Search Space Sampling by Simulated Annealing for Identifying Robust Solutions in Course Timetabling

Can Akkan
Sabancı Business School
Sabancı University
İstanbul, Turkey
can.akkan@sabanciuniv.edu

Ayla Gülcü
Dept. of Computer Science
Fatih Sultan Mehmet University
İstanbul, Turkey
agulcu@fsm.edu.tr

Zeki Kuş
Dept. of Computer Science
Fatih Sultan Mehmet University
İstanbul, Turkey
zeki.kus@stu.fsm.edu.tr

***Abstract*—For many combinatorial optimization problems, it is important to identify solutions that can be repaired without degrading solution quality in case changes in the data associated with the constraints make the initial solution infeasible, while ensuring that the new solution is not too different from the initial one. We propose a novel approach for finding such robust solutions based on a sample of solutions picked from the search space traversed by a simulated annealing algorithm. The sampled solutions are used to form a network of solutions. To explore the practical performance of this approach, we solve the widely studied curriculum-based course timetabling problem of the International Timetabling Competition 2007. With these benchmark instances, and sets of randomly generated disruption scenarios, we analyze the performance of some network-based estimators and show that the diversity of its neighbors is a significant indicator of a solution's robustness.**

***Index Terms*—fitness landscape, simulated annealing, robustness, flexibility, timetabling**

I. Introduction

This paper investigates the use of a network created by a large sample of solutions found through the search process of a simulated annealing (SA) algorithm to estimate the relative robustness of these solutions. In combinatorial optimization there is a sizable literature on the analysis of search spaces. A search space is defined for a given search process as a graph of feasible solutions where two nodes are connected if one can be reached from the other by a specific move operator. In the *fitness landscape analysis* literature, as discussed in a recent survey in [1], researchers focused on characterizing optimization problems mostly to determine which algorithm would be best suited to solving that problem, as well as adapting the algorithms during their execution to improve their performance, explaining unexpected algorithmic behavior in retrospect, or dividing problems into theoretical complexity classes. To the best of our knowledge, fitness landscape analysis has not been used for identifying robust or flexible solutions to combinatorial optimization problems. A work that looks into the relationship between evolvability and robustness in the context of natural systems, such as proteins and bacteria, is reported in [2]. In the context of combinatorial optimization, a robust solution is one that can be repaired when subject to pre-defined types of disruption that render it infeasible such that changes are kept to a minimum while obtaining high quality solutions. These two concerns are equally relevant in many re-optimization contexts. An initial schedule that can be repaired with maximum schedule stability is said to have high *solution robustness*. On the other hand, one that can be repaired with minimal decline in the solution quality is said to have high *quality robustness* [3]. The work presented here addresses both solution and quality robustness, the first one modeled as a constraint and the other as the objective of re-optimization.

A fitness landscape [4] consists of the following three elements: (i) a set X of solutions to the problem, (ii) a notion $\mathcal{X}$ of neighborhood, nearness, distance, or accessibility on X, and (iii) a fitness function $F : X \rightarrow \mathbb{R}$. In most practical combinatorial optimization problems it is inevitable to rely on a sample of solutions for defining the set X, as the set of all feasible solutions grows exponentially with the problem size. In their survey, [1] points to three pieces of work on what they call the *retrospective measures*, measures that involve the actual execution of an optimization algorithm: (i) adaptive walks [5]; (ii) consensus sequence plots [6]; (iii) estimating the number and distribution of local optima by performing a steepest ascent search from a random sample of starting positions [7]. We have chosen SA to find the solutions to be included in the set X because we would like to estimate the robustness of high quality solutions for the underlying optimization problem and SA is a well-established metaheuristic with demonstrated performance in many combinatorial optimization problems, including the course timetabling problem. Thus, our approach could be quite plausible for other combinatorial optimization problems, as well. The neighborhood we make use of is based on the Hamming distance between the solutions, and not a specific move operator since our goal is to understand the characteristics of the neighbors of a given solution in order to estimate the robustness of that solution. Since the robustness concept we adopt includes solution robustness, which we model by a constraint on the Hamming distance between the given solution and the one that repairs it, the topology of the area within a maximum radius defined by the Hamming distance around the given solution matters, not whether we can move between these two solutions with a simple local move. Such a distance measure between solutions is used for

This work was supported by TÜBİTAK grant 217M475.

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

graph coloring heuristics by [8], where it is confirmed that high quality solutions are grouped in clusters within spheres of specific diameter.

II. The robust course timetabling problem

A. The curriculum-based course timetabling problem

We have chosen to use the curriculum-based course timetabling problem (CB-CTP) definition and instances developed for the International Timetabling Competition 2007 (ITC-2007) [9], as they have become widely used benchmarking instances. In the CB-CTP of ITC-2007, a solution is an assignment of a period (day and time slot) and a room to all lectures of each course which satisfies all of the *hard constraints*, which are *Lectures* (all lectures of a course must be scheduled to distinct periods), *Conflicts* (lectures of courses in the same curriculum or taught by the same teacher must be scheduled in different periods), *Availabilities* (if the teacher of the course is not available to teach that course at a given period, then no lectures of the course can be scheduled at that period) and *RoomOccupancy* (two lectures cannot take place in the same room in the same period). The *soft constraints* are *RoomCapacity* (for each lecture, the number of students taking the course must be less than or equal to the number of seats of all the rooms that host its lectures), *MinimumWorkingDays* (the lectures of each course must be spread into the given minimum number of days), *CurriculumCompactness* (lectures belonging to a curriculum should be in consecutive periods) and *RoomStability* (all lectures of a course should be given in the same room).

The objective function, referred to as the *penalty* and denoted by P, is computed as the weighted sum of the violation of the soft constraints. Specifically, for the *RoomCapacity* constraint, each student above the capacity counts as 1 point of penalty. For the *MinimumWorkingDays* constraint, each day below the minimum counts as 5 points of penalty. For the *CurriculumCompactness* constraint, each isolated lecture in a curriculum counts as 2 points of penalty. Finally, for the *RoomStability* constraint, each distinct room used for the lectures of a course, but the first, counts as 1 point of penalty.

B. Disruption scenarios

In a typical pre-enrollment timetabling process, an initial timetable, S_0, is prepared based on a set of constraints provided by the professors and administrators. This timetable is announced to the staff, giving them some time to submit changes in constraints. The timetable is then re-optimized and the students enroll in courses based on this timetable. As discussed in [10], many different types of changes in constraints are possible before enrollment, such as new courses being added, others being canceled; some faculty arriving or leaving; certain periods ceasing to be feasible for some professors, or capacity of some rooms becoming insufficient for some lectures due to an increase in the number of students.

Here we use the disruption scenarios that have been first defined by [11]. These disruptions affect the feasibility of the periods for lectures and availability or capacity sufficiency of the rooms. By assuming disruptions that affect such limited resources, we believe we introduce sufficient variety and complexity. The types of disruptions that make up a scenario are as follows:

IP disruptions: The purpose of this disruption is to represent a situation when an instructor i learns he/she cannot teach at a period p to which one of her lectures is scheduled. Hence, this disruption type is specified by the tuple $\langle i, p\rangle$. For each disruption $\langle i, p\rangle$, unavailability constraints for all courses of instructor i at period p are added.

CP disruptions: The purpose of this disruption is to represent a situation when an instructor learns he/she cannot teach during a consecutive set of periods, and to make up for this unavailability, offers a set of consecutive periods which he designated as unavailable for the initial timetabling. This disruption is specified by a tuple $\langle c, \mathcal{P}_1, \mathcal{P}_2\rangle$ for course c. Given the set of feasible periods for course c, $\mathcal{P}_c^C$, $\mathcal{P}_1 \subseteq \mathcal{P}_c^C$ is a set of consecutive periods on the same day that become infeasible for course c and at least one of these periods is used by course c in S_0. $\mathcal{P}_2 \subseteq \mathcal{P} \setminus \mathcal{P}_c^C$ is a set of consecutive periods that become feasible for course c such that $|\mathcal{P}_2| \leq |\mathcal{P}_1|$.

CS disruptions: In some universities, before students' official registration, trial registrations or surveys are carried out to judge demand for courses. This disruption is introduced to represent a situation in which the initially planned capacity of a course is increased due to an estimated increase in demand. It is assumed that the planned number of students for a course is increased beyond the capacity of the room assigned to at least one lecture of that course (recall that room capacity is a soft constraint). This disruption is specified by a tuple $\langle c, s\rangle$, where s is the new number of students for course c and all events of this course are included in the set of room-disrupted events.

RP disruptions: Classrooms are often used for purposes other than classes, such as seminars, faculty meetings, etc. The purpose of this disruption is to represent such a situation in which a classroom becomes unavailable for scheduling classes for the duration of the semester. It is assumed that the availability of the room is lost for one or two consecutive periods on the same day. This disruption is specified by $\langle r, p, d\rangle$, where p is the first period that room r becomes unavailable, and d is the number of periods that become unavailable.

A set of disruptions of these types is referred to as a *disruption scenario*. All disruptions in a given disruption scenario are aggregated in two sets of disrupted lectures. Lectures e whose assigned periods in S_0 become infeasible due to IP and CP disruptions are denoted as E^P (the set of *period-disrupted* lectures) with size δ^p. Lectures e whose assigned rooms in S_0 become either infeasible due to RP disruptions or have insufficient capacity due to CS disruptions are denoted by E^R (the set of *room-disrupted* lectures) with size δ^r. Then, the set of disrupted lectures, E^D, equals $E^P \cup E^R$ and the number disrupted lectures, δ, equals $|E^D|$.

C. Robustness measure

The robustness objective is expressed as minimizing $E(R(S, Y_S))$, the expected value of a disruption measure $R(S, Y_S)$, where S is a given solution and Y_S is the random variable representing the disruptions.

Let, $\mathcal{F}(\sigma_i)$ be the set of all solutions that are feasible with respect to a disruption scenario σ_i and $D(S_0, S_1)$ be the Hamming distance between assigned-period arrays for all events $T_e(S_0)$ and $T_e(S_1)$ of these two solutions (hence, $D(S_0, S_1)$ is equal to the number of lectures that are assigned to different periods in these two solutions). Then, we define the following neighborhood set for a given solution S_0 and disruption scenario σ_i with δ_i^p period-disrupted and δ_i^r room-disrupted lectures:

$$\mathcal{N}(S_0, \sigma_i) = \{S : D(S_0, S) \leq f(\delta_i^p, \delta_i^r); S \in \mathcal{F}(\sigma_i)\} \quad (1)$$

Thus, if solution S_0 is disrupted by scenario σ_i, then switching to any solution in $\mathcal{N}(S_0, \sigma_i)$ would restore feasibility by rescheduling at most $f(\delta_i^p, \delta_i^r)$ lectures to a different period, where $f : (\mathbb{N}, \mathbb{N}) \rightarrow \mathbb{N}$. If there had been only period-disruptions, the *radius* $f(\delta_i^p, \delta_i^r)$ could be a multiple of δ^p, since every period-disrupted lecture must be moved to a different period. However, we also assume room-disruptions can occur, which may be repaired by moving a lecture to a different room without changing its period. Furthermore, the number of lectures affected by room-disruptions could be relatively large, as in the case of CS disruptions that may lead to the need for rescheduling all lectures of the corresponding course, some of which might be forced to a different period. So, given δ^p and δ^r, we define $f(\delta_i^p, \delta_i^r) = f^p\delta^p + f^r\delta^r,\ f^p > 1, f^r > 0$. For the computational results reported in Section V-D, we set $f^p = 2,\ f^r = 0.25$. Then, we define the robustness measure $R(S_0, \sigma_i)$ as:

$$R(S_0, \sigma_i) = \min_{S \in \mathcal{N}(S_0, \sigma_i)} \Phi_i(S, S_0), \text{ where} \quad (2)$$

$$\Phi_i(S, S_0) = P_{ave} \cdot 1_{D(S,S_0) > \delta_i^p} + (P(S) - P(S_0))^+ \quad (3)$$

where $x^+ \coloneqq \max(0, x)$ and P_{ave} is the average per lecture penalty for a randomly generated sample of solutions, and calculated for each problem instance separately. The solution sample, denoted by $\mathcal{S}$, is the union of the initial populations (each comprised of 40 solutions) of 30 runs of the MOGA algorithm of [12]. Thus, $P_{ave} = (1/(|\mathcal{S}||\mathcal{L}|)) \sum_{S \in \mathcal{S}} P(S)$, where $\mathcal{L}$ is the set of lectures. P_{ave} should be seen as a penalty term added so that solutions which only reschedule period-disrupted events to different periods are favored. Thus, in addition to quality robustness measured by $(P(S) - P(S_0))^+$, $R(S_0, \sigma_i)$ also incorporates a measure of solution robustness. Solution robustness is further ensured by the constraint $D(S_0, S) \leq f(\delta_i^p, \delta_i^r)$ in defining $\mathcal{N}(S_0, \sigma_i)$. If $\mathcal{N}(S_0, \sigma_i) = \emptyset$, then $R(S_0, \sigma_i)$ is set to a large value, B. For the computational experiments we set $B = 1200$, because the largest $\Phi_i(S, S_0)$ for a solution of the five ITC instances was 963.

Since a closed-form calculation of $E(R(S, Y_S))$ is not possible, a set of randomly generated sample of disruption scenarios, $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_N\}$, is used to calculate a sample average $\overline{R}(S, \sigma) = (1/N) \sum_{i=1}^{N} R(S, \sigma_i)$ as an estimate of $E(R(S, Y_S))$, using a reasonably large N. This is known as the Sample Average Approximation method [13].

D. SA Algorithm for Robustness Calculation

Calculation of $R(S, \sigma_i)$ makes use of an SA algorithm that repairs a given disrupted solution by minimizing the penalty function subject to the maximum distance constraint. Since the given solution is no longer feasible due to some disruptions violating hard constraints, the SA algorithm needs to restore the feasibility of the starting solution while at the same time minimize the violation of the soft constraints. Hence, some of the hard constraints in the CB-CTP are relaxed and their violations are penalized with large weights. Since in the CB-CTP there are only teacher unavailability constraints and the disruptions introduce the unavailability of rooms at certain periods, we distinguish these two types of availabilities as *TeacherAvailability* and *RoomAvailability* constraints. As these constraints may not be satisfied after a disruption, they are relaxed and their violations are penalized with a large weight (we used 100). The violations of the soft constraints with respect to the current disruption set are also recalculated to determine the penalty of the initial solution with respect to the new constraints. In addition to these constraints, the *Distance* constraint is introduced to ensure that given the initial solution S_0, and a solution generated in SA, S_1, $D(S_0, S_1) \leq f(\delta_i^p, \delta_i^r)$.

Algorithm 1 SA algorithm

```
   X ← produce_starting_solution()
2: {T_f, cr, nbr_iter} ← set_params( )
   {T_0, nbr_out, nbr_inn} ← comp_params(T_f, nbr_iter)
4: T_cur ← T_0
   for counter ← 0 to nbr_out do
6:   for inner_counter ← 0 to nbr_inn do
       X' ← GenerateRandomSoln(X)
8:     if accept(X', X, T_cur) then
         X ← X'
10:    end if
       T_cur ← T_cur * cr
12:  end for
   end for
```

In this SA algorithm one of the following random moves is used in the $GenerateRandomSoln$ function in line 7 (see Algorithm 1): (i) simple move, moving a lecture to an empty position; (ii) swap move, swapping a lecture with another lecture. These moves always satisfy the *RoomOccupancy* and *Lectures* constraints. At each iteration, a lecture (say l, scheduled at period t_1) and a new position (period, say t_2, and room pair) are randomly selected. If there is no lecture in the new position, then the selected lecture is moved to that position, otherwise the lecture is swapped with the one in the selected position. The $accept$ function in line 8 determines whether the new solution is accepted or not. If the new solution violates the *Distance* or the *Conflicts*

TABLE I: Example: The timetabling instance

c	$\mathcal{L}_c$	$\mathcal{P}_c$	N_c	T_c	K_c	MW_c
1	1, 2	1,2,3	25	tA	cA	2
2	3	2,4	15	tB	cA	1
3	4	1,4	30	tB	cB	1

constraints, it is automatically rejected. Improving solutions are always accepted, and non-improving solutions are accepted with probability $\exp\left(\left(-P(X') - P(X)\right)/T_{cur}\right)$.

A real time initial temperature selection strategy is used that employs a short burn-in period in which worsening moves as well as improving moves are accepted [14]. After this period, we set $T_0 = -1 * avg\delta P / \ln(p_{acc})$, where $avg\delta P$ is the average worsening penalty observed during the burn-in period, and p_{acc} is the initial acceptance probability. T_f is set using the parameter ρ, such that $T_0 = \rho \times T_f$. For the parameter cr, we adopted from the literature the mostly agreed upon value of 0.99. The number of outer iterations, nbr_out, in which the current temperature is cooled, and the number of solutions sampled at each temperature level, nbr_inn, are computed as $nbr\_out = \ln(T_f/T_0)/\ln(cr)$, and $nbr\_inn = nbr\_iter/nbr\_out$, where nbr_iter is the total number of iterations which is calculated based on the total CPU time allowed for the SA algorithm, which we set to be 680 seconds (tested to be sufficient to converge to a good solution). p_{acc} and ρ were set by the parameter tuning approach discussed in [11] to be 0.9 and 10,000, respectively.

E. An Example

We will assume a small timetabling problem in which we need to schedule 3 courses, over a two-day long "week", each day having 2 periods. For each course c, we are given the set of lectures $\mathcal{L}_c$, the set of feasible periods $\mathcal{P}_c$, the number of students N_c, the teacher of the course T_c, the curriculum it belongs to K_c, and the minimum working days the course has to be scheduled in, MW_c (see Table I). This problem has a total of 80 feasible solutions. Each solution i is represented by the array storing the assigned period for lecture l, $A_i(l)$, for $l = 1, \ldots, 4$ and the room each lecture is assigned, $R_i(l)$. The robustness of solution S_9, $A_9(l) = \{1, 3, 2, 4\}$ $R_9(l) = \{r1, r1, r2, r2\}$, with penalty $P_9 = 26$, is defined by the sets of disruptions of each disruption type. The set of IP disruptions for S_9 is $\{\langle 1, 1\rangle, \langle 1, 3\rangle, \langle 2, 2\rangle, \langle 3, 4\rangle\}$. The set of CP disruptions is $\{\langle 1, \{1\}, \{4\}\rangle, \langle 1, \{1, 2\}, \{4\}\rangle, \langle 1, \{3\}, \{4\}\rangle, \langle 2, \{2\}, \{1\}\rangle, \langle 2, \{2\}, \{3\}\rangle, \langle 3, \{4\}, \{1\}\rangle\}$. The set of RP disruptions is $\{\langle r1, 1, 1\rangle, \langle r1, 2, 1\rangle, \langle r1, 1, 2\rangle, \langle r1, 3, 1\rangle, \langle r1, 4, 1\rangle, \langle r1, 3, 2\rangle, \langle r2, 1, 1\rangle, \langle r2, 2, 1\rangle, \langle r2, 1, 2\rangle, \langle r2, 3, 1\rangle, \langle r2, 4, 1\rangle, \langle r2, 3, 2\rangle\}$. Finally, the set of CS disruptions is $\{\langle 1, 21\rangle, \langle 1, 22\rangle, \ldots, \langle 1, 40\rangle\}$, since courses whose first lecture is not assigned to the largest capacity room are exposed to this disruption (see Section V-A). Disruption scenarios are formed by combinations of these disruptions that result in at least 3 disruptions such that at most 1 disruption of type RP, and at most 2 disruptions of each of the other types occur.

TABLE II: Example: Calculation of R for a disruption scenario

$(A_i(l), R_i(l))$	$CP^+(l)$	$W(1)$	$P(S_i)$	$D(S_i, S_9)$	$\Phi_1(S_i, S_9)$
(1, r1) (3, r1) (2, r2) (1, r1)	13 13 0 10	2	36	1	10
(1, r1) (3, r2) (2, r2) (1, r1)	13 0 0 10	2	24	1	0
(1, r1) (3, r1) (2, r1) (1, r1)	13 13 0 10	2	36	1	10
(1, r1) (3, r2) (2, r1) (1, r1)	13 0 0 10	2	24	1	0
(1, r1) (3, r1) (2, r2) (1, r2)	13 13 0 0	2	26	1	0
(1, r1) (3, r2) (2, r2) (1, r2)	13 0 0 0	2	14	1	0
(1, r1) (3, r1) (2, r1) (1, r2)	13 13 0 0	2	26	1	0
(1, r1) (3, r2) (2, r1) (1, r2)	13 0 0 0	2	14	1	0
(1, r1) (3, r1) (4, r2) (1, r1)	13 13 0 10	2	36	2	29.5
(1, r1) (3, r2) (4, r2) (1, r1)	13 0 0 10	2	24	2	19.5
(1, r1) (3, r1) (4, r1) (1, r1)	13 13 0 10	2	36	2	29.5
(1, r1) (3, r2) (4, r1) (1, r1)	13 0 0 10	2	24	2	19.5
(1, r1) (3, r1) (4, r2) (1, r2)	13 13 0 0	2	26	2	19.5
(1, r1) (3, r2) (4, r2) (1, r2)	13 0 0 0	2	14	2	19.5
(1, r1) (3, r1) (4, r1) (1, r2)	13 13 0 0	2	26	2	19.5
(1, r1) (3, r2) (4, r1) (1, r2)	13 0 0 0	2	14	2	19.5

Of course, even for this small example there would be a very large number possible disruption scenarios, so here we demonstrate the robustness calculation for a single scenario, say σ_1, comprised of IP disruption $\langle 3, 4\rangle$, CS disruption $\langle 1, 33\rangle$, and RP disruption $\langle r1, 2, 1\rangle$. Thus $\delta_1^p = 1$ and $\delta_1^r = 2$, and assuming $f^p = 2$ and $f^r = 0.5$, we get $f(\delta_i^p, \delta_i^r)$ to be 2. Of the 80 feasible solutions for the problem at hand, $\mathcal{N}(S_0, \sigma_1)$ has the 16 solutions listed in Table II, where $CP^+(l)$ denotes the room capacity violation of lecture l, $W(c)$ is the number of work days for course c. For this example, we set $P_{ave} = (1/80)\sum_i P(S_i) = 19.5$, using all 80 solutions, rather than a sample of solutions. Then $\Phi_1(S_i, S_0)$ are calculated as shown in Table II, which results in $R(S_9, \sigma_1) = 0$.

III. Network-based robustness estimation

Given the fitness landscape definition in Section I, it is natural to model it as a network in which each solution is represented by a node and edges connect nodes based on the chosen neighborhood definition. Given this network, our objective is to develop a heuristic robustness estimator for a given solution. All the tested estimators make use of the information on the neighbors of the given solution (such as its distances to the neighbors and the objective values of these neighbors). Of course, these neighbors cannot be a full enumeration of all the feasible solutions but we hope some estimators based on this sample of neighbors do indicate how robust the solution is. Given this approach, there are three main algorithm design questions that need to be answered: (1) how to generate the solutions; (2) how to select the sample of solutions from among the generated solutions and how big this sample should be; (3) what are some heuristic estimators that are correlated with the robustness measure $\overline{R}$.

We use SA for generating the solutions because we need to explore the solution space of high quality solutions, so that from among such solutions a decision maker could be provided choices to trade-off solution quality and solution robustness. To this end, SA is a good algorithmic choice because it gradually converges to a good (hopefully optimal)

solution and it has been shown to find some of the best known solutions to the ITC-2007 CB-CTP instances (see e.g. [15] and [16]). For sampling among the solutions generated by the SA algorithm, we have decided to only select from among the accepted solutions (SA uses a probabilistic acceptance rule, see Section IV). For determining which of these accepted solutions get selected to the sample, we defined two parameters. The first one, nc, is the number of collected solutions, and second one is the step size, s. Then, starting with the solution accepted in the last iteration of SA, going backwards and skipping every s^{th} accepted solution, a total of nc solutions are collected into the sample. Note that the final sample size, n, could be slightly less than nc because some of the collected solutions could be identical. As discussed in Section V, we ran our experiments with nc equals 50000 and 100000, and s equals 0 and 1.

For developing network-based robustness estimators, it is important to acknowledge that for a given solution, it would not be sensible to define a robustness estimator that is independent of the size of the disruption. Hence, here we develop robustness estimators for given a maximum disruption size. Letting $G(N, E)$ denote the solution network, we let the neighbors of node v be the set of nodes w such that $D(v, w) \leq \rho$, where ρ is the maximum number of events that would need to be rescheduled to a *different period* in order to respond effectively to a disruption scenario. Thus, the set of edges E contains all pairs of nodes v, w with $D(v, w) \leq \rho$.

Given $G(N, E)$, we hypothesize the following regarding the estimation of robustness of a given solution, v:

- The more the neighbors the better: If the search heuristic is able to find many neighbors to v, then it would be more likely to find feasible solutions to deal with disruptions.
- The lower the increase in the neighbors penalty the better: If $(P(w) - P(v))^+$ is small for the neighbors, w, of the disrupted solution v, it would be more likely to repair v with minimal increase in penalty.
- The more diverse neighbors the better: If neighbors of v are at a diverse set of distances, this might indicate repairing v when disruptions of different sizes occur would be easier. Futhermore, if the neighbors have the events assigned to different periods, it could be easier to repair disruptions. For instance, if all neighbors have event e scheduled at period p, then if a disruption makes period p infeasible for event e, it may be less likely to find a feasible solution to repair v.

A. Robustness estimators utilizing the maximum radius neighborhood

The following set of estimators use the maximum radius around the given solution, i.e. $\rho = f^p N^p + f^r N^r$, where N^p and N^r are the maximum number of period-based and room-based disruptions possible. To simplify notation, in the rest of the paper we will denote $\mathcal{N}(v, 0, \rho)$ as $\mathcal{N}(v)$.

$$d(v) = |\mathcal{N}(v)| \tag{4}$$

$$pd(v) = \sum_{w \in N(v)} \frac{1}{1 + \frac{(P(w) - P(v))^+}{P(v)}} \tag{5}$$

$$div(v) = \frac{\sum\limits_{(u,w) \in E^N(v)} D(u, w) + \sum\limits_{(u,w) \in M^N(v)} \rho}{\rho(d(v)(d(v) - 1)/2)} \tag{6}$$

$$dc(v) = \sum_{d=0}^{\rho} 1_{\mathcal{N}(v,d) \neq \emptyset} \tag{7}$$

where,

$$E^N(v) = \{(u, w) : u, w \in \mathcal{N}(v) \text{ and } (u, w) \in E)\}$$
$$M^N(v) = \{(u, w) : u, w \in \mathcal{N}(v) \text{ and } (u, w) \notin E)\}$$
$$\mathcal{N}(v, d) = \{w : \ D(v, w) = d\}$$

Equation (4) gives the degree of node v in the solution network. Equation (5) is the penalty-weighted degree of solution v in the network. Equation (6) measures the diversity of the neighbors of a given solution v. If the distance between two neighbors is large, we conclude that they are diverse. Since the network has a cutoff distance of ρ, if an edge does not exist between two nodes it implies that the distance between these nodes is at least $\rho + 1$. Given this definition, $div(v) \in [0, 1]$ and the more diverse the neighbors of v are, the closer it gets to 1. Equation (7) gives the number of distances at which there exists a neighbor for solution v, and hence it is a measure how different the neighbors are.

Furthermore, for all neighbors w of solution v (i.e. $w \in \mathcal{N}(v)$), let $\mathcal{P}_u(v)$ denote the set of unique penalty values $P(w)$. Furthermore, let $\mathcal{P}D_u(v)$ represent the set of unique $(P(w), D(v, w))$ tuples. Then, we hypothesize that size of these sets, as measures of diversity among the neighbors of solution v, could be good estimators of its robustnesss:

$$dup(v) = |\mathcal{P}_u(v)| \tag{8}$$

$$upd(v) = |\mathcal{P}\mathcal{D}_u(v)| \tag{9}$$

Note that we do not define a similar estimator for the degrees, as $|\mathcal{D}_u(v)|$, where $\mathcal{D}_u(v)$ is the set of unique distance values $D(v, w)$ for $w \in \mathcal{N}(v)$, because that would be equal to $dc(v)$ defined in (7).

B. Robustness estimators inspired by the fitness landscape literature

For us the topology of the area around a selected solution is important. Among the topological features of fitness landscapes and the associated measures discussed in the survey of [1], the following bear some relevance to the robustness objective being addressed in this work.

Fitness distribution in search space: This concerns how the fitness values are distributed across the search space, in terms of the frequency of different fitness values, taking into account the position of these fitness values within the search space.

Ruggedness: This is about the level of variation in fitness values in a fitness landscape. If neighboring solutions have

very different fitness values, then the result is a rugged landscape. A *basin of attraction* B of a local optimum s^l is defined by [7] as a set of points $s_1 \dots s_k$ of the search space, such that a steepest descent algorithm (assuming a minimization problem) starting at s_i $(1 \leq i \leq k)$ ends at the local optimum s^l in a finite number of steps. A smooth landscape would be one with a single large basin of attraction or a flat landscape.

Neutrality: Neutrality refers to having neighboring solutions with equal fitness values. A *plateau* is defined as a set S^p such that $\forall s^p \in S^p, F(s^p) = a$, where a is a constant, and $F(s)$ is the fitness of solution s [17]. Several measures are suggested in the literature for neutrality, such as average neutrality ratio and average fitness gain [18].

Evolvability: Evolvability is related to an algorithm's ability to evolve the set of generated solutions. [19] describes evolvability with particular reference to genetic algorithms as the ability of a population to produce offspring that are fitter than their parents.

Deceptiveness: The presence of misleading information is known as deception. Fitness distance correlation is one of the most widely used measures developed to predict deceptiveness, and it was first developed for genetic algorithms [20].

Most of the research highlighted above regarding fitness landscapes limit their attention to NK-landscapes problems, which are specially designed for this purpose (see [21] for an overview), or unconstrained optimization problems (such as the binary quadratic programming problem [22]). In large real-life combinatorial optimization problems it would be practically impossible to enumerate the entire search space for a given move, which necessitates sampling of the solutions.

Using the insights gained by looking into the literature briefly discussed above, we have defined the following metrics as potential robustness estimators:

Neutrality degree: Equals the size of the set of neutral neighbors.

$$nd(v) = |\mathcal{N}^{\nu}(v)| \tag{10}$$

where, due to our robustness measure discussed in Section II-C, we have chosen to define the set of neutral neighbors as follows:

$$\mathcal{N}^{\nu}(v) = \{w : w \in \mathcal{N}(v),\ P(w) \leq P(v)\} \tag{11}$$

Neutrality ratio: Equals the ratio between neutrality degree and the degree of the solution (similar to measure with the same name in [18]).

$$nr(v) = |\mathcal{N}^{\nu}(v)|/|\mathcal{N}(v)| \tag{12}$$

Neutral zone depth: This measure is inspired by the neutral walk of [23]. It is equal to the maximum distance from node v, starting with 0, up to which there exists at least one solution with penalty less than or equal to $P(v)$ at every distance.

$$\begin{aligned} nzd(v) = &\max\{j : \exists w_i \text{ s.t. } D(v, w_i) = i, \\ &P(w_i) \leq P(v),\ \forall\ 0 \leq i \leq j\} \end{aligned} \tag{13}$$

Neutral zone spread: This is the number of distances up to δ, for which there is at least one neighbor of v with penalty less than or equal to $P(v)$. Note that, by definition, $nzs \geq nzd$.

$$\begin{aligned} nzs(v) = &|\{i : \exists w_i \text{ s.t. } D(v, w_i) = i, \\ &P(w_i) \leq P(v),\ 0 \leq i \leq \delta\}| \end{aligned} \tag{14}$$

Average fitness loss: This is similar to average fitness gain of [18], which is proposed as a measure of evolvability.

$$afl(v) = \sum_{w \in \mathcal{N}(v)} \left(P(w) - P(v)\right)^{+} / d(v) \tag{15}$$

Fitness-distance correlation: Introduced by [20] to measure search difficulty for predicting the performance of genetic algorithms, in which distances are measured to the global optimum. $fdc(v)$ is defined as the Pearson correlation coefficient between $D(v, w)$ and $P(w)$ for $w \in \mathcal{N}(v)$. A close to 0 correlation would mean there is a mix of high and low penalty solutions at all distances to the solution, and therefore is likely to be an indicator of robustness for solution v.

IV. The Simulated Annealing Algorithm for Sampling Solutions

As stated above, we have obtained the solutions used in building the solution networks by running an SA algorithm. SA not only produced a very large set of solutions but also these solutions are quite diverse due to the randomized characteristic of the SA that gradually converges to a set of good solutions. The SA algorithm (see Algorithm 1) is essentially set up as was done in [15], which proved to produce very good solutions for the ITC2007 CB-UCT problem. Two local moves were used to randomly generate a solution from a given one in each iteration. These are the *simple move* and the *swap* moves. In each iteration, for a randomly selected lecture, a period and a room (a "position") are randomly selected from among all feasible positions (all hard constraints, except for the *Conflict* are met). If there is no lecture already scheduled at that position, a simple move is made, otherwise a swap move is made (only if such moves are feasible). If the move produces a violation of the *Conflict* constraints, it is rejected. Otherwise, if the new solution has a lower penalty value than the previous one, it is accepted. If it has a higher penalty value, it is accepted with probability $\exp\left(\left(-P(X') - P(X)\right)/T_{cur}\right)$.

Since ITC-2007 instances have been extensively researched, we benefited from the findings of earlier studies. [15] found that geometric cooling schedule with a cooling rate of 0.99 (*cr*) gives the best results. [24] showed that as long as the cooling rate is close to 1 the performance of the algorithm is not sensitive to minor changes in its value. Here we used the real-time strategy of [14], which was discussed in Section II-D, to set T_0. We set *cr* to 0.99 and p_{acc} to 0.7 by the parameter optimization we reported in [25].

The number of iterations in which the temperature is updated, nbr_out_iter, and the number of randomly generated solutions at each temperature, nbr_inner_iter are set so that total CPU time is equivalent to the time limit set at the ITC-2007 competition, which was 215 seconds.

When we ran the SA algorithm the number of accepted solutions for the ITC-2007 instances ranged between 463,311 and 987,339. These are quite sufficient numbers for setting up large solution networks. As we discussed above, when we set $nc = 100K$ and $s = 1$ we select 100,000 solutions from among the last 200,000 accepted solutions.

V. Computational study

A. Generating the problem instances

A problem instance is comprised of the timetabling instance of ITC-2007, a disruption scenario, and a feasible solution for the ITC-2007 instance. We selected five timetabling instances because they are the most constrained (thus potentially difficult) instances in terms of conflict intensity, teacher availability, and room occupancy [26]. Specifically, ITC5 and ITC12 are the timetabling instances with the highest conflict intensity; ITC2 and ITC5 are the top two in terms of lowest teacher availability; and finally ITC1 and ITC7 have the highest room occupancy. For each ITC-2007 instance and its solution S_0, the disruptions are randomly generated as follows:

1) IP: $\langle i, p\rangle$ First an instructor i is chosen randomly. Given i, first a course of that instructor is chosen randomly, and then a lecture of the selected course is chosen randomly. The period, p at which the selected lecture is scheduled in S_0 is designated as infeasible for instructor i. Period p becomes unavailable for all courses of instructor i. There can be at most one $\langle i, p\rangle$ disruption for any instructor i.
2) CP: $\langle c, \mathcal{P}_1, \mathcal{P}_2\rangle$ First an instructor, and then a course c of that instructor are chosen randomly. Let p denote the period of a randomly selected lecture of course c. If the previous period is feasible for the course, then the starting period of $\mathcal{P}_1$ is set as $p-1$, otherwise as p. Then if $p+1$ is feasible, then the ending period of $\mathcal{P}_1$ is set as $p+1$, otherwise as p. So the size of $\mathcal{P}_1$ is 1,2 or 3. Then, the number infeasible periods for course c at each day is calculated. If none of the days has at least $|\mathcal{P}_1|$ infeasible periods, the day with the maximum number of infeasible periods is selected. Otherwise, if there is at least one day with $|\mathcal{P}_1|$ or more infeasible periods, one such day is randomly selected. Given the chosen day, a set of consecutive infeasible periods starting with the first infeasible period of that day are assigned to $\mathcal{P}_2$, such that $|\mathcal{P}_2| \leq |\mathcal{P}_1|$. There can be at most one $\langle c, \mathcal{P}_1, \mathcal{P}_2\rangle$ disruption for a given course c.
3) RP: $\langle r, p, d\rangle$ A room r is randomly selected. Duration d is generated from the discrete uniform distribution $DU(1, 2)$. Given d, period p is randomly generated so that if $d = 2$, periods p and $p+1$ are both on the same day. There can be at most one $\langle r, p, d\rangle$ disruption for a given room r.
4) CS: $\langle c, s\rangle$ First, a course c is randomly chosen, so that its first lecture, e_c, is not assigned to the room with the largest capacity, CP^{max} (this could have been any lecture of the course, but for the sake of convenience we choose its first lecture). Then, the new number of students, s, is randomly drawn from $DU(lowlim + 1, lowlim + gap)$, where $lowlim = \max(NS(c), CP(\rho(e_c))$ and $gap = \min(CP^{max} - lowlim, NS(c))$. There can be at most one $\langle c, s\rangle$ disruption for a given course c.

The number of RP disruptions is drawn from $DU(0, 1)$ while the others are drawn from $DU(0, 2)$ so that the total number of disruptions in a disruption scenario is at least 3.

B. Generated networks

In this section we provide a set of descriptive statistics on the solution networks, which confirm that the selected ITC-2007 instances are significantly different from each other for the purposes of this research. In the following discussion, each network generated for ITCi is denoted by $N_i^{nc,s}$. For all four networks associated with each of the five instances, Table III reports a set of statistics on the penalty values and degrees of the solutions. Based on these statistics, one can make several important observations. First, for all instances and networks, the actual number of nodes in the network, n, is very close to nc, showing that SA rarely visited the same solution in the search process. Second, both in terms of their values and variability, we see significant differences between the instances, not only in terms of penalties but also degrees. ITC1 has no variability in penalty values and the degrees have relatively small variability for all networks, with small positive skewness. ITC2 has some small variability and positive skewness for both penalty and degree of nodes. Another significant characteristic of ITC2 networks is the difference in the mean penalty of the networks (75.25 for $N_2^{50,0}$ as opposed to 213.9 for $N_2^{100,1}$). This suggests that convergence to low penalty values occurred relatively late in the SA search process. ITC5 has smaller variability in penalties, compared to ITC2, whereas the degrees are several orders of magnitude larger than those for ITC2. As a matter of fact, ITC5 stands out with extremely large degrees. We also observe that the only networks with negatively skewed degree distributions are three of the four networks built for ITC5. ITC7, on the other hand, differs from the previous instances in having negatively skewed penalty distributions for all four of its networks and has the largest positive skewness values for the degree distributions among all five instances. As was the case for ITC2, ITC7 networks have significant difference between their mean penalty values (297.7 for $N_7^{50,0}$ as opposed to 890.7 for $N_7^{100,1}$), but unlike ITC2 networks the average degree of the networks drop significantly from 98.11 to 20.19. Combining this information with the skewness of the degrees, we can conclude that ITC7 networks has some of the smallest degrees among all networks combined with significant number of solution with poor penalty values. ITC12 networks have negative penalty skewness and positive degree skewness as was the case for ITC7, but unlike ITC7 the degrees are quite large and solution quality difference between the solutions forming these networks is significantly less. Based on these observations we can conclude that there

TABLE III: Statistics on the penalty and degree distributions

		Penalty					Degree				
Network	n	Min.	Q2	Mean	Max.	Skew	Min.	Q2	Mean	Max.	Skew
$N_1^{50,0}$	49733	6	6	6	6	-	16	36.0	37.2	82	0.87
$N_1^{50,1}$	49743	6	6	6	6	-	10	17.0	17.88	40	0.86
$N_1^{100,0}$	99443	6	6	6	6	-	21	36.0	36.74	82	0.87
$N_1^{100,1}$	99460	6	6	6	6	-	9	17.0	17.89	41.0	0.92
$N_2^{50,0}$	48747	75	75	75.25	80	3.13	27	72.0	75.32	220	0.96
$N_2^{50,1}$	49302	75	78	124.9	248	0.47	13	50.0	52.86	172	0.61
$N_2^{100,0}$	98644	75	78	124.9	248	0.47	27	101.0	106.7	343	0.61
$N_2^{100,1}$	99281	75	223	213.9	429	−0.13	13	46.0	49.2	172	1.05
$N_5^{50,0}$	48540	405	405	406.9	424	1.93	8276	46160	44910	48330	−3.61
$N_5^{50,1}$	49165	405	418	420.2	472	0.48	105	29150	22490	37710	−0.71
$N_5^{100,0}$	98299	405	418	420.2	472	0.48	213	58300	44960	75500	−0.71
$N_5^{100,1}$	99090	405	445	448.7	578	0.45	26	1506	12070	39700	0.62
$N_7^{50,0}$	49867	39	323	297.7	647	−0.28	11	54	98.11	1310	3.59
$N_7^{50,1}$	49959	39	565	504.5	1023	−0.34	4	16	30.56	641	5.04
$N_7^{100,0}$	99864	39	566	504.8	1024	−0.34	9	32	61.82	1310	5.07
$N_7^{100,1}$	99959	39	851	890.7	2467	0.24	2	11	20.19	641	7.08
$N_{12}^{50,0}$	48467	378	410	404.6	449	−0.13	196	796	3203	12320	1.22
$N_{12}^{50,1}$	49280	378	433	427.9	494	−0.33	77	288.0	926.8	6233	2.34
$N_{12}^{100,0}$	98377	378	433	428	494	−0.33	156	578	1823	12320	2.36
$N_{12}^{100,1}$	99218	378	469	466.4	571	−0.22	46	186.0	530.8	6233	3.69

is considerable diversity in the characteristics of the networks formed for these instances.

C. Solutions selected for analysis

For each instance, a set of 60 solutions have been selected to carry out correlation analysis between the network-based robustness estimators and the robustness measure. The purpose of the selection procedure was to obtain a diverse set of solutions from each network. While ensuring this diversity, we also wanted to have the selected solutions appear in all four networks for each instance, if possible. For ITC7, and ITC12, all 60 solutions, for ITC1, ITC2, and ITC5, 55, 38, and 46 solutions, respectively, appeared on four networks. The remaining solutions appeared on two networks.

Solution characteristics that were taken into account to achieve the desired diversity were penalty values and the degree of nodes. Frequency tables were formed of all solutions based on intervals of these characteristics, and then a solution was selected from the solutions that fall into selected intervals. The selections were made from among the solutions with penalties that are close to the minimum penalty value, P_{min}, found by the SA algorithm. The P_{min} values for ITC1, 2, 5, 7, and 12 are 6, 75, 405, 39, and 378, respectively.

For ITC1, since all solutions in the networks had the same penalty, solutions were grouped by their degrees and then solutions were selected from each of the subsets of solutions with a range of degrees. For instance, for $N_1^{50,0}$ these degrees were equal to 21, 25, 35, 45, 55, 65 and 75+ and were 10, 15, 20, 25, 30, 35, 37 for $N_1^{50,1}$. For the other instances, for each penalty interval (in the case of ITC2 individual penalty values), we sampled one solution from the smallest degree interval, and the second solution from the largest degree interval. In addition, we chose some solutions from an intermediate degree with small penalty values (often, as the case for $N_2^{50,0}$, these were the among the largest subset of solutions). In a few cases when a degree interval contained only already selected solutions, we moved to the adjacent degree interval for the same penalty interval. For the networks $N_2^{50,0}$ and $N_2^{50,1}$, the subsets of solutions were formed by individual penalty values and degree intervals, since all solutions in these networks had one of the penalties in $\{75, 76, 77, 78, 79, 80\}$. The intervals used for $N_2^{50,0}$ and $N_7^{50,1}$ are given in Table IV, where the top row for each network contains the penalty intervals, and the remaining rows contain the degree intervals used for the corresponding penalty interval. One solution is randomly picked from each penalty-interval, degree-interval pair, while ensuring that it appears in as many networks as possible.

TABLE IV: Penalty and degree intervals used to select the sample of solutions from $N_2^{50,0}$ and $N_7^{50,1}$

$N_2^{50,0}$ – P :	75	76	77	78	79	80
	[13, 32] [73, 92] [173, 192]	[13, 32] [73, 92] [133, 152]	[13, 32] [73, 92] [113, 132]	[33, 52] [93, 112]	[33, 52] [93, 112]	[53, 72] [73, 92]
$N_7^{50,1}$ – P :	39–40	41-42	43–44	45–46	47-48	49–50
	[4, 53] [54, 103] [104, 153] [154, 203]	[4, 53] [54, 103]	[4, 53] [104, 153]	[4, 53] [104, 153]	[[4, 53] [54, 103]	[4, 53] [54, 103]

D. Performance of the estimators

In order to evaluate the performance of the robustness estimators, we have first calculated the correlation coefficients of all estimators among themselves and with $\overline{R}$. As discussed in the previous section, for ITC1 all collected solutions had the same penalty value, and therefore for some of the estimators, the corresponding correlation was undefined. Some of the estimators were strongly correlated with each other in all networks and all instances, in which case we selected one of them for further analysis. Specifically, from d and pd, we chose d, and from dup and upd, we chose dup. Then, among remaining ones, fdc had consistently close to zero correlations with $\overline{R}$ for all networks. For the others, Spearman's correlation coefficient values are reported in Table V.

Recalling that lower $\overline{R}$ values indicate better robustness, we observe that, as we initially hypothesized, increased degree (d) and diversity of neighbors measured by div both are mostly negatively correlated with $\overline{R}$ for all networks and instances (the few exceptions for d are three ITC12 networks, and two ITC5 networks for div). On the other hand, correlation analysis clearly showed that a single estimator cannot be a reliable indicator of the robustness of a solution. Thus, in the next stage, we used linear regression modeling to gain more insight regarding the performance of these indicators and

TABLE V: Correlation of selected estimators with $\overline{R}$

Network	d	nd	nr	nzd	afl	div	dc	dup
$N_1^{50,0}$	-0.12	-0.12	NA	NA	NA	-0.22	-0.06	NA
$N_1^{50,1}$	-0.11	-0.11	NA	NA	NA	-0.22	0.07	NA
$N_1^{100,0}$	-0.16	-0.16	NA	NA	NA	-0.25^{*}	-0.13	NA
$N_1^{100,1}$	-0.15	-0.15	NA	NA	NA	-0.24^{*}	-0.01	NA
$N_2^{50,0}$	-0.26	-0.27	-0.14	$0.36^{\flat}$	0.14	-0.3^{*}	0.1	0.09
$N_2^{50,1}$	-0.22	-0.26	-0.14	0.25	0.14	-0.25	-0.09	0.11
$N_2^{100,0}$	$-0.33^{\sharp}$	-0.18	0.2	0.08	-0.21	-0.21	-0.05	-0.16
$N_2^{100,1}$	$-0.33^{\sharp}$	-0.18	0.24^{*}	$0.26^{\flat}$	-0.25^{*}	-0.16	-0.23^{*}	-0.15
$N_5^{50,0}$	-0.13	-0.23	-0.23	0.09	0.19	0.14	0.05	0.2
$N_5^{50,1}$	-0.21	-0.26^{*}	-0.28^{*}	0.02	0.21	0.09	-0.16	-0.28^{*}
$N_5^{100,0}$	-0.09	-0.13	-0.15	0.08	0.17	-0.11	0.11	-0.16
$N_5^{100,1}$	-0.1	-0.13	-0.16	0.02	0.17	-0.01	-0.1	0.1
$N_7^{50,0}$	-0.14	-0.01	0.11	-0.02	-0.11	-0.12	-0.01	0.1
$N_7^{50,1}$	-0.14	-0.01	0.12	-0.02	-0.12	-0.13	0.05	0.1
$N_7^{100,0}$	-0.14	-0.01	0.11	-0.02	-0.11	-0.12	-0.01	0.1
$N_7^{100,1}$	-0.14	-0.01	0.12	-0.02	-0.12	-0.13	0.05	0.1
$N_{12}^{50,0}$	0.02	-0.04	-0.17	0.07	0.16	-0.12	0.15	0.11
$N_{12}^{50,1}$	-0.01	-0.09	-0.12	0.15	0.1	-0.21	0.24	0.07
$N_{12}^{100,0}$	0.02	-0.04	-0.17	0.07	0.16	-0.12	0.15	0.11
$N_{12}^{100,1}$	0.02	-0.04	-0.16	0.1	0.15	-0.11	0.18	0.14

$\sharp:\ p < .01, \flat:\ p < .05, *:\ p < .10$

TABLE VI: Linear regression model p-values

Network	F stat.	β_d	Neutrality/fitness-related β_{nd}	β_{nr}	β_{nzd}	β_{afl}	Diversity-related β_{div}	β_{dc}	β_{dup}
$N_1^{50,0}$	0.056	0.049					0.020		
$N_1^{50,1}$	0.051	0.043					0.017		
$N_1^{100,0}$	0.062	0.085					0.031		
$N_1^{100,1}$	0.071	0.099					0.035		
$N_2^{50,0}$	0.025				0.025		0.089		
$N_2^{50,1}$	0.104				0.121		0.167		
$N_2^{100,0}$	0.000			0.010		0.000	0.010		
$N_2^{100,1}$	0.000			0.013		0.000	0.015		
$N_5^{50,0}$	0.047		0.029						0.065
$N_5^{50,1}$	0.097		0.416						0.190
$N_5^{100,0}$	0.284					0.285			
$N_5^{100,1}$	0.345					0.345			
$N_7^{50,0}$	0.156		0.118			0.132			
$N_7^{50,1}$	0.179		0.123			0.146			
$N_7^{100,0}$	0.156		0.118			0.132			
$N_7^{100,1}$	0.171		0.123			0.146			
$N_{12}^{50,0}$	0.239	0.095	0.086		0.319		0.097		
$N_{12}^{50,1}$	0.072	0.038	0.035		0.043		0.063		
$N_{12}^{100,0}$	0.239	0.095	0.086		0.319		0.097		
$N_{12}^{100,1}$	0.072	0.038	0.035		0.043		0.063		

obtained the best regression model for each network. The p-values associated with the coefficients of the indicators used as the independent variables, along with the p-values of the F-statistic for the significance of the overall regression model, are reported in Table VI for each of these regression models. For each ITC instance, the model with the smallest p-value for the F-statistic is highlighted with gray shading.

Although having different regression models for different ITC-2007 instances might seem to be a weak result, considering the goals of this research, given the results in Table VI, we can make the following observations: For all instances, except for ITC7, there is at least one statistically significant regression model with F-statistic p-value less than $\alpha = 0.10$. For these four instances, the best regression models are obtained using networks $N_1^{50,1}$, $N_2^{100,0}$, $N_5^{50,0}$ and $N_{12}^{50,1}$. Thus, there is no indication of network size $100K$ being better than $50K$. All of these four models have one diversity measure: div, appears in three of them and dup appears in one. div is the most frequently used estimator, since the other estimators in these four models are d and nd that appear two times, nr, nz, afl, and dup that appear one time. Overall, we can conclude that diversity (mostly measured by div) seems to be a significant factor affecting robustness. Noting that nd (neutrality degree) is closely related to the degree of a node, we observe that d or nd are significant estimators in three of the four models, suggesting that number of neighbors is closely related with robustness, as well. Overall, all models have some combination of the estimators that are designed to quantify the number, diversity and neutrality of neighbors, however the specific estimators within these categories change with different instances.

Looking back into the network statistics discussed in Section V-B, we can gain valuable insights into why the statistical results on ITC7 turned out be unsatisfactory. No other network has such small degrees and so many solutions with inferior penalty values. For instance, comparing the statistics for networks $N_{12}^{50,1}$ and $N_7^{50,1}$, we see that $N_7^{50,1}$ has significantly smaller degree solutions. Average degree of $N_7^{50,1}$ is 30.56 and skewness is 5.04 suggesting most of the solutions have degrees below the mean, whereas the minimum degree of $N_{12}^{50,1}$, 77, is more than double the average for $N_{12}^{50,1}$. In addition, the range of penalty values for $N_7^{50,1}$ is significantly larger, $[39 - 1023]$ as opposed to $[378 - 494]$ for $N_{12}^{50,1}$. Figure 1 shows the first collected solution for ITC7 had approximately 22 times the best penalty value ($Pmin$) and SA took too long to converge to good P values. For other ITC instances SA was able to collect a very large sample of good solutions.

VI. Conclusions

We have looked into whether metrics that are calculated from a large network of solutions sampled from the solution space of a SA algorithm for the curriculum-based course timetabling problem can be used as estimators of how robust some of these solutions are. Statistical analysis carried out on the performance of 12 estimators for well-known benchmark instances of the problem suggests that combined use of metrics that measure the number, neutrality/fitness and diversity of

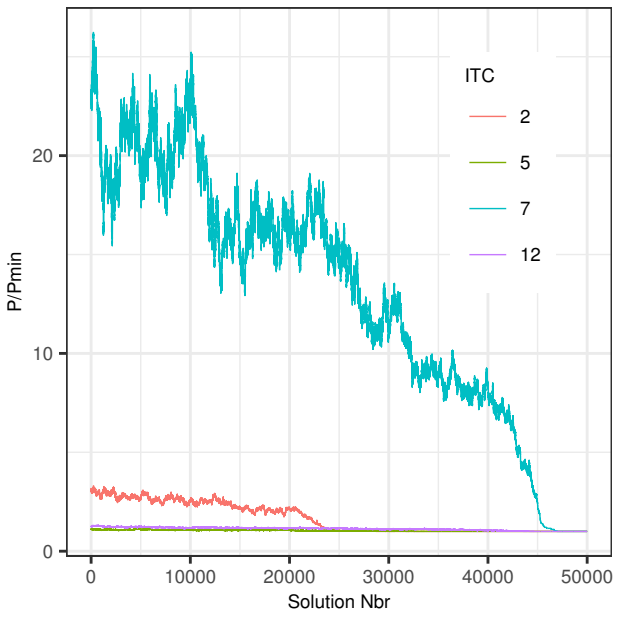

Fig. 1: Penalty convergence comparison for the solutions used in $N_i^{50,1}$ networks for $i = 2, 5, 7, 12$

neighbors of a solution provide statistically significant estimators of robustness of that solution. Among these metrics the ones that measure diversity of the neighbors stand out to be more consistently useful.

Identifying robust solutions is an important concern for many combinatorial optimization problems. Search heuristics expend significant computational effort, finding a large number of solutions, in order to find an optimal or near-optimal solution, and any potential valuable information that could be driven from those solutions regarding robustness is lost when those solutions are thrown away. The approach presented here is a first attempt to capture information from these solutions.

Since SA is a local search-based metaheuristic, the observations here are likely to be relevant for local search, but further research into other local search algorithms could be done to test sensitivity of the findings to different local search algorithms. The results associated with instance ITC7 suggests insufficient search of the parts of the solution space with high quality solutions, could reduce the effectiveness of the network metrics, so different local search algorithms or modified versions of the SA algorithm could improve the performance in such cases. Furthermore, it could be interesting to see if sampling solutions more widely provides better insights. Thus, one might look into other heuristic search strategies such as population-based evolutionary computation and swarm optimization. Finally, similar algorithmic approaches should also be tested on other combinatorial optimization problems.

REFERENCES

[1] K. M. Malan and A. P. Engelbrecht, "A survey of techniques for characterising fitness landscapes and some possible ways forward," Inform. Sciences, vol. 241, pp. 148–163, 2013.

[2] T. Hu, J. L. Payne, W. Banzhaf, and J. H. Moore, "Robustness, evolvability, and accessibility in linear genetic programming," Lect. Notes Comput. Sc., vol. 6621, pp. 13–24, 2011.

[3] W. Herroelen and R. Leus, "Robust and reactive project scheduling: a review and classification of procedures," Int. J. Prod. Res., vol. 42, pp. 1599-1620, 2004.

[4] P. F. Stadler, "Fitness landscapes," Lect. Notes Phys., vol. 585, pp.183–204, 2002.

[5] S. Kauffman and S. Levin, "Towards a general theory of adaptive walks on rugged landscapes," J. Theor. Biol., vol. 128, pp.11–45, 1987.

[6] G. Ochoa, R. Qu, and E. K. Burke, "Analyzing the landscape of a graph based hyper-heuristic for timetabling problems," Proc. 11th Ann. Conf. Genetic Evol. Comput. [GECCO '09], pp. 341–348, Montreal, Québec, Canada, 2009.

[7] J. Garnier and L. Kallel, "How to detect all maxima of a function," in: Theoretical Aspects of Evolutionary Computing, L. Kallel, B. Naudts, and A. Roger,s Eds. Berlin Heidelberg: Springer, 2001, pp.343–370.

[8] D. C. Porumbel, J. K. Hao, and P. Kuntz, "A search space "cartography" for guiding graph coloring heuristics," Comput. Oper. Res., vol. 37, pp. 769–778, 2010.

[9] B. McCollum, et al., "Setting the research agenda in automated timetabling: The second international timetabling competition," INFORMS J. Comput., vol. 22, pp. 120–130, 2010.

[10] A. E. Phillips, C. G. Walker, M. Ehrgott, and D. M. Ryan, "Integer programming for minimal perturbation problems in university course timetabling," Ann. Oper. Res., vol. 252, pp. 283–304, 2017.

[11] C. Akkan, A. Gülcü, and Z. Kuş, "Minimum penalty perturbation heuristics for curriculum-based timetables subject to multiple disruptions", unpublished.

[12] C. Akkan and A. Gülcü, "A bi-criteria hybrid Genetic Algorithm with robustness objective for the course timetabling problem", Comput. Oper. Res., vol. 90, pp. 22–32, 2018.

[13] A. J. Kleywegt, A. Shapiro, and T. Homem-de-Mello, "The sample average approximation method for stochastic discrete optimization," SIAM J. Optimiz., vol. 12, pp. 479–502, 2002.

[14] K. I. Smith, R. M. Everson, and J. E. Fieldsend, "Dominance measures for multi-objective simulated annealing," Proc. IEEE Congress Evol. Comput., vol. 1, pp.23-30, 2004.

[15] 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, 2016.

[16] A. Bettinelli, V. Cacchiani, R. Roberti, P. Toth, "An overview of curriculum-based course timetabling," TOP, vol. 23, iss. 2, pp. 313–349, 2015.

[17] I. Moser, M. Gheorghita, and A. Aleti, "Identifying features of fitness landscapes and relating them to problem difficulty," Evol. Comput., vol. 25, pp. 407–437, 2017.

[18] L. Vanneschi, Y. Pirola, and P. Collard, "A quantitative study of neutrality in GP boolean landscapes," Genetic Evol. Comput. Conf. [GECCO '06], pp. 895–902, 2006.

[19] L. Altenberg,"The evolution of evolvability in genetic programming," in: Advances in Genetic Programming, K. E. Kinnear Jr, Ed., MIT Press, 2004, pp. 47–74.

[20] T. Jones and S. Forrest, "Fitness distance correlation as a measure of problem difficulty for genetic algorithms," Proc. 6th Int. Conf. Genetic Alg., pp. 84–192, 1995.

[21] L. Altenberg, "NK fitness landscapes," Handbook of Evolutionary Computation, Chp. B2.7.2, Oxford Univ. Press, 1997.

[22] P. Merz, "Advanced fitness landscape analysis and the performance of memetic algorithms," Evol. Comput., vol. 12, pp. 303–325, 2004.

[23] C. M. Reidys, M. Christian, P. F. Stadler, "Neutrality in fitness landscapes," Appl. Math. Comput., vol. 117, pp. 321–350, 2001.

[24] S. Ceschia, L. Di Gaspero, and A. Schaerf, "Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrollment course timetabling problem," Comput. Oper. Res., vol. 39, pp. 1615–1624, 2012.

[25] A. Gülcü and C. Akkan, "Robust university course timetabling problem subject to single and multiple disruptions," Eur. J. Oper. Res., vol. 283, pp.630-646, 2020.

[26] A. Bonutti, F. De Cesco, L. Di Gaspero, and A. Schaerf, "Benchmarking curriculum-based course timetabling: formulations, data formats, instances, validation, visualization, and results," Ann. Oper. Res., vol. 194, iss. 1, pp. 59–70, 2012.