Received: August 11, 2022. Revised: September 21, 2022.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

A Modified Great Deluge Algorithm with Dual Decay Rate for School Timetabling

Billel Arbaoui[1]* Juliana Wahid[1] Syariza Abdul-Rahman[2]

[1]*School of Computing, Universiti Utara Malaysia, Malaysia*
[2]*School of Quantitative Sciences, Universiti Utara Malaysia, Malaysia*
* Corresponding author's Email: billel_arbaoui@ahsgs.uum.edu.my

Abstract: Great deluge algorithm is one common metaheuristic solution methodologies to high school timetabling problem. Previous studies focused on great deluge algorithm with linear or non-linear decay rate and selection hyper-heuristics with other local search metaheuristics for solving high school timetabling problem. The decay rate in great deluge algorithm is apparently based on cost and number of iterations i.e., the amount of cost reduction per iteration. However, feasibility of solution and its objective function is still separated from the decay rate especially for difficult and complex high school timetabling problem, which limit the search to converge to a better solution. A modified great deluge algorithm for solving high school timetabling problem with various neighbourhood structures is presented in this paper. Two decay rates are proposed in the original great deluge algorithm to minimize the hard and soft constraints separately. It allows for escaping from local optima based on certain threshold of the two decay rates and thus allowing to explore more of its surroundings in the search space using various neighbourhood structures. The proposed method was examined in terms of the effectiveness of both decay rates in finding different path of solution to avoid local optima. Computational studies on well-known benchmark instances of high school timetabling show that the suggested verification method is successful. The experimental results show that our suggested algorithm outperforms other high school timetabling algorithms in the literature remarkably. It is ranked as the second best among meta-heuristic approaches and the fourth best among mathematic-based approaches of high school timetabling. The performance improvement brought about by the two decay rates is effective with respect to the objective function. High-quality solutions were produced without having the drawback of requiring the solution feasibility and objective function to be separated from the decay rate.

Keywords: Dual decay rate, Great deluge algorithm, School timetabling problem, XHSTT.

1. Introduction

The high school timetabling problem (HSTP) was first addressed in 1975 and is still an ongoing research topic. The HSTP is an NP-complete (nondeterministic polynomial-time complete) problem [1] where it is unknown whether the problem can be solved exactly in a reasonable amount of time especially when the problem size grows. High schools use a weekly timetable for the class session to avoid overlapping time slot and teacher [2, 3]. Exact, heuristic, hyper-heuristic, metaheuristics, and math heuristics are examples of HSTP solving methodologies [4]. Based on research by [5, 6] that focused on the usage of comparable HSTP data sets, an extensible markup language for high school timetabling (XHSTT) were mostly used.

Exact method is a mathematic-based approach which able to yield optimal solution but limited to solving small instances. Various modifications to exact method have been proposed to enhance the search especially in HSTP to cater its problem complexity and size. Some of the mathematic-based approaches proposed for HSTP are Mixed-Integer linear Programming [7, 8], Matheuristic [8, 9], MaxSAT-based large neighborhood search [6, 10] and hyper-heuristic with hidden Markov model [6, 11]. Mixed-Integer linear Programming [6, 7] offered the first precise technique for dealing with

any instance of the XHSTT format. Matheuristic [6, 9] provided a matheuristic technique that mixes VNS with neighbourhoods based on mathematical programming. MaxSAT-based large neighborhood search [6, 10] combined local search with a unique wide neighborhood search based on maxSAT. A novel sequence-based selection hyper-heuristic was developed to solve HSTP using a fixed parametrized technique and a hidden Markov model [6, 11].

Metaheuristic is a method which capable to produce a near-optimal solution within reasonable times for larger problem size. Population and local search algorithms are two types of metaheuristic algorithms. A local search-based algorithm explores a single solution over iterations by applying local changes, while a population-based algorithm is a collection of solutions that are updated repeatedly until the termination condition is met [12]. Examples of metaheuristic approaches for solving HSTP are variable neighbourhood search algorithm (VNS), simulated annealing algorithm and harmony search algorithm. The VNS was used to optimize HTSP with six different neighbourhood structures in [6, 9]. Simulated annealing with iterated local search (SA + ILS), skewed general VNS (SGVNS), and general VNS (GVNS) variants for solving the HSTP were proposed in [6, 13], with six different neighbourhood structures. A hybrid harmony search with simulated annealing (HSA-SA) for HSTP with fixed parameters was introduced in [8, 14]. Seven neighborhood structures were utilized using a hybrid metaheuristic based on SA and iterated local search (GOAL) in [4, 15].

Local search-based algorithms have a role in escaping local optima over iterations through a process known as exploitation [4]. One of the algorithms that expands the search space's possibilities is named as great deluge algorithm (GDA) proposed by Dueck [16]. It is a continuous global optimization solution-based metaheuristic that use the solution search technique to accept inferior solutions based on an acceptance rule. This assists in escaping local optima [17]. In an analogue to GDA, a climber on a hill attempts to avoid getting his or her feet wet (local optima) while the water level rises in the hopes of discovering a new path [18].

There are several studies which employed GDA for solving HSTP. Studies [6, 11, 19] employed GDA as a move acceptance mechanism in a hyper-heuristics algorithm by mixing several reusable components for solving XHSTT data sets. The maximum change in the objective function's expected range is set fixed. In [20, 21], a non-linear reduction rate of GDA was applied to solve three datasets from Tanzanian high schools. Prior work optimised both the hard and soft constraints in HSTP simultaneously using a single water level rate and linear and non-linear GDA. However, those studies employed a single decay rate which may result in optimization being limited and restrict the search to converge to a better solution.

The decay rate in GDA is apparently based on cost and number of iterations i.e., the amount of cost reduction per iteration. However, the feasibility of solution and its objective function is still separated from the decay rate especially for difficult and complex high school timetabling problem, which limit the search to converge to a better solution. This is shown by the previous studies in HSTP which applied linear and non-linear GDA with only one water level rate that used for minimizing both the hard and soft constraints at the same time. This perhaps contributes to limited optimization and good result for one of the constraints may not be achieved. Therefore, in this study, two decay rate is proposed to minimize the hard and soft constraints, so that both constraints minimization process will be optimized separately.

This article is structured as follows: In the second section, the HSTP is discussed. The fundamental of GDA is described in the third section, along with the recommended approach. The tests, results and the discussions are highlighted in Section 4. Finally, the paper's conclusion is included in the last section.

2. High school timetabling problem

The HSTP is concerned with creating a teaching and learning calendar for a specific high school. The HSTP may be described in formal terms as follows. The collection of events E represents the lectures that will be held. Each resource is of a certain type, such as a room, a student, or a teacher, and the collection of resources is marked as R. Each e within E ($e \in E$) event necessitates a subset of resources. A timeslot must be provided to an event. T stands for the set of timeslots, which is created by splitting each day into multiple non-overlapping timeslots.

Some HSTP datasets have been established in different parts of the world such as Australia, Brazil, England, Finland, Greece, and the Netherlands [4]. These datasets were formatted to benchmark datasets by representing it with an XML schema and named as XHSTT. The following are the four aspects of the XHSTT instance:

Times: A set of time periods that occur throughout the day, such as Tuesday 8:00- 9:00 and Monday 9:00-10:00. Resources: Classes, teachers,

rooms, and students are all examples of resources. This also offers a resource collection for a set of teachers (such as physics or mathematics). Events are lessons that need to be scheduled. It comes together at the right time and with the right resources to offer information (duration, workload, time, and resources). The number of timeslots in which the event has time intervals is defined by its duration, and certain events have pre-allocated time intervals [22]. The workload is a collection of events with a specified time and resources. Events are given one or more solution events or subevents. For example, physics lectures lasting nine hours a week may be split into 3 subevents, or they could be arranged as a single nine-hour subevent. **Constraints:** It describes the restrictions on real-world data instances as shown in Table 1.

Table 1. The XHSTT format constraints

Name	Description
Assign Resource	Specifies how resource allocation for solution events must be done
Assign Time	Sets of times that the solution events must assign
Avoid Clashes	Some resources should have no conflicts, according to the Avoid Clashes list
Avoid Split Assignments	Split assignments aren't suitable for a variety of event resources, according to the list
Avoid Unavailable Times	Argues that certain resources cannot engage in many tasks at the same time
Cluster Busy Times	Specify how many time groups a resource can be utilized in
Distribute Split Events	Determines the value limitation for solution events of a given length in the instance event's periods.
Limit Busy Times	Assign the minimum and maximum values on the value of time by considering numerous different scenarios in which a resource might be occupied
Limit Idle Times	Anytime idle resources are available, indicates the time limits
Link Events	Provides the pattern for a large number of events to take place at the same time
Order Events	Determines the sequence in which double-event timings should appear
Prefer Resources	Several "resources" are preferred for allocating to several solution resources, according to the resources' list
Prefer Times	Several "times" are more suitable for several solution event assignments, according to the times' list
Split Events	List into a set of sub-events with a constrained number of options
Spread Events	Specifies that the solution events of a series of events should be spaced out across time

$$Cost = Weight \times CostFunction\,(deviation) \quad (1)$$

It has a relationship between the cost being estimated (as shown in Eq. (1)), and the constraint being unsuccessful to fulfill its requirement. The weight is specified in the range between 0 to 1000 by constraints, and *CostFunction* determines the deviation. Quadratic, Step, and Linear cost functions are the three types of cost functions. Each constraint might be either soft or hard. Hard constraints are the requirements that need to be adhered with which reflect the solution feasibility. Soft constraints, on the other hand, determine the quality of a solution, with smaller numbers of soft constraint violations signifying better solutions. The constraints are divided into three main parts which are 1) fundamental (Assign Resource, Assign Time, Prefer Times, and Prefer Resource), 2) events (Split Events, Distribute Split Events, Avoid Split Assignments, Spread Events, Link Events, and Order Events), and 3) resources (Clashes Avoidance, Unavailable Times Avoidance, Reduce Idle Times, Group Busy Times, Reduce Busy Times, and Reduce Workload).

3. A modified great deluge algorithm

GDA was proposed by Dueck [16]. It is a solution-based metaheuristic for continuous global optimization that allows inferior solutions using the solution search approach based on an acceptance rule. This assists in the escape from local optima [17].

The GDA metaheuristic's fundamental concept is to use a threshold value (water level) to direct the diversification of search by accepting inferior neighbouring candidates [16, 23].

As shown in Fig. 1, the analogy is usually state as a human (current solution) hiking a hill (depict as a maximization problem) that try to move in any direction which does not make his/her feet wet. A path is discovered based on the rises of water level ($S1, S2,$ or $S3$) in order to escape from local optima.

As shown in Algorithm 1, throughout the search, the level's value is linearly dropped (defined as decay rate). The present solution's objective function is dropped/raised as the level value is reduced/raised till convergence. Each iteration will create a new adjacent solution of neighbourhood structure *NS* from the present solution ($soln_{cur}$). The *waterLevel*, which is often selected in line 1

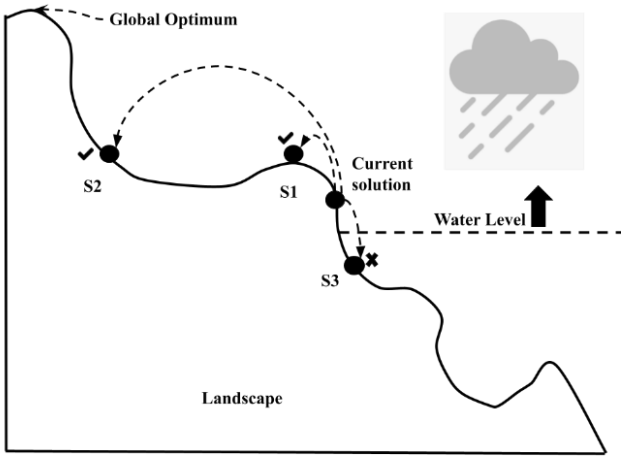

Figure. 1 GDA search as a graphical representation

Algorithm 1: Basic great deluge algorithm

Input: $soln_{cur}, NS, raindSpeed, totalIter$
Output: $soln_{new}$
1 $waterLevel \leftarrow f(soln_{cur})$
2 $decayRate \leftarrow \boldsymbol{Eq.(2)}$
3 **for** $i \leftarrow 0$ **to** $totalIter$ **do**
4 $soln_{new} \leftarrow generateNeighbour(soln_{cur}, NS)$
5 **if** $f(soln_{new}) \leq f(soln_{cur})$ **then**
6 $soln_{cur} \leftarrow soln_{new}$
7 **else if** $f(soln_{new}) \leq waterLevel$ **then**
8 $soln_{cur} \leftarrow soln_{new}$
9 **else**
10 $soln_{new} \leftarrow soln_{cur}$
11 **end if**
12 $waterLevel \leftarrow \boldsymbol{Eq.(3)}$
13 **end for**
14 **return** $soln_{new}$

$soln_{cur}$	Present solution
$soln_{new}$	New solution
NS	Neighbourhood structure
$raindSpeed$	Rain speed
$totalIter$	Total number of iterations
$waterLevel$	Threshold value
$decayRate$	Specified decay rate
i	Iteration
$generateNeighbour$	New neighbouring solution

with the quality of the original solution, serves as the starting point for the algorithm and is lowered/raised by a specific rate throughout each iteration by using Eq. (3). The initial formula Eq. (2) for calculating the rate (*decayRate*) is in line 2.

$$decayRate = f(soln_{cur}) \times \frac{raindSpeed}{totalIter} \quad (2)$$

$$waterLevel = waterLevel - decayRate \quad (3)$$

This study proposes a modified version of GDA (MGDA) where two decay rate is embedded into the original GDA to assist in minimizing the hard and soft constraints of a complex HSTP. Algorithm 2 demonstrates the pseudo code of the MGDA. The *waterLevelSoft* and *waterLevelHard* are representing the current number of soft and hard constraints violation respectively (lines 1 and 2). The decay rates for soft and hard water levels are determined in the initial stages of the algorithm based on current violation of soft or hard constraints multiplied with total iterations (*totalIter*) and then divided by rain speed (*rainSpeed*) by using Eq. (4) and Eq. (5) (as shown in lines 3 and 4).

$$decayRateSoft = f(soln_{cur})_{soft} \times \frac{raindSpeed}{totalIter} \quad (4)$$

$$decayRateHard = f(soln_{cur})_{hard} \times \frac{raindSpeed}{totalIter} \quad (5)$$

Next, based on the *totalIter* the new solution will be produced with ten neighbourhood-moves that selected randomly. The following are the neighbourhood-moves: **Swap Day and Timeslot:** the day and timeslot of two meets are swapped. **Move Day and Timeslot:** this move shifts the day and timeslot of meet to other days and timeslots that are available. **Swap Resources:** this task refers to the type of resources such as teacher, class, and room. This swap works with two resources. Two conditions are required, which are the two resources, such as teacher with only teachers. **Move Resources:** this task moves the available resources that are not assigned into the solutions, which are located in the instance, and unassigns the resources that are already inside the solution. **Block Meet Swap:** Block swapping is the same as ordinary swapping, except that it treats adjacent two or more same meets as one block. For example, when swapping a meet of duration 1 assigned to the first time on Wednesday with a meet of duration 2 assigned to the second time on Wednesday, *Block Meet Swap* swaps the adjacent of *Arab2* meet with the *Hist1* meet.

A group of moves is also applied namely as Kempe and Ejecting Moves. Ejecting indicates unassigning, whereas Kempe indicates moving. The unique objective of all moves is to remove the overlap/clashes. In this study, five types of moves are used, which are as follows: **Kempe Meet Move:** it works with two meets in the solution space. The Kempe meet relocates a meet from its current location to a target meet at an offset and adds extra

demand to the overall demand of the relocated meets. For example, if the operation does not attempt to shift any meetings back the other way. A movement or exchange is generally the consequence, although it might be complicated. **Kempe Meet Move Time:** this move is similar to *Kempe Meet Move* but operates with one meet from the solution space and time from the instance. **Task Ejecting Move Resource:** this transfers the task (teacher, subject, room, student) in the solution back to the instance, unassigning it from all clashing tasks. The resources (teacher, subject, room, student) from the instance are transferred to the solution. This move ignores the matching, merely fails on preassigned times, and unassigns the clashing tasks. **Ejecting Meet Move:** the ejecting meet move is a version of the Kempe meet move. This starts by moving the meet to the targeted meet at offset, then detects the meets that need to be moved back the other direction, just like Kempe meet moves, but instead of moving them, it unassigns and stops them. **Ejecting Meet Move Time**: this move is Ejecting Meet Move but operates with one meet from the solution space and time from the instance.

Algorithm 2: Modified great deluge algorithm

Input: $soln_{cur}, NS, raindSpeed, totalIter$
Output: $soln_{new}$

1 $waterLevelSoft \leftarrow f(soln_{cur})_{soft}$
2 $waterLevelHard \leftarrow f(soln_{cur})_{hard}$
3 $decayRateSoft \leftarrow \boldsymbol{Eq.(4)}$
4 $decayRateHard \leftarrow \boldsymbol{Eq.(5)}$
5 **for** $i \leftarrow 0$ **to** $totalIter$ **do**
6 $soln_{new} \leftarrow generateNeighbour(soln_{cur}, NS)$
7 **if** $f(soln_{new}) \leq f(soln_{old})$ **then**
8 $soln_{cur} \leftarrow soln_{new}$
9 **else if** $(f(soln_{new})_{soft} \leq waterLevelSoft$ or $f(soln_{new})_{soft} = f(soln_{cur})_{soft})$ **and** $(f(soln_{new})_{hard} \leq waterLevelHard$ or $f(soln_{new})_{hard} = f(soln_{cur})_{hard})$ **then**
10 $soln_{cur} \leftarrow soln_{new}$
11 **else**
12 $soln_{new} \leftarrow soln_{cur}$
13 **end if**
14 $waterLevelSoft \leftarrow \boldsymbol{Eq.(6)}$
15 $waterLevelHard \leftarrow \boldsymbol{Eq.(7)}$
16 **end for**
17 **return** $soln_{new}$

$soln_{cur}$	Present solution
$soln_{new}$	New solution
NS	Neighbourhood structure
$raindSpeed$	Rain speed
$totalIter$	Total number of iterations
$waterLevelSoft$	Threshold value for soft
$waterLevelHard$	Threshold value for hard
$decayRateSoft$	Specified decay rate for soft
$decayRateHard$	Specified decay rate for hard
i	Iteration
$generateNeighbour$	New neighbouring solution

Next after new solution is produced, if the new neighboring solution's quality assessment value is better than or equal to the current best solution, it will always be selected (lines 7 and 8). A solution that is inferior to the current best solution will be approved only if its quality assessment value is less than or equal to the threshold (*waterLevelSoft* and *waterLevelHard*) as shown in lines 9 and 10.

To avoid local optima, the MGDA always accepts a modification that improves solution quality, and a worse solution if the solution quality falls below or equals a threshold (*waterLevelSoft* and *waterLevelHard*). Each iteration, new values for *waterLevelSoft* and *waterLevelHard* are generated by using *decayRateSoft* Eq. (6) and *decayRateHard* Eq. (7), respectively (lines 14 and 15).

$$waterLevelSoft = waterLevelSoft - decayRateSoft \quad (6)$$

$$waterLevelHard = waterLevelHard - decayRateHard \quad (7)$$

Throughout the search, the value of level is slightly reduced in a linear way (given decay rate for soft and hard). As the level values of soft and hard are reduced, the current solution iterative approach is reduced as well, until convergence.

4. Experiment, results and discussion

This section explains the experimental setup as well as the results, and discussion about comparisons with meta-heuristic, and state-of-the-arts approaches. The proposed MGDA's experimental environment was implemented in C programming language on AMD Ryzen 7 3700X 8-Core with 3.89 GHz processor and 16 GB RAM. To evaluate the proposed approach, XHSTT datasets[1] (real high school timetabling problem) from small to large instances were used. The initial solution of this

[1] High School Timetabling Project (https://www.utwente.nl/en/eemcs/dmmp/hstt/).

study is utilized from [24], which is a constructive solution technique.

The number of iterations (*totalIter*) used in this study is 200 multiply the size of data instance. The iteration number is a setting that applied for both the GDA and MGDA. The value was determined through a series of extensive trials with various parameter combinations. The initial soft and hard water levels for MGDA and the water level for GDA were set using objective function values, and *rainSpeed* was set at 0.2 and 0.4, respectively to progressively raise the water level. After a series of extensive experiments employing various parameter value combinations, the values are determined. The overview of parameter settings is shown in Tables 2 and 3.

The MGDA results are shown in Table 4 and compared with the initial solution and GDA results. The results indicate the objective function value of the solution which is represented by the total number of hard and soft constraint violations that aim to be minimized. Overall, the MGDA performed respectable results in almost all datasets when compared with the initial solution [24]. The MGDA performed better in all large datasets such as Netherlands datasets (*NL-KP-03*, *NL-KP-05*, *NL-KP-09*, *Kottenpark2008* and *GEPRO*), Denmark datasets (*DK-FG-12*, *DK-HG-12*, and *DK-VG-09*), *KS-PR-11*, and *UK-SP-06*. For some medium datasets, MGDA performed rather well results when compared with the initial solution [24] such as Australia data instances (*AU-BG-98*, and *AU-TE-99*), *ES-SS-08*, *FI-WP-06*, *GR-PA-08*, *Aigio2010*, *IT-I4-96*, *US-WS-09*, and *ZA-WD-09*.

The MGDA also obtained better results for small datasets such as *ZL-LW-09*, *BrazilInstance5* and *BR-SM-00*. Overall, the MGDA is able to optimize 21 out of 39 datasets (in bold font) when compared with the initial solution [24]. The MGDA was also tested on three Malaysian school timetabling problem datasets [25] which are *MalaysiaSFBT01*, *MalaysiaSFBT02* and *MalaysiaSFBT03* and produced optimal solution for all the three datasets.

GDA was difficult to enter the acceptance threshold when comparing to MGDA based on win-loss records. This is due to the GDA separate the

Table 2. The MGDA parameter settings

Parameters	Values
rainSpeed	0.2
totalIter	200 × size of data instance
waterLevelSoft	Soft objective function value
waterLevelHard	Hard objective function value

Table 3. The GDA parameter settings

Parameters	Values
rainSpeed	0.4
totalIter	200 × size of data instance
waterLevel	Total objective function value

Table 4. Results of the MGDA, GDA, and the initial solution [24]

Instance	Initial solution [24]	MGDA	GDA
Aigio2010	0.00008	**0.00006**	**0.00006**
ArtificialSchool	3.00006	3.00006	3.00006
AU-BG-98	1.00557	**1.00469**	**1.00469**
AU-SA-96	0.00005	0.00005	0.00005
AU-TE-99	0.00087	**0.00085**	**0.00085**
BrazilInstance1	0.00048	0.00048	0.00048
BrazilInstance3	0.00065	0.00065	0.00065
BrazilInstance5	0.00083	**0.00068**	**0.00068**
BrazilInstance7	0.00130	0.00130	0.00130
BR-SA-00	0.00011	0.00011	0.00011
BR-SM-00	2.00125	**2.00119**	**2.00119**
BR-SN-00	0.00096	0.00096	0.00096
DK-FG-12	0.01926	**0.01907**	0.02096
DK-HG-12	12.04790	12.03664	**12.03521**
DK-VG-09	2.04382	**2.03584**	2.03434
ElementarySchool	0.00003	0.00003	0.00003
ES-SS-08	0.00565	**0.00555**	**0.00555**
FI-MP-06	0.00091	0.00091	0.00091
FI-PB-98	0.00005	**0.00005**	0.00007
FI-WP-06	0.00014	**0.00013**	**0.00013**
GEPRO	1.01735	**1.00613**	1.00705
GR-H1-97	0.00000	0.00000	0.00000
GR-P3-10	0.00000	0.00000	0.00000
GR-PA-08	0.00006	**0.00005**	**0.00005**
IT-I4-96	0.00042	**0.00040**	**0.00040**
Kottenpark2008	19.35165	**14.34855**	14.42141
KS-PR-11	0.00009	**0.00004**	0.00006
MalaysiaSFBT01	0.00000	0.00000	0.00000
MalaysiaSFBT02	0.00010	**0.00000**	**0.00000**
MalaysiaSFBT03	0.00006	**0.00000**	**0.00000**
NL-KP-03	0.00929	**0.00921**	0.01252
NL-KP-05	1.02226	**1.01850**	3.01682
NL-KP-09	7.04190	**5.04180**	8.06275
Preveza2008	0.00002	0.00002	0.00002
SecondarySchool2	0.00002	0.00002	0.00002
UK-SP-06	13.00636	**13.00636**	13.00880
UniversityInstance3	0.00006	0.00006	0.00006
UniversityInstance5	0.00000	0.00000	0.00000
US-WS-09	0.00518	**0.00145**	0.00184
VillageSchool	0.00013	0.00013	0.00013
ZA-WD-09	2.00000	2.00000	2.00000
ZL-LW-09	2.00034	**1.00034**	**1.00034**
Wins	-	**10**	**1**
Losses	-	**1**	**10**
Ties	-	**31**	**31**

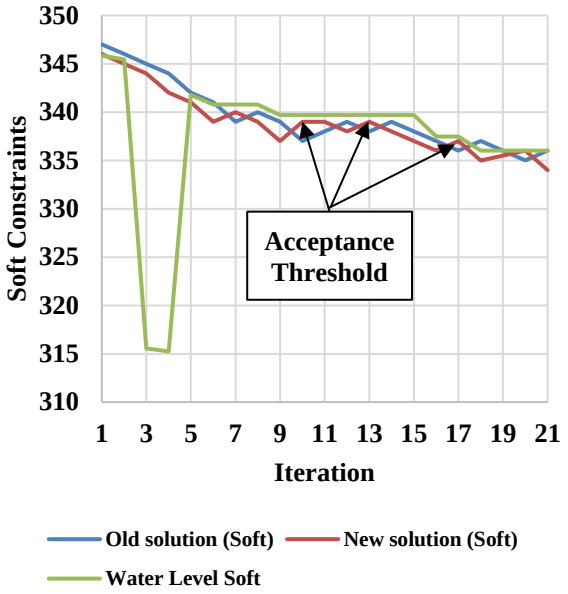

Figure. 2 The soft water level for *US-WS-09* using MGDA

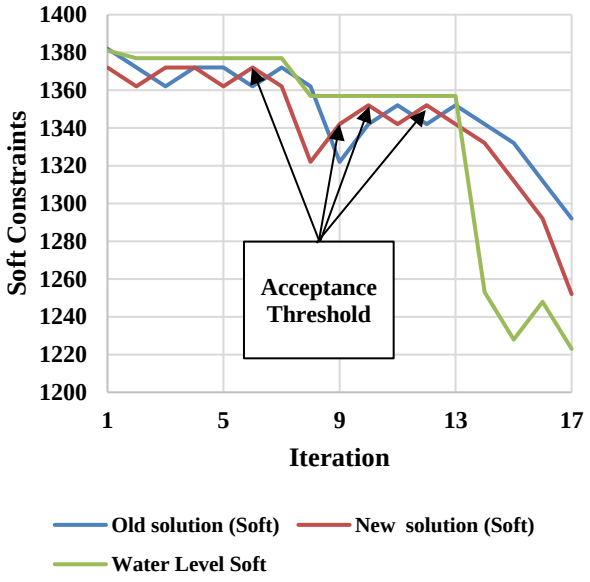

Figure. 3 The soft water level for *NL-KP-03* using MGDA

information on hard and soft constraints' violation which unable to assist decay rate of GDA to converge to a better solution.

However, distinct situation occurred to *DK-HG-12* which produced better result using GDA compared to MGDA. This is maybe due to the used of various good neighbourhood structures that have been tailored made for solving this HSSTP. From the results presented in Table 4, out of 42 instances, MGDA outperformed GDA in 10 instances, with the remaining 31 datasets were tied. Further investigation and analysis on GDA and MGDA

Figure. 4 The soft water level for *AU-BG-98* using MGDA

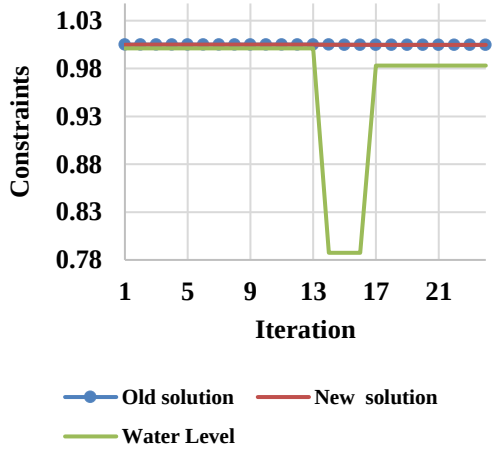

Figure. 5 The water level for *AU-BG-98* using GDA

performance is also presented below. Table 5 compares the results of the MGDA with the results of several meta-heuristics approaches which are VNS [9], SA+ILS [13], SGVNS [13], GVNS [13], HSA-SA [14] and GOAL [15]. In comparison to the meta-heuristics, the MGDA offers the best solutions on 13 datasets and ties on 4 datasets out of 39 datasets. To compare the performance of the meta-heuristics techniques with MGDA, the ranking system used in ITC2011 based on average rank is employed. The constraint violations of the best run on each instance are used to evaluate each method. The algorithm with the lowest average rank is claimed to be the best. When compared to other approaches based on average rank, the MGDA comes in the second place, after the VNS and followed by SGVNS, GVNS, GOAL, HSA-SA and finally SA+ILS.

Table 6 shows the best HSTP results produced utilizing several mathematical-based approaches,

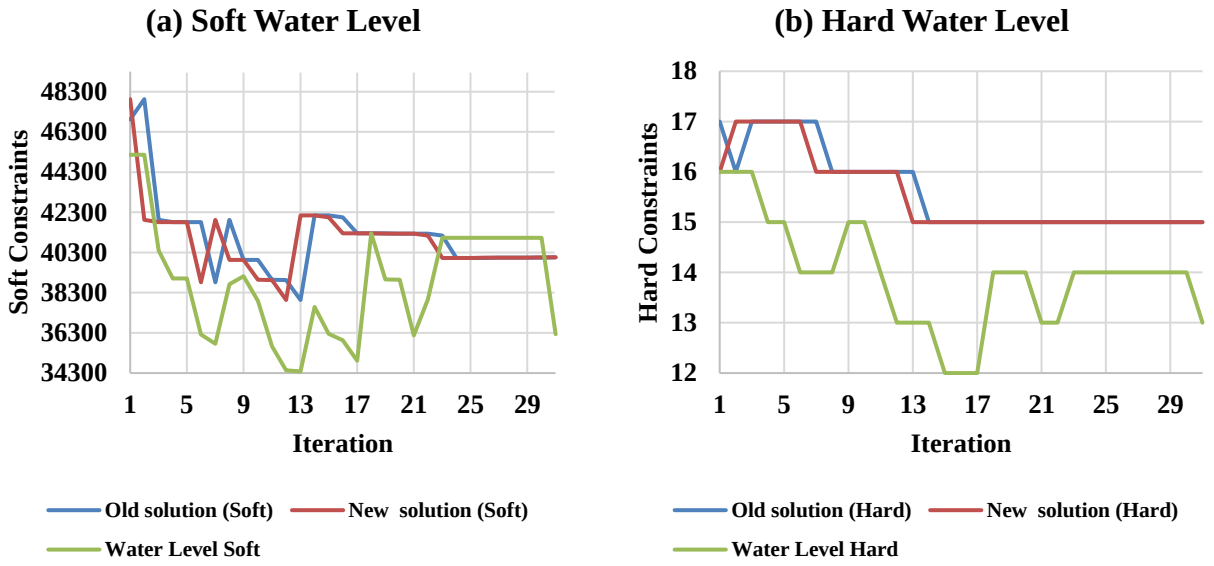

Figure. 6 The (a) soft and (b) Hard water level for *Kottenpark2008* using MGDA

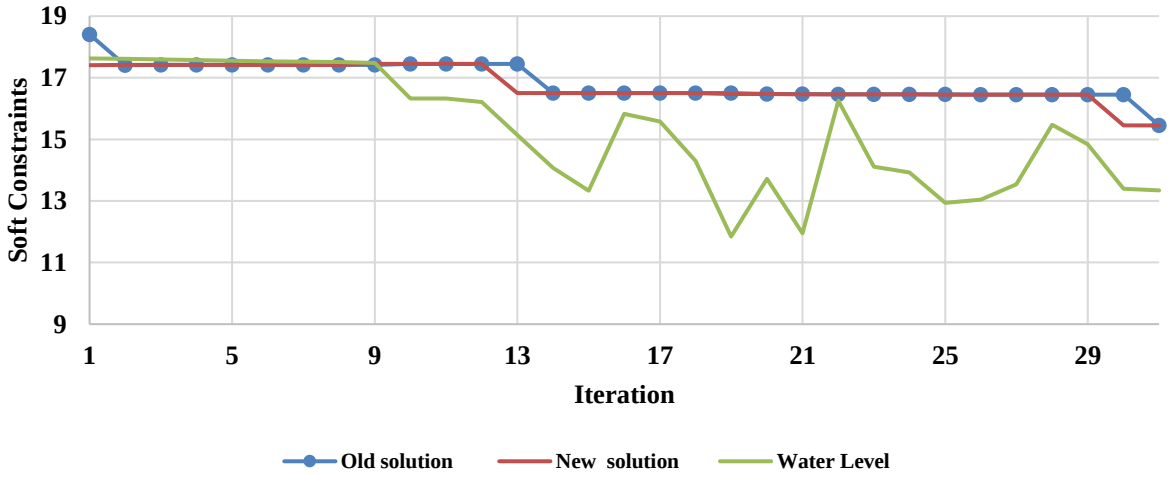

Figure. 7 The water level for *Kottenpark2008* using GDA

including A) Mixed-Integer Linear Programming [7], B) Matheuristics [9], C) MaxSAT-based large neighborhood search [10], and D) Hidden Markov Model Approach to Heuristic Selection [11]. In general, MGDA is successful in minimizing soft constraint violation and reasonable at minimizing hard constraint violations. In comparison to the mathematic-based methods, the MGDA obtains the best results for seven datasets (*Aigio2010*, *BrazilInstance1*, *Kottenpark2008*, *Preveza2008*, *SecondarySchool2*, *UK-SP-06* and *VillageSchool*) and four ties (*ElementarySchool*, *GR-H1-97*, *GR-P3-10*, *UniversityInstance5*). Of the ties, three of them has reached optimality. The average rank is also used to compare the performance of the four algorithms (A, B, C, and D) with MGDA. The best approach was discovered to be B, followed by A, D, MGDA, and C, in that order. MGDA, on the other hand, has been proven to be quite competitive with other mathematic-based methods and capable of achieving some of the best results.

The soft and hard water levels influence the MGDA solution search. Figs. 2, 3, 4, 5, 6, and 7 show the soft and hard water levels based on soft and hard decay rates, respectively. These figures are for the data instances *US-WS-09*, *NL-KP-03*, *AU-BG-98*, and *Kottenpark2008* respectively. The soft water level parameter for the *US-WS-09* and *NL-KP-03* data instances are shown in Figs. 2 and 3, respectively. In the *US-WS-09* dataset at iterations 10, 13, and 17, and in the *NL-KP-03* dataset at iterations 6, 9, 10, and 12, a new solution was selected that was lower than the soft water level (acceptance threshold). We can plainly see that the MGDA attempted to choose a different path in order to avoid local optima.

Table 5. Results obtained using MGDA compared to meta-heuristics-based approaches

Instance	**MGDA**	**VNS** [9]	**SA + ILS** [13]	**SGVNS** [13]	**GVNS** [13]	**HAS-SA** [14]	**GOAL** [15]
Aigio2010	0.00006	-	-	-	-	-	**0.00000**
ArtificialSchool	**3.00006**	-	-	-	-	-	-
AU-BG-98	1.00469	2.00398	6.00450	**1.00401**	4.00370	-	-
AU-SA-96	**0.00005**	4.00021	14.00050	13.00046	12.00051	-	-
AU-TE-99	**0.00085**	1.00036	7.00161	7.00163	7.00151	-	-
BrazilInstance1	0.00048	-	0.00012	**0.00011**	**0.00011**	0.00020	-
BrazilInstance3	**0.00065**	-	-	-	-	0.00154	0.00101
BrazilInstance5	**0.00068**	-	0.00164	0.00149	0.00158	0.00148	-
BrazilInstance7	**0.00130**	-	0.00264	0.00248	0.00249	0.00234	-
BR-SA-00	**0.00011**	0.00022	-	-	-	0.00048	0.00032
BR-SM-00	2.00119	3.00099	0.00091	**0.00090**	0.00094	0.00162	1.00136
BR-SN-00	**0.00096**	0.00104	0.00149	0.00131	0.00148	0.00186	0.00160
DK-FG-12	0.01907	**0.01669**	-	-	-	-	-
DK-HG-12	12.03664	**12.03371**	-	-	-	-	-
DK-VG-09	2.03584	**2.02765**	-	-	-	-	-
ElementarySchool	**0.00003**	-	-	-	-	**0.00003**	**0.00003**
ES-SS-08	0.00555	0.00415	-	-	-	-	**0.00012**
FI-MP-06	0.00091	**0.00084**	0.00086	0.00088	0.00087	0.00090	-
FI-PB-98	0.00005	**0.00000**	-	-	-	-	-
FI-WP-06	0.00013	0.00011	**0.00001**	**0.00001**	**0.00001**	-	-
GEPRO	1.00613	-	1.00566	1.00441	**1.00434**	-	-
GR-H1-97	**0.00000**	**0.00000**	-	-	-	-	-
GR-P3-10	**0.00000**	**0.00000**	-	-	-	-	-
GR-PA-08	0.00005	**0.00003**	-	-	-	-	0.00008
IT-I4-96	**0.00040**	0.00042	-	-	-	0.00082	0.00061
Kottenpark2008	**14.34855**	-	-	-	-	-	-
KS-PR-11	**0.00004**	**0.00004**	-	-	-	-	-
NL-KP-03	0.01376	**0.01151**	0.01409	0.01281	0.01216	-	0.05355
NL-KP-05	1.01850	8.04460	0.01078	**0.00877**	0.00881	-	24.13930
NL-KP-09	**5.04180**	8.08370	-	-	-	-	19.05565
Preveza2008	**0.00002**	-	-	-	-	-	-
SecondarySchool2	0.00002	-	-	-	-	**0.00000**	**0.00000**
UK-SP-06	13.00636	53.01524	0.18092	**0.12466**	0.12542	-	-
UniversityInstance3	0.00006	-	-	-	-	-	**0.00005**
UniversityInstance5	**0.00000**	-	-	-	-	-	**0.00000**
US-WS-09	0.00145	**0.00124**	-	-	-	-	-
VillageSchool	**0.00013**	-	-	-	-	-	-
ZA-WD-09	2.00000	3.00000	-	-	-	-	**0 .00441**
ZL-LW-09	1.00034	**0.00016**	0.00022	0.00024	0.00024	-	-
Rank Average	**2.08**	**2.04**	**3.40**	**2.40**	**2.67**	**3.27**	**2.75**
Rank	**2**	**1**	**7**	**3**	**4**	**6**	**5**

Table 6. Results obtained using MGDA compared to the-state-of-the-art methods

Instance	**MGDA**	**A** [7]	**B** [9]	**C** [10]	**D** [11]
Aigio2010	**0.00006**	-	-	0.04582	-
ArtificialSchool	3.00006	-	-	**0.00012**	-
AU-BG-98	1.00469	3.00494	2.00398	-	**0.00520**
AU-SA-96	0.00005	8.00052	3.00021	-	**0.00002**
AU-TE-99	0.00085	1.00140	1.00036	-	**0.00061**
BrazilInstance1	**0.00048**	-	-	-	-
BrazilInstance3	0.00065	**0.00026**	-	0.00075	-
BrazilInstance5	0.00068	**0.00030**	-	0.00224	-
BrazilInstance7	0.00130	**0.00122**	-	0.00603	-
BR-SA-00	0.00011	**0.00005**	0.00005	0.00057	0.00010
BR-SM-00	2.00119	0.00061	**0.00052**	0.00214	2.00117
BR-SN-00	0.00096	0.00060	**0.00035**	0.00352	0.00101
DK-FG-12	0.01907	2.23705	0.01668	-	**0.01522**
DK-HG-12	12.03664	293.3211	12.03371	-	**12.0263**
DK-VG-09	2.03584	20.18966	2.02765	-	**2.02731**
ElementarySchool	**0.00003**	-	-	0.00003	-
ES-SS-08	0.00555	0.00357	**0.00351**	-	0.00517
FI-MP-06	0.00091	0.00088	**0.00077**	0.00504	0.00089
FI-PB-98	0.00005	-	**0.00000**	0.01309	0.00008
FI-WP-06	0.00013	**0.00001**	0.00002	0.00812	0.00007
GEPRO	1.00613	**1.00566**	-	-	-
GR-H1-97	**0.00000**	-	**0.00000**	**0.00000**	**0.00000**
GR-P3-10	**0.00000**	-	**0.00000**	0.02329	**0.00000**
GR-PA-08	0.00005	0.00008	**0.00003**	0.00141	0.00004
IT-I4-96	0.00040	**0.00027**	**0.00027**	0.16979	0.00038
Kottenpark2008	**14.34855**	-	-	-	-
KS-PR-11	0.00004	0.00003	**0.00000**	0.29946	0.00003
NL-KP-03	0.01376	0.01410	0.01103	-	**0.00466**
NL-KP-05	1.01850	0.01078	8.04460	-	**0.00811**
NL-KP-09	5.04180	**0.09035**	7.64470	-	2.07495
Preveza2008	**0.00002**	-	-	0.05617	-
SecondarySchool2	**0.00002**	-	-	0.03523	-
UK-SP-06	**13.00636**	-	53.01524	-	19.01294
UniversityInstance3	0.00006	**0.00005**	-	0.00007	-
UniversityInstance5	**0.00000**	-	-	**0.00000**	-
US-WS-09	0.00145	-	**0.00124**	-	0.00512
VillageSchool	**0.00013**	-	-	-	-
ZA-WD-09	2.00000	**0.00000**	**0.00000**	**0.00000**	9.00000
ZL-LW-09	1.00034	-	**0.00000**	0.01039	0.00052
Rank Average	**2.31**	**2.25**	**1.86**	**2.95**	**2.04**
Rank	**4**	**2**	**1**	**5**	**3**

The differences between the two techniques in avoiding local optima can be seen in Figs. 4 and 5. In Fig. 4, it is obviously can be seen that the MGDA soft water level was perfectly dropped with both the old and new soft solution. However, in Fig. 5 the GDA is attempting to reach the acceptance threshold, yet it is a challenging task.

Fig. 6 shows the MGDA with soft and hard water levels which is the trend is different from the other datasets. The hard constraints of *Kottenpark2008* data instance decreased by one hard constraint at iteration 2 in which the hard water level is hard to reach by new solution. By comparison with GDA water level, the search is difficult to enter the acceptance threshold expat early iterations and it is shown in Fig. 7.

5. Conclusions

The GDA is one of powerful metaheuristic approaches and has potential for more search improvement. It is known from the literature that finding a good timetable in HSTP utilising modern metaheuristic algorithms is still difficult. Overall, the literature shows that GDA continues to be remarkably active in solving challenging optimization problems such as efficient task scheduling in grid computing [26], feature selection problems [27] and feature selection from academician data [28]. Thus, this research presents a modified great deluge metaheuristic for solving HSTP.

Computational tested on well-known benchmark datasets verified its effectiveness. The suggested technique was examined in terms of the effectiveness of both decay rates and comparisons with the standard GDA and other metaheuristic and mathematic-based methodologies when tested on HSTTP benchmark datasets were also presented. When compared to previous heuristic algorithms of HSTP, MGDA is not only effective at producing good quality timetables, but also very efficient in solution searching through two decay rates for MGDA. It is regarded as an improved variation of GDA. It effectively navigates and directs the search in each neighbourhood structure while avoiding from being caught in the local optimum of each neighbourhood structure. This is due to the advantage of the excellent diversification capability of two decay rates. On average, the proposed strategy was ranked as the second best among meta-heuristics methods and the fourth best among mathematic-based methods. The modification strategy used by great deluge was effective and simple to execute. Overall, the MGDA was able to provide satisfactory results especially on large datasets. Future study should be focused on hybridizing population and local search algorithms with MGDA to improve the intensity of MGDA.

Conflicts of interest

There are no ethical concerns with this study.

Author contributions

The theoretical algorithm is structured by the first author, who also carried out the numerical experiment and discussed the results. The second author finished the comprehensive review. The third author provided the overall review.

Acknowledgments

The authors would like to express their gratitude to the Ministry of Higher Education for supporting this study under the Fundamental Research Grant Scheme (FRGS/1/2017/ICT02/UUM/03/1), as well as RIMC, Universiti Utara Malaysia, for the study's administration.

References

[1] S. Even, A. Itai, and A. Shamir, "On the Complexity of Timetable and Multicommodity Flow Problems", *SIAM J. Comput.*, Vol. 5, No. 4, pp. 691–703, 1976, doi: 10.1137/0205048.

[2] A. Schaerf, "Local search techniques for large high school timetabling problems", *IEEE Trans. Syst. Man, Cybern. Part ASystems Humans.*, Vol. 29, No. 4, pp. 368–377, 1999, doi: 10.1109/3468.769755.

[3] S. Faudzi, S. Abdul-Rahman, and R. Abd Rahman, "An Assignment Problem and Its Application in Education Domain: A Review and Potential Path", *Adv. Oper. Res.*, Vol. 2018, pp. 1–19, 2018, doi: 10.1155/2018/8958393.

[4] J. S. Tan, S. L. Goh, G. Kendall, and N. R. Sabar, "A survey of the state-of-the-art of optimisation methodologies in school timetabling problems", *Expert Syst. Appl.*, Vol. 165, No. August 2020, p. 113943, 2021, doi: 10.1016/j.eswa.2020.113943.

[5] R. A. Rahman, J. Wahid, A. A. Wahab, S. Hassan, R. Ahmad, and A. Ahmad, "Systematic Literature Review of Metaheuristic Methodologies for High School Timetabling Problem", In: *Proc. of Applied Informatics International Conference 2022 (AIIC2022)*, Penang, Malaysia, 2022.

[6] S. Ceschia, L. Di Gaspero, and A. Schaerf, "Educational Timetabling: Problems,

Benchmarks, and State-of-the-Art Results", *arXiv*, Jan. 2022, doi: 10.48550/ARXIV.2201.07525.

[7] S. Kristiansen, M. Sørensen, and T. R. Stidsen, "Integer programming for the generalized high school timetabling problem", *J. Sched.*, Vol. 18, No. 4, pp. 377–392, 2015, doi: 10.1007/s10951-014-0405-x.

[8] R. Esmaeilbeigi, V. M. Hau, J. Yearwood, and V. Nguyen, "The multiphase course timetabling problem", *Eur. J. Oper. Res.*, Vol. 300, No. 3, pp. 1098–1119, 2022, doi: 10.1016/j.ejor.2021.10.014.

[9] G. H. G. Fonseca, H. G. Santos, and E. G. Carrano, "Integrating matheuristics and metaheuristics for timetabling", *Comput. Oper. Res.*, Vol. 74, pp. 108–117, 2016, doi: 10.1016/j.cor.2016.04.016.

[10] E. Demirović and N. Musliu, "MaxSAT-based large neighborhood search for high school timetabling", *Comput. Oper. Res.*, Vol. 78, pp. 172–180, Feb. 2017, doi: 10.1016/j.cor.2016.08.004.

[11] A. Kheiri and E. Keedwell, "A Hidden Markov Model Approach to the Problem of Heuristic Selection in Hyper-Heuristics with a Case Study in High School Timetabling Problems", *Evol. Comput.*, Vol. 25, No. 3, pp. 473–501, Sep. 2017, doi: 10.1162/evco_a_00186.

[12] M. Koopialipoor and A. Noorbakhsh, "Applications of Artificial Intelligence Techniques in Optimizing Drilling", *Emerging Trends in Mechatronics*, IntechOpen, 2020.

[13] U. R. Teixeira, M. J. F. Souza, S. R. D. Souza, and V. N. Coelho, "An Adaptive VNS and Skewed GVNS Approaches for School Timetabling Problems", *Handbook of Metaheuristics*, Vol. 146, M. Gendreau and J. Y. Potvin, Eds. Boston, MA: Springer US, pp. 101–113, 2019.

[14] M. Sørensen and F. H. W. Dahms, "A Two-Stage Decomposition of High School Timetabling applied to cases in Denmark", *Comput. Oper. Res.*, Vol. 43, No. 1, pp. 36–49, 2014, doi: 10.1016/j.cor.2013.08.025.

[15] G. H. G. Fonseca, H. G. Santos, and E. G. Carrano, "Late acceptance hill-climbing for high school timetabling", *J. Sched.*, Vol. 19, No. 4, pp. 453–465, 2016, doi: 10.1007/s10951-015-0458-5.

[16] G. Dueck, "New Optimization Heuristics", *J. Comput. Phys.*, Vol. 104, No. 1, pp. 86–92, 1993, doi: 10.1006/jcph.1993.1010.

[17] K. L. Du and M. N. S. Swamy, *Search and Optimization by Metaheuristics*. Cham: Springer International Publishing, 2016.

[18] A. Baykasoglu, "Design optimization with chaos embedded great deluge algorithm", *Appl. Soft Comput.*, Vol. 12, No. 3, pp. 1055–1067, Mar. 2012, doi: 10.1016/j.asoc.2011.11.018.

[19] L. N. Ahmed, E. Özcan, and A. Kheiri, "Solving high school timetabling problems worldwide using selection hyper-heuristics", *Expert Syst. Appl.*, Vol. 42, No. 13, pp. 5463–5471, 2015, doi: 10.1016/j.eswa.2015.02.059.

[20] A. Mushi, "Non-Linear Great Deluge Algorithm for Tanzanian High Schools Timetabling", *Int. J. Adv. Res. Comput. Sci.*, Vol. 2, No. 4, 2011.

[21] A. R. Mushi, "The Application of Late Acceptance Heuristic Method for the Tanzanian High School Timetabling Problem", *Tanzania J. Sci.*, Vol. 46, No. 1, pp. 9–18, 2020, [Online]. Available: http://journals.udsm.ac.tz/index.php/tjs 9.

[22] G. H. G. D. Fonseca, H. G. Santos, T. Â. M. Toffolo, S. S. Brito, and M. J. F. Souza, "GOAL solver: a hybrid local search based solver for high school timetabling", *Ann. Oper. Res.*, Vol. 239, No. 1, pp. 77–97, 2016, doi: 10.1007/s10479-014-1685-4.

[23] K. Eng, A. Muhammed, M. A. Mohamed, and S. Hasan, "A hybrid heuristic of Variable Neighbourhood Descent and Great Deluge algorithm for efficient task scheduling in Grid computing", *Eur. J. Oper. Res.*, Vol. 284, No. 1, pp. 75–86, 2020, doi: 10.1016/j.ejor.2019.12.006.

[24] B. Arbaoui, J. Wahid, and S. A. Rahman, "Enhancing the Sorting Layers in the Initial Stage of High School Timetabling", In: *Proc. of 2020 IEEE 10th Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, 2020, pp. 59–63, doi: 10.1109/ISCAIE47305.2020.9108804.

[25] B. Arbaoui, J. Wahid, and S. A. Rahman, "The School Timetabling Problem at a Malaysian School: A Real-World Datasets and Solutions", In: *Proc. of 2nd International Conference on Design Innovation, Social Science & Technology 2021 (ICDISST2021)*, Langkawi, Malaysia, 2021.

[26] K. L. Eng, A. Muhammed, M. A. Mohamed, and S. Hasan, "A hybrid heuristic of Variable Neighbourhood Descent and Great Deluge algorithm for efficient task scheduling in Grid computing", *Eur. J. Oper. Res.*, Vol. 284, No. 1, pp. 75–86, 2020, doi: 10.1016/j.ejor.2019.12.006.

[27] M. K. Alsmadi *et al.*, "Cuckoo algorithm with

great deluge local-search for feature selection problems", *Int. J. Electr. Comput. Eng.*, Vol. 12, No. 4, p. 4315, 2022, doi: 10.11591/ijece.v12i4.pp4315-4326.

[28] N. S. Jaddi, S. Abdullah, and M. Z. A. Nazri, "A Recurrence Population-based Great Deluge Algorithm with Independent Quality Estimation for Feature Selection from Academician Data", *Appl. Artif. Intell.*, Vol. 35, No. 13, pp. 1081–1105, 2021, doi: 10.1080/08839514.2021.1972253.