Solving High School Timetabling Problems Worldwide Using Selection Hyper-heuristics

Leena N. Ahmed, Ender Özcan, Ahmed Kheiri[1]

ASAP Research Group
University of Nottingham, School of Computer Science
Jubilee Campus, Wollaton Road, Nottingham, NG8 1BB, UK
email: {itxln1, Ender.Ozcan, psxak1}@nottingham.ac.uk

Abstract

High school timetabling is one of those recurring NP-hard real-world combinatorial optimisation problems that has to be dealt with by many educational institutions periodically, and so has been of interest to practitioners and researchers. Solving a high school timetabling problem requires scheduling of resources and events into time slots subject to a set of constraints. Recently, an international competition, referred to as ITC 2011 was organised to determine the state-of-the-art approach for high school timetabling. The problem instances, obtained from eight different countries across the world used in this competition became a benchmark for further research in the field. Selection hyper-heuristics are general-purpose improvement methodologies that control/mix a given set of low level heuristics during the search process. In this study, we evaluate the performance of a range of selection hyper-heuristics combining different reusable components for high school timetabling. The empirical results show the success of the approach which embeds an adaptive great-deluge move acceptance method on the ITC 2011 benchmark instances. This selection hyper-heuristic ranks the second among the previously proposed approaches including the ones competed at ITC 2011.

Keywords: Adaptive Operator Selection, Adaptive Move Acceptance, Great

[1]Ahmed Kheiri (a.kheiri@exeter.ac.uk) is, now, an Associate Research Fellow at the College of Engineering, Mathematics and Physical Sciences, University of Exeter.

Deluge, Combinatorial Optimisation, Constraint Satisfaction, Educational Timetabling

1. Introduction

Educational timetabling problem is classified as one of the hard computational problems (Broder, 1964), which are of interest to the researchers and practitioners from the fields of operational research and artificial intelligence. Educational timetabling problem has many variants including examination timetabling, university course timetabling and high school timetabling (Pillay, 2010b). The focus of this work is on high school timetabling.

The solution to high school timetabling problem requires the scheduling of events, such as courses and classes, and resources, such as teachers and classrooms to a number of specific time slots subject to a set of *hard* and *soft* constraints. The hard constraints must be satisfied, and the soft constraints represent preferences. A feasible solution to a given problem is the solution that satisfies all the hard constraints. High school timetabling is a well-known NP-hard combinatorial optimisation problem (de Werra, 1997; Even et al., 1976) recurring at many educational institutes. There are many variants of high school timetabling problem and they mainly differ due to many reasons, such as the educational system in a given country.

High school timetabling was subject of a recent challenge, the third International Timetabling Competition (ITC 2011) (Post et al., 2013), to encourage researchers and practitioners to deal with the real-world complexities of the high school timetabling problem without any simplification and support development of automated state-of-the-art methods for high school timetabling. Those real-world problem instances obtained across the world became a benchmark after the competition. Many different approaches have been proposed, each solving a particular problem, including simulated annealing (da Fonseca et al., 2014; Zhang et al., 2010), tabu search (Bello et al., 2008) and evolutionary algorithm (Shambour et al., 2013; Domrös and Homberger, 2012; Raghavjee and Pillay,

2012). More on high school timetabling can be found in (Pillay, 2010a, 2012).

Hyper-heuristics are general purpose solution methodologies which perform search over the space of heuristics rather than the solutions to solve hard computational problems (Burke et al., 2013). There are two general classes of hyper-heuristics identified in the scientific literature; high level methodologies that *generate* or *select* low level heuristics. The latter class is the focus of this study (Burke et al., 2010). Bilgin et al. (2007) argued that the performance of a selection hyper-heuristic varies depending on the choice of its components. In this study, fifteen hyper-heuristics combining five different heuristic selection components with three different move acceptance components are investigated for high school timetabling using the ITC 2011 benchmark. The performance of the best selection hyper-heuristic is analysed further and compared to the previously proposed approaches including the ones competed at ITC 2011.

Section 2 overviews selection hyper-heuristics. Section 3 describes the third International Timetabling Competition and selection hyper-heuristic components that are employed for solving the high school timetabling problem from the competition. Section 4 provides the empirical results. Finally, Section 5 presents the conclusions and remarks.

2. An Overview of Selection Hyper-heuristics

Heuristics are rule-of-thumb methods designed for a specific computationally difficult problem and often cannot be reused to solve another problem. This observation is also valid for meta-heuristics implemented for a specific problem. Although there are many successful examples of (meta-)heuristics capable of solving instances from a particular domain in the literature, their design require extensive knowledge about the relevant problem domain (Bilgin et al., 2007). On the other hand, hyper-heuristics have emerged as reusable general-purpose search methodologies with reusable components that can be applied to a wide range of problems (Cowling et al., 2001). The foundation of the current studies on hyper heuristics dates back to the early 1960s. Fisher and Thompson (1963)

stated that combining scheduling rules in production scheduling would make a great improvement than using them individually. Their study can be credited for putting the initial ideas forward and leading to the succeeding studies on hyper-heuristics. Cowling et al. (2001) initially defined hyper-heuristics as "heuristics to choose heuristics". The authors claimed that hyper-heuristics operate at a higher level of abstraction than meta-heuristics.

There are two main types of hyper-heuristics (Burke et al., 2010): (i) methodologies to *select* heuristics and (ii) methodologies to *generate* new heuristics. The former class, also known as selection hyper-heuristics, is the focus of this study. Often, selection hyper-heuristics operate on a single-point based search framework which has two common consecutive stages, *heuristic selection* and *move acceptance* as identified by Özcan et al. (2008). An initial solution is improved iteratively through passing into these two stages, consecutively. The heuristic selection selects a heuristic from a set of pre-defined low level heuristics and generates a new solution; the move acceptance decides whether to accept or reject the new solution. If the solution is accepted, it replaces the original solution. The process iterates until a set of termination criteria is satisfied as demonstrated in Figure 1.

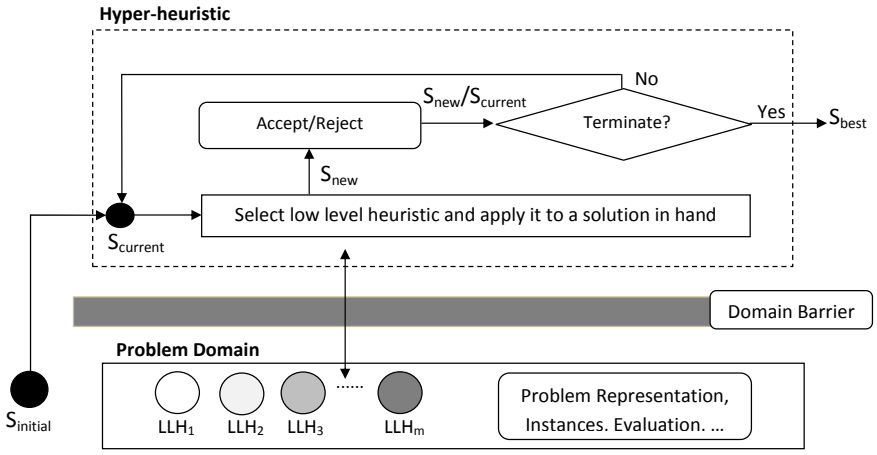


Figure 1: Demonstration of a generic selection hyper-heuristic framework

In most of the selection hyper-heuristics, a domain barrier is featured (see Figure 1). The domain barrier is a filter which prevents any problem specific information to be passed to the hyper-heuristic level (Burke et al., 2013), thus it allows the reusability of the selection hyper-heuristic components. Although the selection and the move acceptance methods are general and reusable, Bilgin et al. (2007) stated that using different combinations of the two methods yields to a different performance.

There is a large number of studies on heuristic selection and move acceptance methods used within selection hyper-heuristics (Burke et al., 2013). In here, we describe the relevant selection hyper-heuristic components which have been used in this study. Most of the simple selection methods were tested by Cowling et al. (2001), initially. Simple Random (SR) uses a uniform probability distribution to randomly select one of the heuristics at each step. Random Descent (RD) performs similarly to SR except that it applies the selected heuristic repeatedly until there is no further improvement. Random Permutation (RP) generates an initial permutation of the low level heuristics and applies one heuristic at a time from that permutation at each step, sequentially. Random Permutation Descent (RPD) orders the low level heuristics randomly similar to RP, but operates in the same way as RD while applying a chosen low level heuristic.

Choice Function (CF) scores the low level heuristics based on a combination of three different measures and the heuristic with the highest score is chosen at each step. The first measure f_1 is calculated based on the previous performance of each low level heuristic according to the following formula:

$$f_1(LLH_i) = \sum_n \alpha^{n-1} \frac{I_n(LLH_i)}{T_n(LLH_i)} \tag{1}$$

where $I_n(LLH_i)$ is the evaluation function change, $T_n(LLH_i)$ is the time taken to apply LLH_i in n previous invocation, and α is a value between 0 and 1. The second measure f_2 records the pair-wise dependencies between the heuristics. When a heuristic LLH_i gets invoked right after the invocation of heuristic

LLH_k, the value of f_2 is measured using the following formula:

$$f_2(LLH_k, LLH_i) = \sum_n \beta^{n-1} \frac{I_n(LLH_k, LLH_i)}{T_n(LLH_k, LLH_i)} \tag{2}$$

where $I_n(LLH_k, LLH_i)$ is the evaluation function change, $T_n(LLH_k, LLH_i)$ is the time taken to apply LLH_i following LLH_k in n previous invocation, and β is a value between 0 and 1. The third measure f_3 is the time passed ($\tau(LLH_i)$) since the last time the heuristic LLH_i was invoked.

$$f_3(LLH_i) = \tau(LLH_i) \tag{3}$$

The choice function ranks the heuristics based on a score given to each heuristic. This score is calculated using the three above measures with the following formula:

$$F(LLH_i) = \alpha f_1(LLH_i) + \beta f_2(LLH_k, LLH_i) + \delta f_3(LLH_i) \tag{4}$$

where α and β weight the first two measures to give intensification to the search process of the heuristic, while δ weights f_3 to give diversification. More on choice function and its variants can be found in (Drake et al., 2012).

There are a number of deterministic and non-deterministic acceptance methods that are used as a move acceptance component within selection hyper-heuristics. Cowling et al. (2001) described some of the simple deterministic methods, including accepting all moves (AM) criterion which accepts all the generated solutions, accepting only improving moves (OI) which accepts only the improved solutions, and accepting improve or equal moves (IE) which accepts only the non-worsening solutions. Simulated Annealing (SA) move acceptance was used in (Bai and Kendall, 2005; Bai et al., 2007; Bilgin et al., 2007). Simulated annealing accepts the worsening solutions with a probability given by the following equation:

$$p_t = e^{-\frac{\Delta f}{\Delta F(1-\frac{t}{T})}} \tag{5}$$

where Δf is the change in the evaluation function at time (step) t, T is the time limit (maximum number of steps), ΔF is the range for the maximum change in the evaluation function after applying a heuristic.

Kendall and Mohamad (2004) utilised Great Deluge (GD) as a move acceptance strategy for channel assignment problem in the industry of cellular communication. Great deluge move acceptance accepts a worsening solution if the cost of that solution is better than or equal to a specific cost value called the level at each step. For the initial level, the value of the first generated candidate solution is used, and the level at each step is then updated with a linear rate using the following formula:

$$\tau_t = f_0 + \Delta F \times (1 - \frac{t}{T}) \tag{6}$$

where τ_t is the value of the threshold level at time (step) t, T is the time limit (maximum number of steps), ΔF is the expected range for the maximum change in the evaluation function, and f_0 is the final cost value.

Özcan et al. (2006) demonstrate and compare between four different selection hyper-heuristic frameworks discriminating between *mutational* and *hill climbing* low level heuristics. The mutational low level heuristics perturb a given solution mostly at random and act as a diversification component which enables the search process to explore the other regions potentially leading to high quality solutions. The hill climbing heuristics always produce non-worsening solutions. Figure 1 is the traditional framework which uses all low level heuristics without any discrimination. The experimental results on a set of benchmark functions showed that frameworks using mutational heuristics first and then applying a hill climbing similar to the process in iterated local search (Özcan et al., 2006) deliver better performance in the overall. More on hyper-heuristics can be found in (Burke et al., 2013, 2003; Chakhlevitch and Cowling, 2008; Ross, 2005; Özcan et al., 2008).

3. Selection Hyper-heuristics for High School Timetabling

3.1. The Third International Timetabling Competition - ITC 2011

The third international timetabling competition (ITC 2011[2]) (Post et al., 2013) was organised by the Centre of Telematics and Information Technology at the University of Twente in Netherlands, to determine the state-of-the-art approach among the modern approaches for the high school timetabling problem, allow researchers to try their techniques in real-world practical problem, and to encourage the researchers and practitioners for further development of algorithms in the area of high school timetabling. The competitors were given an ANSI C library[3], referred to as KHE, which was designed by Jeff Kingston for implementing their algorithms to solve high school timetabling problems. The competition consisted of three rounds. Since the time limit for the first and third rounds of the competition was in months, the focus of this study is the second round of ITC 2011 in which algorithms were required to operate as time-contract algorithms, hence they had to terminate with a solution in a given maximum amount of time. In the second round, the time limit was defined as 1000 nominal seconds based on the organisers' machine. A tool was provided for benchmarking of user machines. The competitors' solvers were run by the organisers for 10 times with different random seeds each for 1000 seconds on 18 hidden instances. The result for each run was ranked, and all the ranks were averaged to determine the winner.

The problem instances for the competition were obtained from different educational institutions across the world. A standard format for the definition of the instances based on XML schema, referred to as XHSTT (XML for high school timetabling) was used (Post et al., 2012, 2013), supported by KHE. The problem instances of the ITC 2011 consisted of four components (Post et al., 2013). The first component defines the instance times, which are individual units of times during which the events take place. The second component is the

[2]ITC 2011 website: http://www.utwente.nl/ctit/hstt/

[3]`http://sydney.edu.au/engineering/it/~jeff/khe/`

resources which are entities that attend the events such as teachers, students, and rooms. The third component is the events which represent the meeting between resources. Each event has a specific time, duration and any number of resources. Finally the fourth component is the constraints which are conditions an ideal solution should satisfy. The ITC 2011 instances contain 15 types of constraints: assign resource, assign time, split events, distribute split events, prefer resources, prefer times, avoid split assignments, spread events, link events, avoid clashes, avoid unavailable times, limit idle times, cluster busy times, limit busy times, limit workload. In this study, the selection hyper-heuristics are tested on the instances used in the second round of the competition. Table 1 summarises the characteristics of the instances used in that round.

Table 1: Characteristics of round 2 instances used during ITC 2011 competition

Instance-Country	Times	Teachers	Rooms	Classes	Students	Event	Duration
Instance2-Brazil	25	14		6		63	150
Instance3-Brazil	25	16		8		69	200
Instance4-Brazil	25	23		12		127	300
Instance6-Brazil	25	30		14		140	350
ElementarySchool-Finland	35	22	21	60		291	445
SecondarySchool2-Finland	40	22	21	36		469	566
Aigio 1st HS 2010-Greece	35	37		208		283	532
WesternGreeceUni3-Greece	35	19		6		210	210
WesternGreeceUni4-Greece	35	19		12		262	262
WesternGreeceUni5-Greece	35	18		6		184	184
Instance4-Italy	36	61		38		748	1101
Instance1-Kosovo	62	101		63		809	1912
Kottenpark2003-Netherlands	38	75	41	18	453	1156	1203
Kottenpark2005A-Netherlands	37	78	42	26	498	1235	1272
Kottenpark2008-Netherlands	40	81	11	34		1047	1118
Kottenpark2009-Netherlands	38	93	53	48		1166	1301
Woodlands2009-South Africa	42	40				278	1353
School-Spain	35	66	4	21		225	439

Each constraint contains a Boolean variable to indicate whether the constraint is hard or soft. The penalty for violating a hard constraint is much higher than the soft constraint according to the competition rules. The quality

(degree of violations) of a solution for a given problem instance is computed using a minimising evaluation function (cost) which contains two components: *feasibility*, and *preferences* (objective). They are calculated using weighted sum of hard and soft constraint violations for a given solution, respectively. The weights are defined as input for each problem instance. To represent the quality of a given solution, the two values of infeasibility and objective are concatenated in the form: infeasibility-value.objective-value, using "sufficient" number of digits in the objective part. For example a solution of 12.000032, indicates an infeasibility value of 12, and objective value of 32. A solution is considered better than the other if it has a smaller infeasibility value or the same infeasibility but less objective value (Post et al., 2013).

In the second round of ITC 2011, four teams submitted their solvers. The team HySST (Kheiri et al., to appear) employed a multi-stage hyper-heuristic approach that operates on a set of mutational and hill climbing heuristics, which operate on a candidate solution with a direct representation. The proposed approach switches between exploration and exploitation stages automatically and use appropriate heuristics at each stage. Moreover, this solver embeds an adaptive threshold move acceptance, controlling its parameter setting during the search process enabling partial restarts whenever necessary. The HySST solver has some system parameters which are tuned and fixed for high school timetabling. The team HFT (Domrös and Homberger, 2012) designed an evolutionary strategy which uses only mutation as a genetic operator as their solver. The main characteristic of this solver is that it uses an indirect representation, encoding solutions using a permutation of sub-events. Moreover, the HFT solver uses a population size of 1, accepting improving moves only as the replacement strategy. At each evolutionary cycle, the candidate solution in hand gets decoded and used to construct a complete new timetable, which is followed by evaluation and replacement. The proposed algorithm can be considered to be a basic random mutation hill climbing algorithm in the overall. The team Lectio (Sørensen et al., 2012) used an approach based on adaptive large neighbourhood search which passes through three main stages. The first stage determines how

many variables gets unassigned. The following stage uses an adaptive strategy deciding which remove and insertion (reassign) type of move operators to invoke successively, producing a new solution. Then finally, a simulated annealing with reheating method decides to accept or reject that new solution. Lectio applied parameter tuning using irace tool on nine system parameters of the solver for an improved performance. The team GOAL (da Fonseca et al., 2014) proposed a three stage approach using two different meta-heuristics for high school timetabling. The first stage constructs an initial solution using the KHE library. Then simulated annealing with reheating and iterated local search stages are employed respectively to improve upon that initially generated solution. The GOAL solver contains seven different neighbourhood operators, two of them being representative of 'large' neighbourhoods, while the remaining are fairly simple 'small' moves, such as swap or move events. The simulated annealing with reheating utilises a subset of six neighbourhood operators, while iterated local search utilises two of them. A neighbour operator is chosen based on a prefixed probability and applied to a solution in hand at any step.

The results of the second round of the competition revealed that team GOAL is the winner, team HySST ranked second, Lectio third and HFT fourth. Soon after the competition, Kalender et al. (2013) applied a hyper-heuristic using a greedy-gradient approach for selection and simulated annealing for move acceptance and applied it to the round 2 instances of ITC 2011. The greedy-gradient is a learning heuristic selection method that selects heuristics based on their scores. The results showed the success of the approach performing better than HySST using the same ranking method as used in the second round of the competition.

3.2. Solution Method

The same problem domain layer in the framework proposed by Kheiri et al. (to appear) is used for implementing a range of combinations of hyper-heuristic components. The initial solutions are constructed using the heuristic provided with the KHE software library. The selection hyper-heuristics are then used to

mix a set of nine low level heuristics, including seven mutational and two hill climbing heuristics as briefly described below:

- **MH**$_1$ consists of two independent perturbation operators which are invoked successively. The first operator is invoked with a probability of 1% and splits a randomly chosen event taking longer than 1 period into two events whose durations sum up to the duration of the original event. Then the second operator is invoked, exchanging the start time of two randomly chosen events regardless of their duration and whether this exchange causes any overlaps afterwards. For example, this heuristic could choose a Mathematics meeting with a duration of two hours, splitting it into one hour long two separate meetings and then swap the start time of Geography and Biology meetings. Assuming the special case that Geography with a duration of 2 time slots starts on Tuesday at the first time slot and Biology with a duration of 3 time slots starts on Tuesday at the third time slot, this heuristic will swap the start time of both meetings even though they overlap.

- **MH**$_2$ chooses an event randomly, and reschedules it to a random time slot. For example, assuming that Biology meeting taking place on Tuesday at the first time slot is chosen, this heuristic could reschedule this meeting to the last time slot on Thursday.

- **MH**$_3$ exchanges the start time of two randomly selected events resolving overlaps that could occur after this operation. If the two randomly selected events have the same duration, then the classical exchange operation will be performed. The difference between the exchange operation in MH$_1$ and MH$_3$ becomes apparent only when swapping the start time of two successive events with different durations. For example, assuming the special case that Geography with a duration of 2 time slots starts on Tuesday at the first time slot and Biology with a duration of 3 time slots starts on Tuesday at the third time slot, MH$_3$ would move the start time of Geography to the fourth time slot on Tuesday.

- **MH**$_4$ chooses a resource randomly assigned to an event, and reassigns it to another event. For example if Room1 is assigned to the History meeting, after applying this heuristic, Room1 could be assigned to the Mathematics meeting.

- **MH**$_5$ randomly swaps two resources. For example, assuming that the Biology meeting is assigned to Room1, the Geography meeting is assigned to Room2, and those resources are chosen, after applying this heuristic, the Biology meeting gets assigned to Room2, while the Geography meeting gets assigned to Room1.

- **MH**$_6$ randomly chooses an event and an associated resource, then reassigns a random resource to the event. For example, assuming that Geography meeting is chosen and Teacher2 is assigned as the teacher of that meeting, after applying this heuristic, Teacher5 could become the teacher of that Geography meeting.

- **MH**$_7$ merges separate, but contiguous events of the same type. For example, assuming that Geography with a duration of two time slots is assigned to the first time slot on Thursday, and another Geography meeting with a duration of one period is assigned to the third time slot on Thursday, after applying this heuristic, the two classes are merged into a single class with a duration of three time slots starting from the first time slot on Thursday.

- **HC**$_1$ merges events to reduce the cost of the solution by employing a first improvement hill climbing operator

- **HC**$_2$ makes moves based on ejection chains to reduce the cost of the solution by changing the assignments of the resources

Both hill climbers make their moves according to a specific constraint, hoping the solution improves upon the other types of constraints. This could produce a net worsening in the final cost, but such worsening moves are rejected. More on those low level heuristics can be found in (Kheiri et al., to appear).

The goal of this work is to compare the performance of different selection hyper-heuristics embedding different reusable heuristic selection and move acceptance methods and report the best performing approach unlike the work in (Kheiri et al., to appear). Each hyper-heuristic component exhibit different characteristics some with learning and some without learning; some are adaptive methods and some are not. We investigate the performance of 15 selection hyper-heuristics, formed by combining each selection method in {SR, RP, RD, RPD, CF} with each acceptance method in {IE, SA, GD} over 18 problem instances from the second round of the ITC 2011 competition.

SA and GD are adaptive move acceptance methods which are implemented different than the versions described in Section 2. The ΔF value in the simulated annealing and great deluge move acceptance methods is set to 0.01% of the cost of the best solution in hand, and to 1% if the best solution violates only soft constraints, as suggested in (Kalender et al., 2013). The f_0 value in great deluge is set to 0.001% of the cost of the best solution in hand, and to 0.1% if the best solution violates only soft constraints.

4. Computational Results

The experiments are conducted on the second round problem instances of the ITC 2011 competition. A total number of fifteen selection hyper-heuristic methods are investigated as described in Section 3. Each method is applied to the same set of eighteen instances taking into account the rules of the ITC 2011 competition, that is, each method is run for ten trials with a time limit of 1000 nominal seconds for each instance. A benchmarking software tool provided at the ITC 2011 website is used to obtain the equivalent time limit on our local machines. The selection hyper-heuristics are evaluated with the aim of determining the best algorithm that delivers the high quality solutions to the high school timetabling problem instances. Then the performance of the best hyper-heuristic is compared to some previously proposed approaches.

4.1. Comparison of the Selection Hyper-heuristics

Table 2 summarises the results from fifteen selection hyper-heuristics, each combining a heuristic selection method from {SR, RP, RD, RPD, CF} with a move acceptance from {IE, SA, GD} over the ITC 2011 problem instances. Each entry in the table provides the number of instances for which the corresponding selection hyper-heuristic produces the best in terms of best-of-run or average best, including ties, over the 10 trials. The table also gives the score for each hyper-heuristic using the ranking strategy utilised in the second round of the ITC 2011 competition. From Table 2, RPGD and RDSA generate the best average in three instances. RDGD generates the best and the minimum cost values in five instances including four draws. RDIE also obtains the best results in five instances including 3 draws. The ranking results put the selection methods {SR, RP, RD, RPD} combined with {GD} in the top of the fifteen selection hyper-heuristics that were tested. The results also show that the heuristic selection methods with no learning (i.e., SR, RP) or learning with the shortest memory (i.e., RD) perform better than the CF learning heuristic selection method regardless of the move acceptance. RPGD and SRGD are the best approaches based on their scores. On the other hand, considering the average results, RPGD performs slightly better than SRGD producing the best average cost on three instances, while SRGD is successful on one instance. Hence, the performance of top ranking great deluge based selection hyper-heuristics are compared further.

Table 3 summarises the results. The Mann-Whitney-Wilcoxon test is used as a statistical test to perform pairwise comparison of results (costs) of 10 runs from RPGD versus each hyper-heuristic in {SRGD, RDGD, RPDGD} at 95% confidence level. Indeed, RPGD performs better than the other algorithms on average considering that the number of instances for which it produces the best average results is six (in bold in Table 3), which is more than any of the other hyper-heuristic. The average performance difference between RPGD and the other selection hyper-heuristics is not statistically significant for almost all instances, as confirmed by the Mann-Whitney-Wilcoxon test. Although Table 2 shows that SRGD and RPGD has the same scores, RPGD performs better than

SRGD on twelve instances and this performance difference is statistically significant on the Woodlands2009-South Africa instance (Table 3). Hence, RPGD is taken under consideration for further analysis and performance comparison to previously proposed approaches.

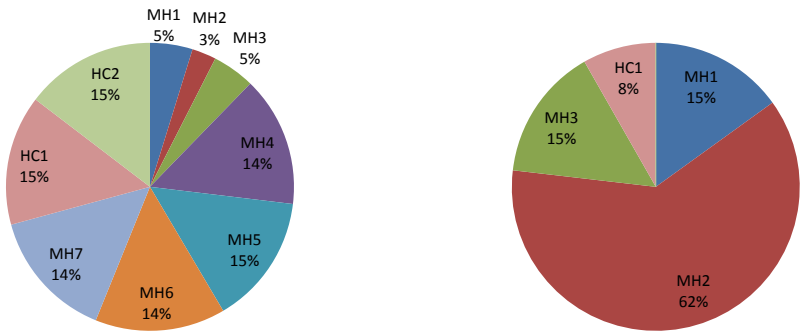
4.2. An Analysis of RPGD

RPGD creates a random permutation of low level heuristics and applies each one of them on the solution in hand one by one. For example, given five heuristics, a random permutation of low level heuristics could be $<LLH_3$, LLH_2, LLH_4, LLH_5, $LLH_1>$. A circular list for the permutation is employed and at each step, next low level heuristic from the list is chosen. For example, assuming $<LLH_3$, LLH_2, LLH_4, LLH_5, $LLH_1>$, after the invocation of LLH_1, LLH_3 is chosen as the next heuristic. In between the heuristic invocations, the adaptive great deluge move acceptance is used for accepting or rejecting a new solution. Mixing all low level heuristics regardless of their nature in this manner, implicitly generates an algorithm similar to iterated local search algorithm which is known to be an effective solver for combinatorial optimisation problems (Lourenço et al., 2010). RP does not employ perturbation and local search as fixed order single step/stage processes. The perturbation and local search components are fixed as the permutation of low level heuristics is decided randomly. Hence, RPGD can be considered as a multi-stage hyper-heuristic in which number of stages is decided randomly depending on the nature of successive low level heuristics in the random permutation. Exploration of the search space is performed using a (set of) mutational heuristic(s) at a stage while exploitation takes place by the help of a (set of) hill climbing heuristic(s), invoked afterwards. In the next exploration or exploitation process, a (set of) different heuristic(s) is utilised from the list. For example, given five low level heuristics, where LLH_1 and LLH_4 are hill climbers, and LLH_2, LLH_3 and LLH_5 are mutational heuristics, the random permutation $<LLH_3$, LLH_2, LLH_4, LLH_5, $LLH_1>$ automatically creates four stages. In the first stage, LLH_3 and LLH_2 are used for perturbation (exploration), while in the following stage LLH_4 is

used for local search (exploitation). Then, similarly, LLH_5 is used for perturbation in the third stage, while LLH_1 is used as a local search component in the following stage. This exploration and exploitation cycle repeats itself under this fixed setting while solving a given instance until the hyper-heuristic terminates.

The experimental results indicate that the contribution of each low level heuristic varies for the improvement of an initial solution within the given time limit. The *utilisation rate* of a low level heuristic is the ratio of the total number of times a low level heuristic is invoked, to the total number of low level heuristics invocations during the search process (Özcan et al., 2008). The utilisation rate is obvious for each low level heuristic for the RP heuristic selection. They are all equally used, but then again it does not mean that they contributed equally to improvement. Figure 2 depicts the improvement oriented average percentage utilisation rate for the low level heuristics over 10 runs considering only the total number of low level heuristics invocations that generated improvement for two selected sample problem instances of WesternGreeceUni5-Greece and Instance1-Kosovo. It has been observed that the hill climbing low level heuristics are more successful, resulting with a high utilisation rate for WesternGreeceUni5-Greece instance (Figure 2(a)). However, surprisingly, the mutational operators MH_4, MH_5, MH_6 and MH_7 generate improvement almost as much as the hill climbing operators. Moreover, the remaining mutational operators MH_1, MH_2 and MH_3 yield poorer performance when compared to them. MH_1-MH_3 are event oriented random perturbation operators and they modify a given solution randomly by, for example, swapping or changing the timing of events in the timetable, while MH_4-MH_7 are resource oriented operators (see Section 3). The analysis clearly show that the low level heuristics which perturb the resource allocations between events in the timetable are likely to result with an improved solution. Although MH_1-MH_3 do not seem to yield more improvements than other heuristics, they act as diversification components, potentially leading to better solutions and increasing the utilisation rate of the other low level heuristics, considering that GD allows worsening solutions. For the rest of the problem instances, similar phenomena are observed, except for the the Instance1-Kosovo. Figure 2(b) il-

lustrates the different behaviour of RPGD while solving this instance. Although all low level heuristics are invoked, only the event oriented low level heuristics, MH_1-MH_3 and HC_1 generate improving solutions during the search process.

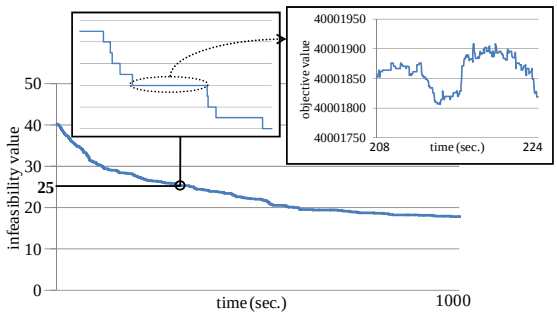


(a) (b)

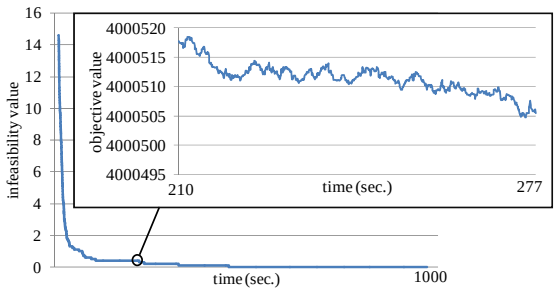
Figure 2: Average utilisation rate for the low level heuristics over 10 runs based on only improving moves for (a) WesternGreeceUni5-Greece, and (b) Instance1-Kosovo

Figure 3 illustrates the progress of the infeasibility value and partially objective value in time averaged over 10 runs on two selected problem instances, namely; Instance4-Brazil and WesternGreeceUni5-Greece, representative of the remaining instances. RPGD either makes a gradual and slow improvement as illustrated in Figure 3(a) or a large and rapid improvement as in Figure 3(b). In any case, a solution is improved continuously over time. There are certain stages during when the search process gets stuck at a local optimum. No change in the infeasibility value has been observed for a period of time, when RPGD works on repairing of the soft constraint violations. The use of adaptive strategy within the great deluge move acceptance method changing the threshold is helping the search process to jump to other potentially "good" regions in the search space yielding further improvements in time. Figure 3(a) shows that the degree of improvement that RPGD achieves is limited and still struggles resolving hard constraints within the given duration. In Figure 3(b), the sudden drop in the infeasibility value happens right at the beginning of the search process. After a while the infeasibility value reaches nearly to zero indicating that the solu-

tion is satisfying all the hard constraints. After this point onward during the search process, the improvement in the solution slows down while the algorithm attempts to repair the soft constraint violations.



(a)



(b)

Figure 3: Plots showing the improvement of cost in terms of infeasibility and objective values in time averaged over 10 runs for the instances (a) Instance4-Brazil, and (b) WesternGreeceUni5-Greece

4.3. Comparison of RPGD to the Best Known Approaches

The performance of RPGD algorithm is compared to the performance of the four finalists in the second round of ITC 2011 competition, GOAL, HySST, Lectio, and HFT. Additionally, the greedy-gradient simulated annealing (GGSA) approach proposed in (Kalender et al., 2013) is considered in the performance comparison. The scoring of each method is based on the same ranking strategy used in the second round of the ITC 2011 competition. Table 4 provides

the best cost values obtained by each approach in 10 runs for each instance. HySST and GGSA use the same selection hyper-heuristic framework. HySST cannot generate the best result on any of the instances among the algorithms being compared, while GGSA generates the best result on ElementarySchool-Finland and Kottenpark2003-Netherlands. RPGD wins on 9 instances against HySST and they tie on 2 instances, while it is the winner on 10 instances against GGSA. RPGD performs well on the problem instances from Greece, Italy and Netherlands and its performance is superior on most of the large problem instances. In the overall, GOAL still turns out to be the best approach for high school timetabling. RPGD obtains the new best known results on three instances: Instance1-Kosovo, Kottenpark2005A-Netherlands and Kottenpark2009-Netherlands. Overall, RPGD ranks the second with a score of 3.08 among the previously proposed algorithms. However, the difference between RPGD and the third approach (GGSA) is only 0.06.

5. Conclusion

The goal of hyper-heuristic research is to provide automated intelligent search methodologies that can be applied to a wide range of computationally hard problems. The theoretical work on such methodologies is limited. Lehre and Özcan (2013) recently demonstrated on some benchmark functions that mixing multiple move operators, or acceptance methods yield more efficient algorithms than using a single operator. In this study, a set of selection hyper-heuristics combining five different selection methods, with three move acceptance methods are experimented and their performance is analysed for high school timetabling. The results revealed that the selection method Random Permutation (RP) when combined with and adaptive Great Deluge (GD) move acceptance criterion performed better than the other selection hyper-heuristics and ranked the second comparing to some previously proposed methods.

The experimental results confirm that the choice of selection hyper-heuristic components influences its overall performance. The adaptation ability of move

acceptance component is also a crucial element in the overall performance of a selection hyper-heuristic. Adaptive move acceptance criteria perform better than the deterministic or probability based move acceptance, such as simulated annealing, under the single point based hyper-heuristic search framework for high school timetabling. Combining an adaptive heuristic selection method with an adaptive move acceptance does not necessarily result with a better performing selection hyper-heuristic. The results indeed show that the internal dynamics between adaptive components of a selection hyper-heuristic could cause the search process to get stuck at a local optimum during the search process. The reinforcement learning based adaptive heuristic selection method CF performs the worst when combined with the adaptive move acceptance method GD for high school timetabling, while RP, a heuristic selection method with no learning performs the best when combined with the same move acceptance method.

Both the HySST (Kheiri et al., to appear) and proposed solver contain adaptive threshold move acceptance, however GD adapts itself to changes better in the overall, generating better solutions to given instances. The move acceptance component of HySST is not as general as GD, since it uses a set of discrete threshold values which are tuned for high school timetabling and so it might not perform well on other problem domains. Moreover, the heuristic selection component of HySST is also not as general as RP and relies on the nature of the low level heuristics distinguishing between mutational and hill climbing heuristics. For example, if ruin and recreate or crossover type of operators are introduced, there is no strategy within the heuristic selection component of HySST to handle them properly. However, RP being a simple yet effective strategy can handle any type of operator.

A hyper-heuristic controls the mixing of low level heuristics and their parameter setting. The success of RP on timetabling is worth to consider in the future design of hyper-heuristics. Hence, we plan to apply this hyper-heuristic on other problem domains, but more importantly we plan to investigate into learning heuristic selection methods which orders chosen low level heuristics. Human design of such strategies could be an extremely difficult task, and so

data science techniques, such as machine learning (Asta and Özcan, 2015) or other metaheuristics, such as genetic programming Burke et al. (2009) can be embedded into hyper-heuristics, constructing such strategies automatically for improved performance.

The framework used during the experiments for high school timetabling is forward compatible, meaning that new hyper-heuristic components developed in the future can easily be tested on this problem domain. Moreover, new low level domain specific heuristics can be designed and best performing selection hyper-heuristics could be re-evaluated managing those low level heuristics. Moreover, if crossover operators are implemented as low level heuristics, then adaptive memetic algorithms or other memetic computing techniques (Neri and Cotta, 2012) could be utilised as population based approaches, further enabling the development of hybrid approaches using hyper-heuristics as local search components.

References

Asta, S., Özcan, E., 2015. A tensor-based selection hyper-heuristic for cross-domain heuristic search. Information Sciences 299, 412 – 432.

Bai, R., Burke, E.K., Kendall, G., McCollum, B., 2007. A simulated annealing hyper-heuristic methodology for flexible decision support. Technical Report Technical Report NOTTCS-TR-2007-8. School of CSiT, University of Nottingham.

Bai, R., Kendall, G., 2005. An investigation of automated planograms using a simulated annealing based hyper-heuristics, in: Ibaraki, T., Nonobe, K., Yagiura, M. (Eds.), Metaheuristics: Progress as Real Problem Solver. Springer, pp. 87–108.

Bello, G.S., Rangel, M.C., Boeres, M.C.S., 2008. An approach for the class/teacher timetabling problem, in: Proceedings of the 7th International

Conference on the Practice and Theory of Automated Timetabling (PATAT '08), pp. 1–6.

Bilgin, B., Özcan, E., Korkmaz, E.E., 2007. An experimental study on hyper-heuristics and exam scheduling, in: Burke, E.K., Rudová, H. (Eds.), Practice and Theory of Automated Timetabling VI, Springer Berlin Heidelberg. pp. 394–412.

Broder, S., 1964. Final examination scheduling. Communications of the ACM 7, 494–498.

Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Qu, R., 2013. Hyper-heuristics: a survey of the state of the art. Journal of the Operational Research Society 64, 1695–1724.

Burke, E.K., Hart, E., Kendall, G., Newall, J., Ross, P., Schulenburg, S., 2003. Hyper-heuristics: an emerging direction in modern search technology, in: Glover, F., Kochenberger, G. (Eds.), Handbook of Metaheuristics. Kluwer, pp. 457–474.

Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Woodward, J.R., 2010. A classification of hyper-heuristic approaches, in: Gendreau, M., Potvin, J.Y. (Eds.), Handbook of Metaheuristics. Springer US. volume 146 of *International Series in Operations Research and Management Science*, pp. 449–468.

Burke, E.K., Hyde, M.R., Kendall, G., Ochoa, G., Özcan, E., Woodward, J.R., 2009. Exploring hyper-heuristic methodologies with genetic programming, in: Kacprzyk, J., Jain, L.C., Mumford, C.L., Jain, L.C. (Eds.), Computational Intelligence. Springer Berlin Heidelberg. volume 1 of *Intelligent Systems Reference Library*, pp. 177–201.

Chakhlevitch, K., Cowling, P., 2008. Hyperheuristics: recent developments, in: Cotta, C., Sevaux, M., Sörensen, K. (Eds.), Adaptive and Multilevel

Metaheuristics. Springer Berlin Heidelberg. volume 136 of *Studies in Computational Intelligence*, pp. 3–29.

Cowling, P., Kendall, G., Soubeiga, E., 2001. A hyperheuristic approach to scheduling a sales summit, in: Burke, E., Erben, W. (Eds.), Practice and Theory of Automated Timetabling III. Springer Berlin Heidelberg. volume 2079 of *Lecture Notes in Computer Science*, pp. 176–190.

Domrös, J., Homberger, J., 2012. An evolutionary algorithm for high school timetabling, in: Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling (PATAT '12), pp. 485–488.

Drake, J.H., Özcan, E., Burke, E.K., 2012. An improved choice function heuristic selection for cross domain heuristic search, in: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (Eds.), Parallel Problem Solving From Nature (PPSN XII), Springer Berlin Heidelberg. pp. 307–316.

Even, S., Itai, A., Shamir, A., 1976. On the complexity of timetable and multicommodity flow problems. SIAM Journal on Computing 5, 691–703.

Fisher, H., Thompson, G.L., 1963. Probabilistic learning combinations of local job-shop scheduling rules, in: Muth, J.F., Thompson, G.L. (Eds.), Industrial Scheduling, Prentice-Hall, Inc, New Jersey. pp. 225–251.

da Fonseca, G.H.G., Santos, H.G., Toffolo, T.A.M., Brito, S.S., Souza, M.J.F., 2014. GOAL solver: a hybrid local search based solver for high school timetabling. Annals of Operations Research , 1–21.

Kalender, M., Kheiri, A., Özcan, E., Burke, E.K., 2013. A greedy gradient-simulated annealing selection hyper-heuristic. Soft Computing 17, 2279–2292.

Kendall, G., Mohamad, M., 2004. Channel assignment optimisation using a hyper-heuristic, in: Proceedings of the 2004 IEEE Conference on Cybernetic and Intelligent Systems (CIS2004), Singapore. pp. 790–795.

Kheiri, A., Özcan, E., Parkes, A.J., to appear. A stochastic local search algorithm with adaptive acceptance for high-school timetabling. Annals of Operations Research .

Lehre, P.K., Özcan, E., 2013. A runtime analysis of simple hyper-heuristics: to mix or not to mix operators, in: Proceedings of the Twelfth Workshop on Foundations of Genetic Algorithms (FOGA XII '13), ACM, New York, NY, USA. pp. 97–104.

Lourenço, H.R., Martin, O.C., Stützle, T., 2010. Iterated local search: framework and applications, in: Gendreau, M., Potvin, J.Y. (Eds.), Handbook of Metaheuristics. Springer US. volume 146 of *International Series in Operations Research and Management Science*, pp. 363–397.

Neri, F., Cotta, C., 2012. Memetic algorithms and memeting computing optimization: a literature review. Swarm and Evolutionary Computation 2, 1–14.

Özcan, E., Bilgin, B., Korkmaz, E.E., 2006. Hill climbers and mutational heuristics in hyperheuristics, in: Runarsson, T.P., Beyer, H.G., Burke, E., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (Eds.), Parallel Problem Solving from Nature (PPSN IX). Springer Berlin Heidelberg. volume 4193 of *Lecture Notes in Computer Science*, pp. 202–211.

Özcan, E., Bilgin, B., Korkmaz, E.E., 2008. A comprehensive analysis of hyper-heuristics. Intelligent Data Analysis 12, 3–23.

Pillay, N., 2010a. An overview of school timetabling research, in: Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling (PATAT '10), pp. 321–335.

Pillay, N., 2010b. A study into the use of hyper-heuristics to solve the school timetabling problem, in: Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists (SAICSIT '10), ACM, New York, NY, USA. pp. 258–264.

Pillay, N., 2012. Hyper-heuristics for educational timetabling, in: Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling (PATAT '12), pp. 316–340.

Post, G., Ahmadi, S., Daskalaki, S., Kingston, J.H., Kyngas, J., Nurmi, C., Ranson, D., 2012. An XML format for benchmarks in high school timetabling. Annals of Operations Research 194, 385–397.

Post, G., Di Gaspero, L., Kingston, J.H., McCollum, B., Schaerf, A., 2013. The third international timetabling competition. Annals of Operations Research , 1–7.

Raghavjee, R., Pillay, N., 2012. A comparison of genetic algorithms and genetic programming in solving the school timetabling problem, in: Fourth World Congress on Nature and Biologically Inspired Computing (NaBIC 2012), IEEE. pp. 98–103.

Ross, P., 2005. Hyper-heuristics, in: Burke, E.K., Kendall, G. (Eds.), Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques. Springer. chapter 17, pp. 529–556.

Shambour, M.K.Y., Khader, A.T., Kheiri, A., Özcan, E., 2013. A two stage approach for high school timetabling, in: Lee, M., Hirose, A., Hou, Z.G., Kil, R.M. (Eds.), Neural Information Processing. Springer Berlin Heidelberg. volume 8226 of *Lecture Notes in Computer Science*, pp. 66–73.

Sørensen, M., Kristiansen, S., Stidsen, T.R., 2012. International timetabling competition 2011: an adaptive large neighborhood search algorithm, in: Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling (PATAT '12), pp. 489–492.

de Werra, D., 1997. The combinatorics of timetabling. European Journal of Operational Research 96, 504–513.

Zhang, D., Liu, Y., M'Hallah, R., Leung, S.C.H., 2010. A simulated annealing with a new neighborhood structure based algorithm for high school

timetabling problems. European Journal of Operational Research 203, 550–558.

Table 2: The performance comparison of the fifteen selection hyper-heuristics which are run for 10 trials on all problem instances. The second column provides the number of instances for which the corresponding selection hyper-heuristic produces the best average cost (including ties). The third column provides the number of instances for which the corresponding selection hyper-heuristic produces the best-of-trials cost (including ties). The last column provides the score of each hyper-heuristic, which is computed using the same scoring scheme as in ITC 2011 for ranking different approaches.

HH	#best/tie avg.	#best/tie min.	score
SRIE	1 / 0	0 / 2	7.28
SRSA	0 / 0	0 / 1	8.75
SRGD	1 / 0	1 / 2	**5.33**
RPIE	1 / 0	1 / 2	6.58
RPSA	1 / 1	0 / 1	7.88
RPGD	3 / 0	2 / 0	**5.33**
RDIE	1 / 0	2 / 3	6.94
RDSA	3 / 0	2 / 1	7.78
RDGD	2 / 0	1 / 4	**5.34**
RPDIE	0 / 0	2 / 2	6.71
RPDSA	1 / 1	1 / 1	7.65
RPDGD	2 / 0	2 / 2	**5.40**
CFIE	0 / 0	0 / 0	13.02
CFSA	1 / 0	0 / 0	12.91
CFGD	0 / 0	0 / 0	13.10

Table 3: The average performance comparison of top four selection hyper-heuristics based on the mean cost (quality) of resultant solutions over 10 trials. The pairwise statistical test considering RPGD vs. each hyper-heuristic in {SRGD, RDGD and RPDGD} is based on Mann-Whitney-Wilcoxon. Given two hyper-heuristics RPGD vs X: $>$ ($<$) indicates that RPGD (X) performs significantly better than X (RPGD), while $\geq$ ($\leq$) indicates that RPGD (X) performs slightly better X (RPGD). The best mean values for each instance are highlighted in bold.

Instance	RPGD	vs.	SRGD	vs.	RDGD	vs.	RPDGD
Instance2-Brazil	0.00001199	$\geq$	0.30001082	$\leq$	**0.00001073**	$\leq$	0.00001121
Instance3-Brazil	0.00001856	$\geq$	0.00001904	$\leq$	**0.00001831**	$\geq$	0.00001883
Instance4-Brazil	12.00001607	$\geq$	12.20001649	$\geq$	12.70001556	$\leq$	**11.90001555**
Instance6-Brazil	**0.00003098**	$\geq$	0.00003123	$\geq$	0.00003166	$\geq$	0.10003282
ElementarySchool-Finland	0.00000043	$\geq$	0.00000045	$\leq$	0.00000041	$<$	**0.00000037**
SecondarySchool2-Finland	**0.00000179**	$\geq$	0.00000224	$\geq$	0.00000218	$\geq$	0.00000218
Aigio 1st HS 2010-Greece	3.20009039	$\geq$	4.00009285	$\leq$	3.00009371	$\leq$	**2.40008882**
WesternGreeceUni3-Greece	**0.00000124**	$\geq$	0.00000128	$\geq$	0.00000132	$\geq$	0.00000133
WesternGreeceUni4-Greece	0.00000269	$\leq$	**0.00000246**	$\leq$	0.00000247	$\leq$	0.00000251
WesternGreeceUni5-Greece	0.00000036	$\leq$	0.00000028	$\leq$	**0.00000026**	$\leq$	0.00000034
Instance4-Italy	0.00008743	$\leq$	0.00007749	$\leq$	0.00007982	$\leq$	**0.00007532**
Instance1-Kosovo	27.50101321	$\leq$	**26.40098814**	$\leq$	26.80095814	$\geq$	29.30101231
Kottenpark2003-Netherlands	2.10667418	$\leq$	**1.30571195**	$\leq$	1.70599276	$\leq$	1.80647190
Kottenpark2005A-Netherlands	35.60301159	$\leq$	**35.30292741**	$\geq$	35.60309963	$\geq$	36.60288450
Kottenpark2008-Netherlands	32.41802756	$\geq$	33.11790055	$\leq$	**32.01804926**	$\geq$	34.81819965
Kottenpark2009-Netherlands	**33.5325698**	$\geq$	35.43325980	$\geq$	38.13352230	$\geq$	35.73329525
Woodlands2009-South Africa	**2.0000288**	$>$	2.10002925	$\geq$	2.00002921	$\geq$	2.10002907
School-Spain	**0.00022784**	$\geq$	0.00023358	$\geq$	0.00023082	$\geq$	0.20036330

Table 4: The performance comparison of RPGD to HySST, GOAL, HFT, Lectio and GGSA over 10 trials showing the best cost, indicated as infeasibility-value.objective-value for each instance. The best values are highlighted in bold.

Problem	RPGD	HySST	GOAL	HFT	Lectio	GGSA
Instance2-Brazil	0.00096	1.00069	1.00051	5.00183	**0.00019**	0.00046
Instance3-Brazil	0.00152	0.00096	**0.00087**	26.00264	0.00112	0.00122
Instance4-Brazil	10.00143	2.00238	16.00104	63.00225	**1.00172**	1.00234
Instance6-Brazil	0.00266	2.00229	4.00207	21.00423	**0.00183**	0.00201
ElementarySchool-Finland	0.00004	0.00004	**0.00003**	29.00080	**0.00003**	**0.00003**
SecondarySchool2-Finland	0.00009	0.00006	**0.00000**	28.01844	0.00014	0.00035
Aigio 1st HS 2010-Greece	0.00596	0.00322	**0.00006**	45.03665	0.00653	0.00514
WesternGreeceUni3-Greece	0.00010	0.00010	**0.00005**	14.00198	30.00002	0.00016
WesternGreeceUni4-Greece	0.00019	0.00016	**0.00005**	233.00277	35.00070	0.00030
WesternGreeceUni5-Greece	0.00002	0.00001	**0.00000**	9.00174	4.00013	0.00004
Instance4-Italy	0.00520	0.04012	**0.00169**	250.05966	0.00225	0.00882
Instance1-Kosovo	**17.09084**	1065.17431	38.09789	986.42437	274.04939	71.35367
Kottenpark2003-Netherlands	0.40862	0.47560	0.87084	203.8792	34.5596	**0.18738**
Kottenpark2005A-Netherlands	**26.26129**	26.35251	27.37026	393.40463	185.83973	30.27471
Kottenpark2008-Netherlands	24.99999	32.71562	**10.33034**	INVALID	84.99999	51.99999
Kottenpark2009-Netherlands	**22.99999**	33.99999	25.14030	337.99999	97.9606	31.99999
Woodlands2009-South Africa	2.00279	2.00047	2.00012	59.00336	**0.00094**	0.00121
School-Spain	0.01451	0.01247	**0.00597**	63.13873	0.01927	0.04005
Average ranking	3.08	3.29	**1.52**	5.86	3.56	3.14