempe Chain, where K denotes the number of timeslots pair involved in a move. When $K = 1$ then there will be only 1 pair of timeslots involved, and this is exactly the original Kempe chain as mentioned above. Henceforth, this neighborhood structure will be expressed by N_3.

Based on our experience and literature review, implementing this simple SA is less likely to end up with a good quality solution f. Using a constant *isoIt* has a drawback that in the low energy level state, the number of solutions with that level or below becomes scarce. Therefore, at this level, an increased number of iterations is needed. The use of a single neighborhood also lessens the ability of the algorithm to explore the solution space. Therefore, the following algorithm which we call E2SA is proposed.

In Pseudocode 2 we use three types of neighborhood structure to enhance the exploration and exploitation ability of the algorithm. The Simple (N_1) and Swap (N_2) neighborhood structures are used to exploit the current search area, as they have little impact on the current solution. Meanwhile, the Kempe chain neighborhood structure is used to strengthen the exploration ability of the algorithm as it has the ability to move to another solution that is significantly different from the current one.

Pseudocode 2: The pseudocode of Explorative and Exploitative Simulated Annealing (E2SA)

```
1. Input: Initial solution L
2. Initialize: maxItMNSA, T, #InnerIter, matrix D, time_limit, #noimpr
3. For i = 0 to maxItMNSA
4.    For j = 0 to #InnerIter
5.       Choose an event randomly for N1 move
6.       If spare resource is available
7.          Choose one randomly
8.          Calculate delta_cost
9.          If acceptable
10.            Make the move
11.            Update matrix D
12.            Update best solution L
13.         Endif
14.      Else
15.         Choose an exam for N2 move
16.         Calculate delta_cost
17.         If acceptable
18.            Make the move
19.            Update matrix D
20.            Update best solution L
21.         Endif
22.      Endif
23.   Endfor
24.   If (no better L is found in #noimpr iteration and time_limit is not exceeded)
25.      Perform N3 move
26.      Recalculate T
27.   Endif
28. Endfor
```

In addition to this feature, we also employ the idea of increasing the number of inner iterations as the search process progressing. This technique is used to give more chances for the algorithm to find a new better solution in the low-cost area. Finally, for some hard instances, we rerun Pseudocode 2 on the best solution found with a very low constant temperature to intensively exploit the area.

5. Validation: Experiments and Results Using the Proposed Framework.

This section aims to demonstrate how the proposed framework is deployed to various ETPs: from the Stemming Phase to the Solution Finding Phase. First, we present some metadata of the datasets; second, to justify the effectiveness of the proposed framework, we show that in all instances the stemming process can be done effectively and efficiently. We demonstrate diagrammatically the steps to transform the inputs into the stemmed forms and report the time consumption to accomplish the steps. In the third part, we present the effectiveness of the solver in finding feasible and optimal solutions using the stemmed form and the E2SA.

5.1. The datasets.

Five datasets are used in the experiments to validate the framework. In Table 1 we present the dataset names, number of instances, the constraints, and the links to the dataset.

TABLE 1. Dataset name, number of instances, constraints and link

Dataset name	Dataset type	# of instances	Hard constraints	Soft constraints	Link
Socha	UCTP	11	SHC1, SHC2, SHC3, DHC1, DHC2.	SC1, SC2, SC3	http://iridia.ulb.ac.be/supp/IridiaSupp2002-001/
STKIP	UCTP	5	SHC1, SHC2, SHC3, DHC1, DHC2, DHC3, DHC5.	SC1, SC2, SC3	NA
Lewis	UCTP	60	SHC1, SHC2, SHC3, DHC1, DHC2.	NA	www.dcs.napier.ac.uk/benp/centre/timetabling/harderinstances
Carter	ExTP and CTTP	13	SHC2, SHC3, DHC2.	SC4	http://www.cs.nott.ac.uk/∼pszrq/data.htm
ITC-2007 PE	UCTP	24	SHC1, SHC2, SHC3, DHC1, DHC2.	SC1, SC2, SC3	www.cs.qub.ac.uk/itc2007

Furthermore, some details regarding each instance of the dataset are presented in Table 2. The datasets are categorized as small, medium, and big based on some parameters which include the number of events, number of students, number of rooms, and number of timeslots.

5.2. **Stemming process.** In Figure 3 to Figure 5 we present diagrammatically the flow of the stemming process for each dataset. As shown in the diagrams, every dataset has different input formats. Therefore, to conduct the stemming process we create a method for each instance. Thus, the methods are problem-dependent. The methods are briefly described next.

5.2.1. *Socha's dataset.* Part of Socha's dataset inputs are packed in .tim file. The file contains the number of events and rooms, the number of students, and the number of features. It also includes the information on which events are taken by each student (stored as Student-Event matrix), which features are required by each event (Event Features matrix), and which features are provided by each room (Room Features matrix). The transformation process from the input files to the stemmed forms is depicted in Figure 3. The inputs read from the file are drawn in a solid line box, whereas inputs that have to be included directly in the source code (hard coding) are drawn in dashed line boxes.

Figure 3 shows the steps of transforming the inputs into stemmed form. The output of this process contains three elements which are event domain matrix D (and adjusted event domain), conflict matrix, and GOSP-based objective function f. The objective function in this dataset has been described in Equation (5). Note that there is no penalty if a student or GOSP has to attend two events that are scheduled in two consecutive timeslots in a day.

5.2.2. *STKIP dataset.* The instances are collected from a medium-size Indonesian college from the year 2014 to the year 2016. Instance's name indicates which academic year and semester it belongs to. The input file contains the event-duration matrix, event-lecturer matrix, event-student matrix, event-feature matrix, day conflict matrix, and lecturer-day instance. These matrices are the sources of HCs. There are altogether 5 instances. Apart from some additional constraints, each instance is presented almost exactly in the same

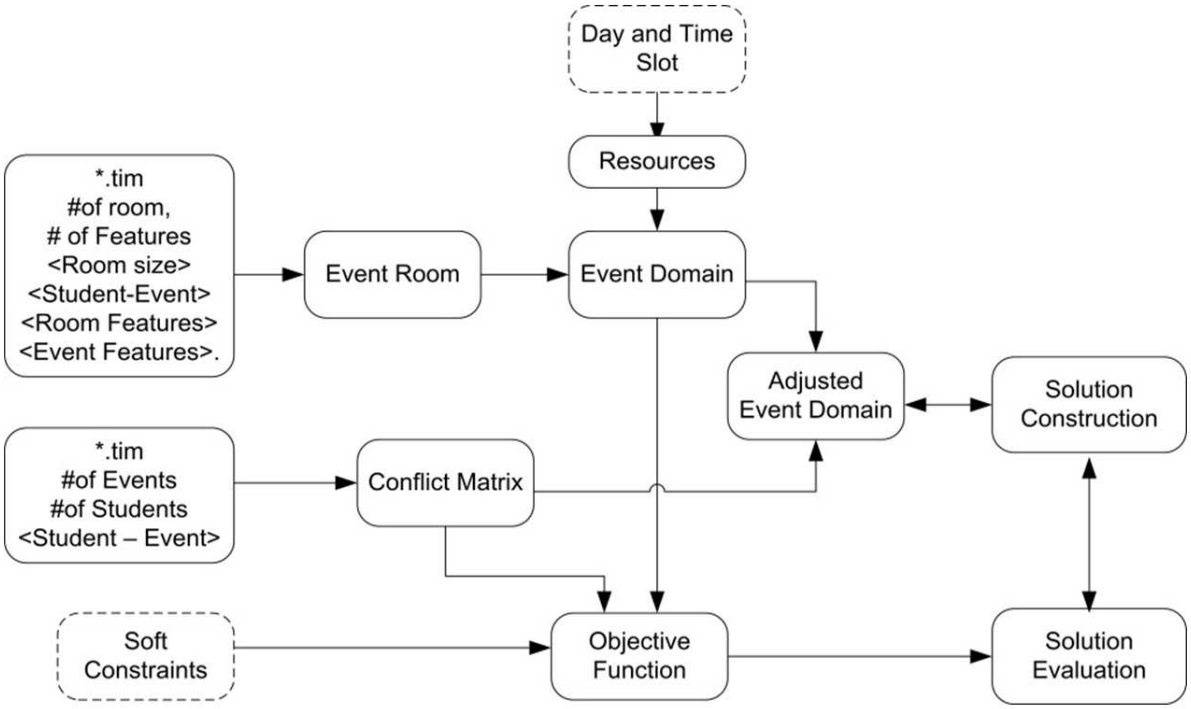


FIGURE 3. Transformation to stemmed form of Socha's dataset

way as in Socha's dataset. Therefore, the stemming process is identical to process in Figure 3.

5.2.3. *Lewis' dataset.* The dataset contains 60 instances and was created using an instance generator. The problem is identical to that of Socha's dataset. It contains the same hard constraints, nonetheless without soft constraint. Therefore, the goal of this ETP is just to find feasible solutions. The steps of transforming the inputs to the stemmed form are exactly similar to Socha's dataset as described in Figure 3.

5.2.4. *Carter's dataset.* The dataset comprises university and high school ExTP. It contains 13 instances and was introduced by Carter et al. [21] in 1996. The instances were taken from eight Canadian academic institutions; from the King Fahd University of Petroleum and Minerals in Dhahran; and from Purdue University, Indiana, USA. In this dataset, there are no room-related restrictions involved. Hence, as long as no conflict occurs, any number of events can be put into a timeslot. Unlike the previously mentioned datasets, the inputs of this ETP are packed into two files, .crs and .stu.

Because of this, it takes some extra steps to extract the set of events, resources, domain matrix, and constraint graph from the instances. The set of resources can be created by assuming there is an artificial room with infinite capacity. All available real rooms will be considered as one single artificial room. All events are permitted to be assigned into this room.

The number of timeslots varies from instance to instance and is part of the original requirements posed by the university. The domain matrix and the constraint graph can then be constructed exactly in the same way as in the UCTP problem. The only difference between the ExTP and the UCTP problems is that in this problem, a resource can be used by more than one event, i.e., a resource may be still available for an event although it has already been taken by others.

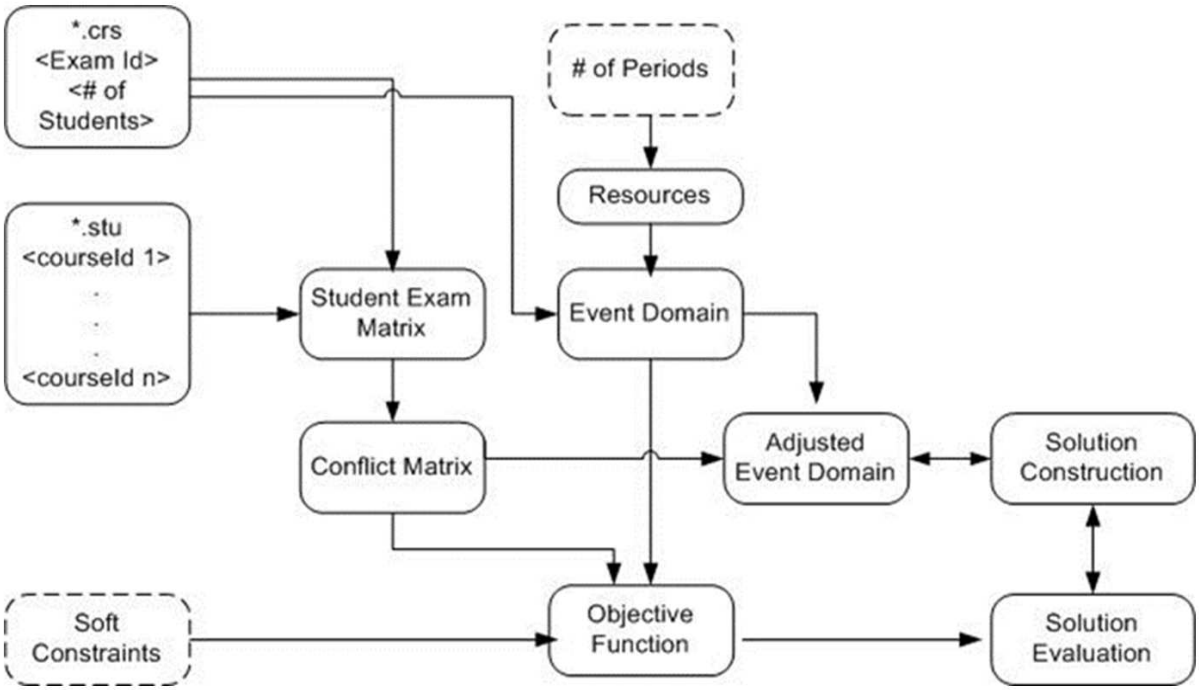


FIGURE 4. Transformation to stemmed form of Carter's dataset

There are two variants of problems posed on this dataset, which include

1) Find the minimum number of timeslots with which a feasible timetable can be constructed. This problem is better known as the graph coloring problem reduction of the ExTP (see [21]).
2) Given a fixed number of timeslots, find a feasible timetable where the conflicting exams are spaced out as evenly as possible. This problem is known as uncapacitated with cost problem (see [21]). 'Un-capacited' refers to the fact that there is no limitation in room capacity at any timeslot. The objective function for this problem is recognized as a proximity cost and was posed by Carter et al. [21].

5.2.5. *ITC-2007 PE dataset.* The problem is recognized as Track Two of the 2007 International Timetabling Competition. The problem allows students to enroll in any course and the timetabling is created based on the enrolment information. There are five types of hard constraints and three types of SC. Figure 5 shows the flow to extract the set of events, set of rooms, set of timeslots, set of constraints from the instances.

The SC posed to the problem is similar to the Socha's dataset, and therefore the same objective function is used to handle the SC.

5.3. **Stemming results and performance.** Figure 3 to Figure 5 show the process of transforming the raw input of each dataset to the stemmed form. To show the effectiveness of the process, we conduct an experiment for all instances and record the time consumption during the stemming process. The experiments are carried out on a PC Intel (Core) i3 3220 3.30 GHz running under Microsoft Windows 7 Professional. Each instance is run only once. In Table 2 we present the average running time under the "Time" column for each instance category measured in milliseconds (ms). Note that there is no category for Carter's dataset, as the dataset contains heterogeneous instances. The "Time Avail" column presents the typical running time set by researchers to solve the corresponding instance measured in seconds (s). There is no single benchmark time for each instance. In addition, there are many papers that do not report the run time explicitly.

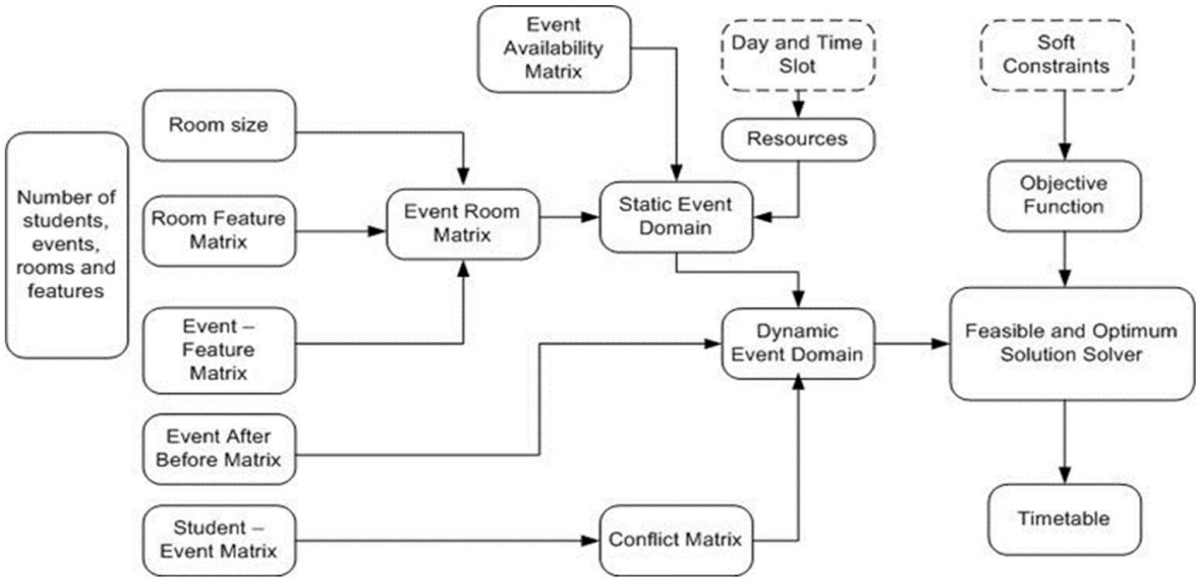


FIGURE 5. Transformation to stemmed form of ITC-2007 post enrollment dataset

TABLE 2. Time consumption of the stemming process

Dataset	**Category**	**# of instances**	**# of events**	**# of students**	**# of rooms**	**# of timeslots**	**Time (ms)**	**Time Avail (s)**
Socha_S	Small	5	100	80	5	45	57.4	120
Socha_M	Medium	5	400	200	10	45	408.2	600
Large	Large	1	400	400	10	45	750	1000
STKIP	Small	5	120	600	11-15	35	517.2	600
Lewis_S	Small	20	200-225	200-1000	5-6	45	1321.85	180
Lewis_M	Medium	20	390-425	400-1000	10-11	45	1805	900
Lewis_B	Big	20	1000-1075	800-1200	25-28	45	7194.2	1200
Carter								
Car91	Big	1	682	16925	N/A	35	30089	3600
Car92	Big	1	543	18419	N/A	32	24597	3600
Ear83	Small	1	190	1125	N/A	24	755	720
Hec92	Small	1	81	2823	N/A	18	745	420
Kfu93	Medium	1	461	5349	N/A	20	6366	3120
Lse91	Medium	1	381	2726	N/A	18	5758	2820
Pur93	Big	1	2419	30029	N/A	42	176354	111600
Rye92	Medium	1	482	11483	N/A	23	30414	3600
Sta83	Small	1	139	611	N/A	13	370	480
Tre92	Small	1	261	4360	N/A	23	2997	1080
Uta82	Big	1	622	21266	N/A	35	33324	3600
Ute82	Small	1	184	2750	N/A	10	1418	660
Yor83	Small	1	181	941	N/A	21	570	540
ITC-PE	Medium	24	200-600	300-2000	10-20	45	2303.45	240-600

In Figure 6 we show the proportion of the running time of each instance to the whole time that is dedicated to solving the instance.

In all instances, the method consumes very small proportions of resources that are dedicated to the whole process of the timetable construction. As we can see from the picture, the time consumption to carry out the stemming process is less than 1 percent.

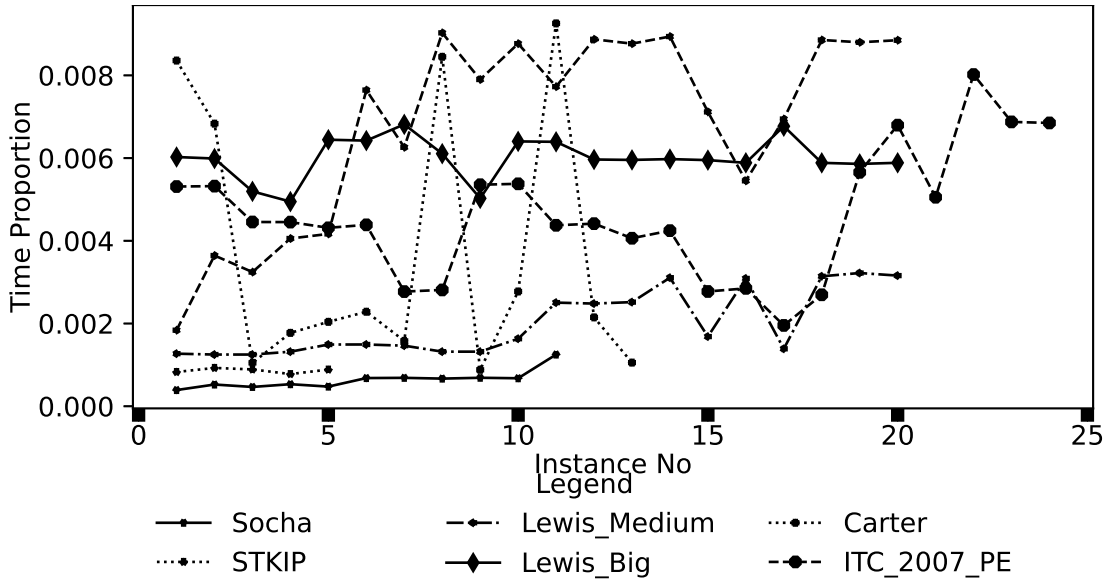


FIGURE 6. The proportion of the run time spent in the stemming process to the whole available time

This shows that any solver built on the stemming form will not substantially sacrifice the time during the solution-finding process. On the contrary, it will obtain many benefits as demonstrated later in the paper.

5.4. **Solution finding.** The four stages approach as described in Subsection 4.4 is deployed for all datasets. The least saturation degree CH is used to generate the initial feasible solutions. Whenever it fails to produce a feasible solution within the number of timeslots provided, we apply the relaxation strategy described in [39]. With the three neighborhood structures at hand, we tested several scenarios to minimize the soft constraint violations. From some preliminary experiments, we found that E2SA is the most promising scenario.

All computational tests were run on a PC Intel (Core) i3 3220 3.30 GHz running under Microsoft Windows 7 Professional.

5.4.1. *Socha's dataset.* Each instance is tested for 30 runs. In each run, the algorithm spent no longer than 120, 600, and 1000 seconds for small, medium, and large instances, respectively.

The CH was able to find feasible solutions for all instances, except for the large one. Thus, we applied the strategy described in [39] to deal with it. This strategy was able to quickly find a feasible solution. Then, we apply the E2SA heuristics, to finding the optimal solution.

The Solvers in Table 3 are named after the dataset used for the performance test. The following are the references to the corresponding solver.

- Socha1: Obit et al. (2017) [17]
- Socha2: Aziz et al. (2017) [10]
- Socha3: Fong et al. (2015) [29]
- Socha4: Fong et al. (2014) [30]
- Socha5: Abdullah et al. (2012) [31]
- Socha6: Ceschia et al. (2012) [32]
- Socha7: Abdullah et al. (2007) [33]
- Socha8: Socha et al. (2002) [27]
- Socha9: Rezaeipanah et al. (2020) [34]
- Us: E2SA (2020)

Table 3 clearly shows the effectiveness of the model and the solver. Our method (E2SA) can find the global optimum solutions for all small instances, and significantly improve the state-of-the-art results for the rests except Medium1 as an exception.

TABLE 3. Comparison of the results reported in the literature. Each cell shows the minimum cost and when available the average cost (in parenthesis). The values marked in bold indicate the best cost in the corresponding instance. The *inf* notation indicates that no feasible solution was found.

Instance name	Us (E2SA)	Socha1 (2017)	Socha2 (2017)	Socha3 (2015)	Socha4 (2014)	Socha5 (2012)	Socha6 (2012)	Socha7 (2007)	Socha8 (2002)	Socha9 (2020)
Small1	**0(0)**	**0**	**0**	**0**	**0**	**0**	**0**	**0(0)**	(1)	0
Small2	**0(0)**	**0**	**0**	**0**	**0**	**0**	**0**	**0(0)**	(3)	0
Small3	**0(0)**	**0**	**0**	**0**	**0**	**0**	**0**	**0(0)**	(1)	0
Small4	**0(0)**	**0**	**0**	**0**	**0**	**0**	**0**	**0(0)**	(1)	0
Small5	**0(0)**	**0**	**0**	**0**	**0**	**0**	**0**	**0(0)**	**(0)**	**0**
Medium1	14(30.6)	20	27(33)	57(70)	52(70)	175	**9(26)**	242(245)	(195)	84
Medium2	**12(23.1)**	19	41(48)	54(79)	45(79)	197	15(25)	161(162)	(184)	99
Medium3	**21(48)**	53	62(69)	114(132)	96(132)	216	36(49)	265(267)	(248)	142
Medium4	**5(24)**	11	30(38)	74(82)	52(82)	149	12(23)	181(183)	(164)	84
Medium5	**1(18)**	22	18(23)	64(75)	56(75)	190	3(10)	151(152)	(219)	112
Large	**139(280)**	460	399(405)	502(549)	461(503)	912	208(259)	*Inf*	(851)	516

Table 3 and Figure 7(a) and Figure 7(b) show the performance of E2SA as compared to the other methods. The best results are highlighted in bold font. The best and average results showed that the E2SA outperformed the other methods in all instances. The results can be attributed to the implementation of the framework which contains two core components: stemmed form and the E2SA solver.

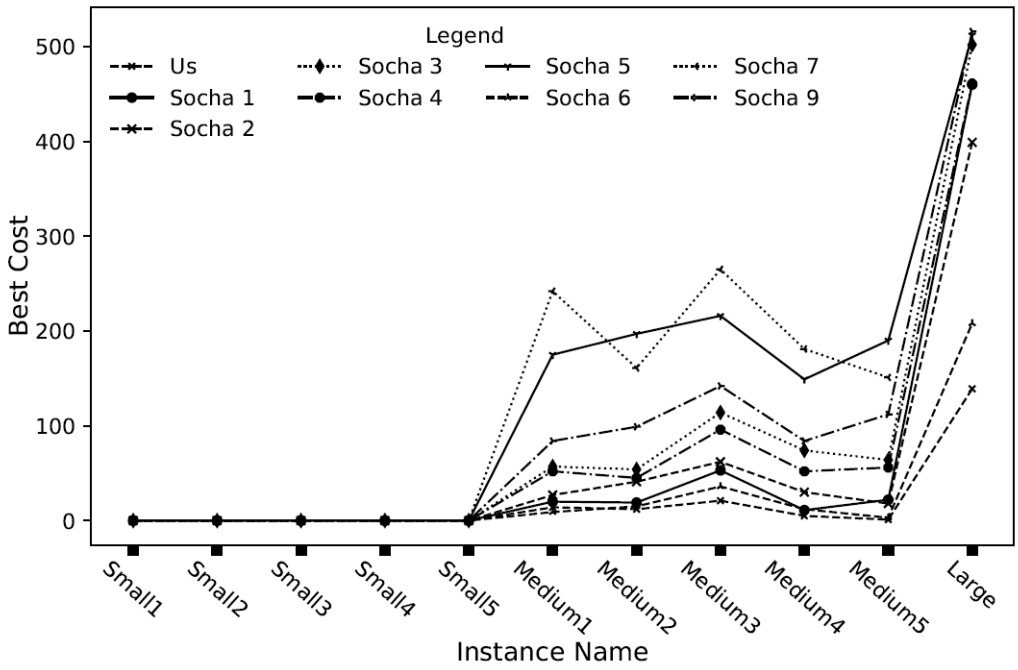
5.4.2. *The STKIP instances.* The process of creating the elements of the stemmed form is similar to that of Socha's dataset. However, an additional weighting schema for the constraint matrix is applied to accommodating the precedence relation and different day relation between events.

In all instances, the CH is applied to generating the feasible timetable, and the E2SA is used to minimize the SC violation. Instance information and the results are presented in the following table.

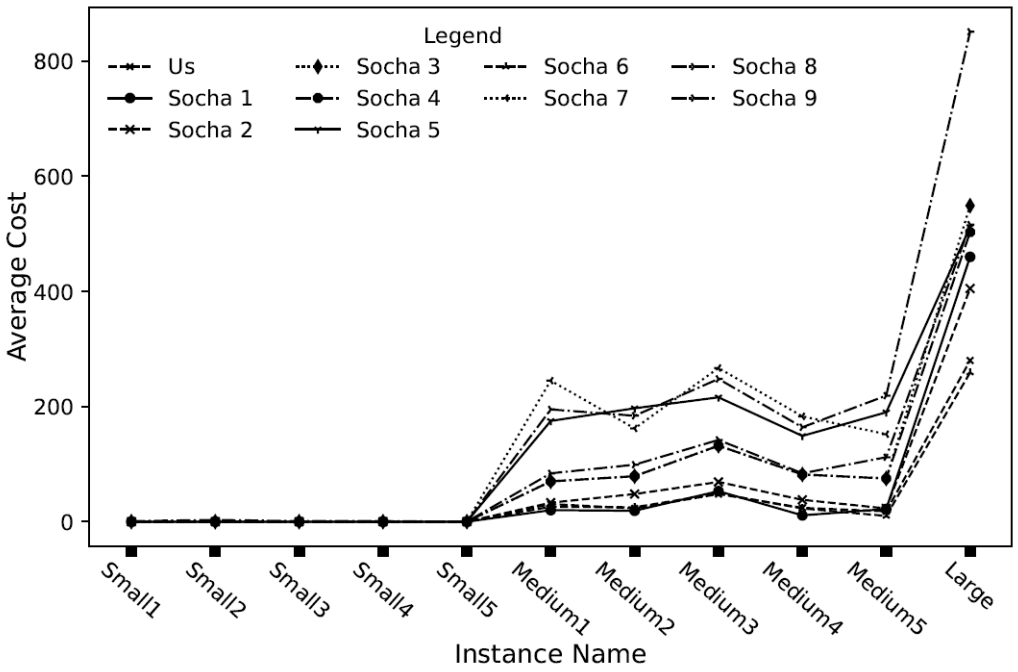
TABLE 4. The results of STKIP instances. The parenthesis in the feasible solution column indicates the minimum number of rooms required. The best OF column presents the best cost found through the 4 stages approach.

Instance name	# of events	# of students	# of rooms	# features	# of timeslot	Feasible solution	Best OF
S20141	183	632	19	2	25	0 (17)	92
S20142	170	631	13	2	25	0 (12)	124
S20151	170	637	15	4	25	0 (13)	134
S20152	169	545	13	2	25	0 (12)	125
S20161	118	391	13	2	25	0 (10)	123

Table 4 demonstrates the effectiveness of the framework. In all instances, the stemmed forms are easily extracted and based on this form the solver can be run smoothly. CH can find the feasible solution for all instances in a few milliseconds. The "Feasible solution" column shows that the CH bears another benefit to the management that it can reduce the number of rooms to a minimum level. However, the best solutions found by this solver are not the perfect ones. There are still some kinds of SC violations. We tried to run more than a hundred times for each instance and still no perfect solution was found. This may indicate that for all instances the global optimum solution simply does not exist. From the management perspective, this problem is not a big deal. Implementing



(a) The best (minimum) cost



(b) The average cost

FIGURE 7. Cost of 10 solvers in Socha's dataset

this best solution in practice brings some kinds of inconvenience to some students. For example, there may be classes that are scheduled in the last timeslot of the day.

5.4.3. *Lewis' dataset.* In [39] the authors present a relaxed strategy to solve the problem. It is shown that the proposed framework in combination with some heuristics can solve more instances of the problem. Recently in [13] we try to solve the problem using Stage 2 Ver 1 and found new feasible solutions for 2 instances.

5.4.4. *Carter's dataset.* The model and the implementation approach described in this paper are used to solve the problem. The results are very competitive, as it can find better solutions for many instances of the problem [14].

The results and further details on the implementation of the proposed approach in solving the problem can be found in [15].

5.4.5. *International Timetabling Competition 2007 course timetable datasets. Post enrolment university course timetabling problem.* The combination of the stemming form and the heuristics deployed in handling the problem, has made it possible to find the feasible solutions for all instances with minimum SC violation. The results using this approach along with results from other researchers are presented in Table 5.

TABLE 5. Comparison of the results found by our methods and the methods used by other authors

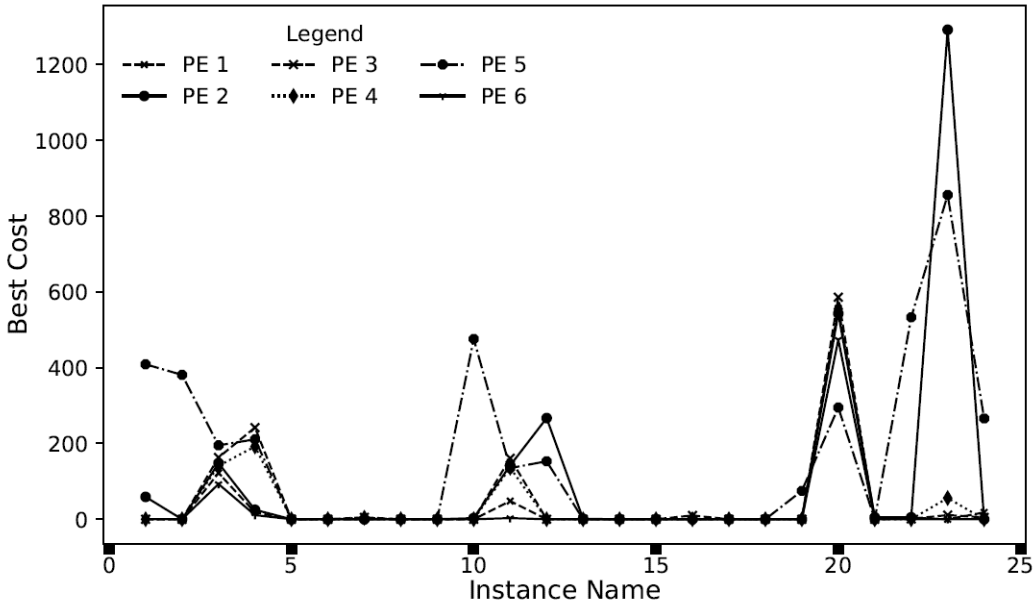
Instance name	**PE1 Min(Ave)**	**PE2 Min(ave)**	**PE3 Min(Ave)**	**PE4 Min(Ave)**	**PE5 Min(Ave)**	**PE6 (Us) Min(Ave)**
1	0(377)	59(3992)	0(307.6)	0(209.4)	409	0(356.4)
2	0(382.18)	0(142.2)	0(63.4)	0(10.01)	381	0(108.5)
3	122(181.76)	148(209.9)	163(199.4)	141(188.6)	195	93(172.2)
4	18(319.40)	25(349.6)	242(328.8)	192(320.9)	211	11(315.4)
5	0(7.52)	0(7.7)	0(2.7)	0(2.9)	0	0(3.2)
6	0(22.82)	0(8.6)	0(33.2)	0(54.7)	0	0(53.2)
7	0(5.45)	0(4.9)	5(18.0)	4(14.5)	0	0(2.5)
8	0(0.60)	0(1.5)	0(0.0)	0(1.6)	0	0(1.5)
9	0(514.37)	0(258.8)	0(100.7)	0(15.2)	0	0(387.3)
10	0(1202.41)	3(186.4)	0(65.3)	0(30.5)	476	0(125.4)
11	48(202.58)	142(269.5)	161(244.3)	136(201.6)	135	3(231.4)
12	0(340.22)	267(400)	0(318.2)	0(303.5)	153	0(328.7)
13	0(79.02)	1(120.0)	0(99.5)	0(90.4)	0	0(56.6)
14	0(0.53)	0(3.6)	0(0.2)	0(25.6)	0	0(9.2)
15	0(139.92)	0(48.0)	0(192.0)	0(12.5)	0	0(15.3)
16	0(105.16)	0(50.1)	10(105.8)	0(45.8)	0	0(23)
17	0(0.07)	0(0.0)	0(0.8)	0(0.5)	0	0(2.4)
18	0(2.16)	0(41.1)	0(12.5)	0(7.7)	0	0(6.7)
19	0(346.08)	0(951.5)	0(516.7)	0(11.0)	75	0(876.3)
20	557(724.54)	543(700.2)	586(650.7)	555(664.0)	295	474(657.8)
21	1(32.09)	5(35.9)	0(12.5)	0(25.7)	0	0(98.4)
22	4(1790.08)	5(19.9)	1(136.0)	0(5.8)	533	0(879.3)
23	0(514.13)	1292(1707.7)	11(504.4)	56(713.6)	856	0(786.2)
24	18(328.18)	0(105.3)	5(192.6)	0(77.5)	266	0(68.3)

The minimum (min) and/or the average (ave) cost found by the corresponding author(s) is shown. The references of the solvers follow:

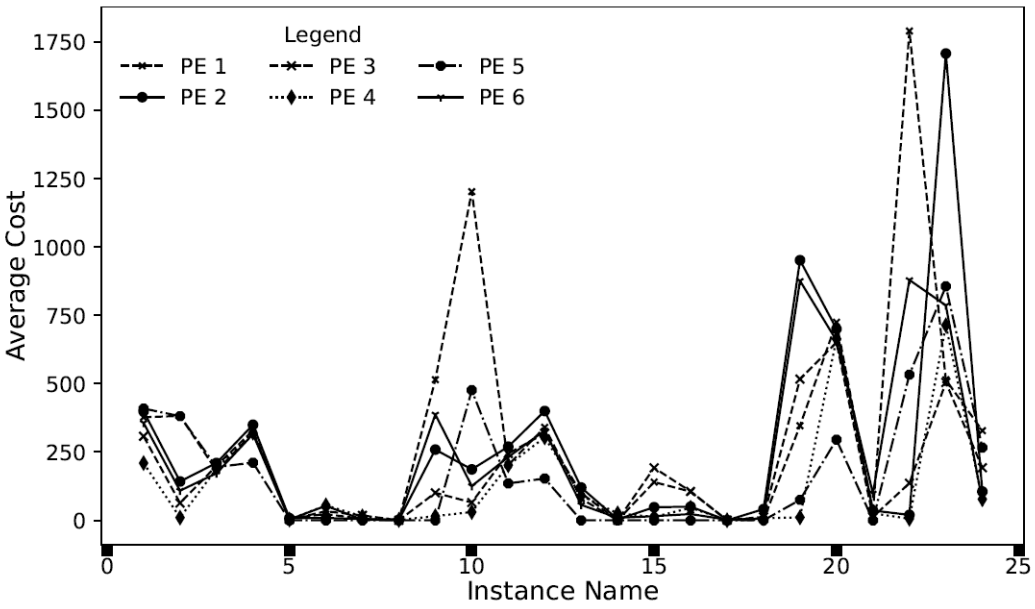
- PE1: Lewis and Thompson (2015) [7]
- PE2: Ceschia et al. (2012) [32]
- PE3: Goh et al. (2017) [35]
- PE4: Goh et al. (2019) [36]
- PE5: Rezaeipanah et al. (2020) [34]
- PE6 (Us): E2SA (2020)

The complete procedures of the framework are applied to this dataset. As shown in Table 2 and Figure 6, all instances are easily transformed into their stemmed form. The CH can find feasible solutions for all instances in a few milliseconds. Based on these feasible solutions the E2SA was applied to minimizing the SC violations. The robustness of this approach becomes evident as demonstrated by Table 5, and Figure 8(a). E2SA can find

global optimum solutions for 20 out of 24 instances. Moreover, it improves the solution's quality of three instances. Table 5 and Figure 8(b) demonstrate another dimension of the robustness of the algorithm. In many instances, E2SA shows a reliable performance as the average costs are close to the best costs of the corresponding instances. It is, however, there are big gaps between the best cost and the average cost for some instances. These gaps may be attributed to the role of the last stage of the algorithm that it will intensify the search in the low-cost region of the best solution for some instances. From a different perspective, however, the gaps demonstrate the capability of the algorithm to intensively exploit the promising region.



(a) The best (minimum) cost



(b) The average cost

FIGURE 8. The costs (a) and the average costs (b) found by some solvers in the ITC-2007 PE dataset

6. **Conclusions.** ETPs are highly constrained problems. Any small perturbation to an input may bring a significant impact on the solution. In this context, the development of efficient and effective input handling and solution finding is crucial.

This paper starts with an assumption that in its core an ETP contains the same basic elements that are events, resources, and constraints.

Based on this assumption, a generic solver can be developed to solve the problem through its core elements. An ETP that has been presented in the form of its core element is referred to as a stemmed form.

To realize this idea, we propose a 4 layers model ranging from identifying the core elements to solving the problem. The timetabling construction process involves the search for the proper resource for each event, taking into consideration the constraints imposed on the problem. The difference between ETPs relies on the constraints posed to them.

In the feasibility problem, several heuristics can be implemented straightforwardly by simply using the approach. Moreover, they are very effective to keep the search in the feasible subspace when it comes to the optimization process.

The application of the domain matrix and constraint graph support the optimization process of the search in yet another way. By using the domain matrix, a neighborhood structure can be defined immediately. This neighborhood structure is very effective in exploiting a promising solution area.

To show the effectiveness of the approach, experiments are conducted on some ETPs chosen from several publicly available datasets and a real dataset taken from an Indonesian college. The experiments confirm the assumption that by using this approach it is possible to deploy the same solver for those problems. However, some tuning activities are needed during the preprocessing stage and the solution finder stage. The results from these experiments also demonstrate that the qualities of the solutions are comparable to other results and in some cases improve some state-of-the-art results.

REFERENCES

[1] S. Kristiansen and T. R. Stidsen, *A Comprehensive Study of Educational Timetabling – A Survey*, DTU Management Engineering Report, http://orbit.dtu.dk/files/60366101/A_Comprehensive_Study.pdf, 2013.

[2] N. R. Sabar, M. Ayob, G. Kendall and R. Qu, A honey-bee mating optimization algorithm for educational timetabling problems, *Eur. J. Oper. Res.*, vol.216, no.3, pp.533-543, 2012.

[3] C. Gotlieb, The construction of class-teacher timetables, *Communications of the ACM*, vol.5, no.6, pp.73-77, 1963.

[4] R. A. O. Vrielink, E. A. Jansen, E. W. Hans and J. van Hillegersberg, Practices in timetabling in higher education institutions: A systematic review, *Ann. Oper. Res.*, vol.275, no.1, pp.145-160, doi: 10.1007/s10479-017-2688-8, 2019.

[5] A. Kheiri, E. Özcan and A. J. Parkes, A stochastic local search algorithm with adaptive acceptance for high-school timetabling, *Ann. Oper. Res.*, vol.239, no.1, pp.135-151, 2016.

[6] R. Bellio, S. Ceschia, L. Di Gaspero, A. Schaerf and T. Urli, Feature-based tuning of simulated annealing applied to the curriculum-based course timetabling problem, *Comput. Oper. Res.*, vol.65, pp.83-92, doi: 10.1016/j.cor.2015.07.002, 2016.

[7] R. Lewis and J. Thompson, Analysing the effects of solution space connectivity with an effective metaheuristic for the course timetabling problem, *Eur. J. Oper. Res.*, vol.240, no.3, pp.637-648, doi: 10.1016/j.ejor.2014.07.041, 2015.

[8] S. Abdul-Rahman, E. K. Burke, A. Bargiela, B. McCollum and E. Özcan, A constructive approach to examination timetabling based on adaptive decomposition and ordering, *Ann. Oper. Res.*, vol.218, no.1, pp.3-21, doi: 10.1007/s10479-011-0999-8, 2014.

[9] L. N. Ahmed, E. Özcan and A. Kheiri, Solving high school timetabling problems worldwide using selection hyper-heuristics, *Expert Syst. Appl.*, vol.42, no.13, pp.5463-5471, 2015.

[10] R. A. Aziz, M. Ayob, Z. Othman, Z. Ahmad and N. R. Sabar, An adaptive guided variable neighborhood search based on honey-bee mating optimization algorithm for the course timetabling problem, *Soft Comput.*, vol.21, no.22, pp.6755-6765, doi: 10.1007/s00500-016-2225-8, 2017.
[11] A. Bettinelli, V. Cacchiani, R. Roberti and P. Toth, An overview of curriculum-based course timetabling, *TOP*, vol.23, no.2, pp.313-349, 2015.
[12] S. Kristiansen, M. Sørensen and T. R. Stidsen, Integer programming for the generalized high school timetabling problem, *J. Sched.*, vol.18, no.4, pp.377-392, 2015.
[13] T. Mauritsius, A. N. Fajar, P. John et al., Novel local searches for finding feasible solutions in educational timetabling problem, *2017 5th International Conference on Instrumentation, Communications, Information Technology, and Biomedical Engineering (ICICI-BME)*, pp.270-275, 2017.
[14] T. Mauritsius and Harisno, A three phase heuristic for examination timetabling problem, *Proc. of 2018 International Conference on Information Management and Technology (ICIMTech)*, pp.466-471, 8528116, 2018.
[15] T. Mauritsius, N. Legowo and F. E. Gunawan, Reducing the timeslot used in examination timetable problem, *2018 International Conference on Information Management and Technology (ICIMTech)*, pp.211-216, 2018.
[16] B. McCollum and N. Ireland, University timetabling: Bridging the gap between research and practice, *PATAT*, pp.15-35, 2006.
[17] J. H. Obit, R. Alfred and M. H. Abdalla, A PSO inspired asynchronous cooperative distributed hyper-heuristic for course timetabling problems, *Adv. Sci. Lett.*, vol.23, no.11, pp.11016-11022, 2017.
[18] Y. Maruyama, Investigation of mothers' and fathers' concerns about programming education in elementary school, *ICIC Express Letters, Part B: Applications*, vol.10, no.12, pp.1085-1092, 2019.
[19] H. Babaei, J. Karimpour and A. Hadidi, A survey of approaches for university course timetabling problem, *Comput. Ind. Eng.*, vol.86, pp.43-59, doi: 10.1016/j.cie.2014.11.010, 2015.
[20] G. Lach and M. E. Lübbecke, Curriculum based course timetabling: New solutions to Udine benchmark instances, *Ann. Oper. Res.*, vol.194, no.1, pp.255-272, 2012.
[21] M. W. Carter, G. Laporte and S. Y. Lee, Examination timetabling: Algorithmic strategies and applications, *J. Oper. Res. Soc.*, vol.47, no.3, pp.373-383, 1996.
[22] T. B. Cooper and J. H. Kingston, The complexity of timetable construction problems, *International Conference on the Practice and Theory of Automated Timetabling*, pp.281-295, 1995.
[23] D. de Werra, A. S. Asratian and S. Durand, Complexity of some special types of timetabling problems, *J. Sched.*, vol.5, no.2, pp.171-183, 2002.
[24] E. K. Burke, J. Mareček, A. J. Parkes and H. Rudová, Decomposition, reformulation, and diving in university course timetabling, *Comput. Oper. Res.*, vol.37, no.3, pp.582-597, 2010.
[25] M. A. Al-Betar and A. T. Khader, A harmony search algorithm for university course timetabling, *Ann. Oper. Res.*, vol.194, no.1, pp.3-31, 2012.
[26] S. M. Al-Yakoob, H. D. Sherali and M. Al-Jazzaf, A mixed-integer mathematical modeling approach to exam timetabling, *Comput. Manag. Sci.*, vol.7, no.1, p.19, 2010.
[27] K. Socha, J. Knowles and M. Sampels, A max-min ant system for the university course timetabling problem, *International Workshop on Ant Algorithms*, pp.1-13, 2002.
[28] B. McCollum, A. Schaerf, B. Paechter, P. McMullan, R. Lewis, A. J. Parkes, L. D. Gaspero, Q. Rong and E. K. Burke, Setting the research agenda in automated timetabling: The second international timetabling competition, *INFORMS Journal on Computing*, vol.22, no.1, pp.120-130, 2010.
[29] C. W. Fong, H. Asmuni and B. McCollum, A hybrid swarm-based approach to university timetabling, *IEEE Trans. Evol. Comput.*, vol.19, no.6, pp.870-884, 2015.
[30] C. W. Fong, H. Asmuni, B. McCollum, P. McMullan and S. Omatu, A new hybrid imperialist swarm-based optimization algorithm for university timetabling problems, *Inf. Sci.*, vol.283, pp.1-21, 2014.
[31] S. Abdullah, H. Turabieh, B. McCollum and P. McMullan, A hybrid metaheuristic approach to the university course timetabling problem, *J. Heuristics*, vol.18, no.1, pp.1-23, doi: 10.1007/s10732-010-9154-y, 2012.
[32] S. Ceschia, L. Di Gaspero and A. Schaerf, Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem, *Comput. Oper. Res.*, vol.39, no.7, pp.1615-1624, doi: 10.1016/j.cor.2011.09.014, 2012.
[33] S. Abdullah, E. K. Burke and B. McCollum, Using a randomised iterative improvement algorithm with composite neighbourhood structures for the university course timetabling problem, *Metaheuristics*, pp.153-169, 2007.

[34] A. Rezaeipanah, S. S. Matoori and G. Ahmadi, A hybrid algorithm for the university course timetabling problem using the improved parallel genetic algorithm and local search, *Applied Intelligence*, https://doi.org/10.1007/s10489-020-01833-x, 2020.
[35] S. L. Goh, G. Kendall and N. R. Sabar, Improved local search approaches to solve post enrolment course timetabling problem, *Eur. J. Oper. Res.*, vol.261, no.1, 2017.
[36] S. L. Goh, G. Kendall and N. R. Sabar, Simulated annealing with improved reheating and learning for the post enrolment course timetabling problem, *J. Oper. Res. Soc.*, vol.70, no.6, 2019.
[37] D. de Werra, The combinatorics of timetabling, *Eur. J. Oper. Res.*, vol.96, no.3, pp.504-513, 1997.
[38] E. K. Burke and S. Petrovic, Recent research directions in automated timetabling, *Eur. J. Oper. Res.*, vol.140, no.2, pp.266-280, 2002.
[39] M. Tuga, R. Berretta and A. Mendes, A hybrid simulated annealing with Kempe chain neighborhood for the university timetabling problem, *The 6th IEEE/ACIS International Conference on Computer and Information Science (ICIS2007)*, pp.400-405, 2007.
[40] S. Kirkpatrick, Optimization by simulated annealing: Quantitative studies, *J. Stat. Phys.*, vol.34, nos.5-6, pp.975-986, 1984.
[41] P. Kostuch, The university course timetabling problem with a three-phase approach, *International Conference on the Practice and Theory of Automated Timetabling*, pp.109-125, 2004.
[42] B. Hajek, Optimization by simulated annealing: A necessary and sufficient condition for convergence, in *Lecture Notes Monograph Series, Vol. 8, Adaptive Statistical Procedures and Related Topics*, Institute of Mathematical Statistics, 1986.

Appendix A. List of Abbreviations.

- CH: Constructive Heuristic
- CTTP: Class-Teacher Timetabling Problems
- DHC: Dynamic Hard Constraints
- E2SA: Explorative and Exploitative Simulated Annealing
- ETP: Educational Timetabling Problem
- ExTP: Examination Timetabling Problems
- GA: Genetic Algorithm
- GOSP: Group of Similar Participants
- IPGALS: Improved Parallel Genetic Algorithm and Local Search
- ITC-2007 PE: Post Enrollment Track of the 2007 International Timetabling Competition
- LD: Largest Degree
- LS: Local Search
- LSD: Least Saturation Degree
- LWD: Largest Weighted Degree
- SA: Simulated Annealing
- SAIRL: Simulated Annealing with Improved Reheating and Learning
- SAR: Simulated Annealing with Reheating
- SC: Soft Constraints
- SHC: Static Hard Constraints
- TSSP: Tabu Search which is equipped with Sampling and Perturbation
- UCTP: University Course Timetabling Problem