Towards a Many-Objective Optimiser for University Course Timetabling

James Sakal(✉), Jonathan Fieldsend, and Edward Keedwell

University of Exeter, Exeter, UK
{js1188,J.E.Fieldsend,E.C.Keedwell}@exeter.ac.uk

Abstract. The University Course Timetabling Problem is a combinatorial optimisation problem in which feasible assignments of lectures are sought. Weighted sums of violations of various constraints are used as a quality measure, with lower scores (costs) being more desirable. In this study, we develop a domain-specific many-objective optimiser, based on constructive heuristics and NSGA-III, in which the violations of different constraints are cast as separate objectives to be minimised concurrently. We show that feasible solutions can be attained consistently in a first phase and that a targeted objective can be fully optimised in a second phase. A set of non-dominated solutions is returned, representing a well-spread approximation to the Pareto front, from which a decision maker could ultimately choose according to *a posteriori* preferences.

Keywords: Many-objective · Optimisation · Timetabling

1 Introduction

The generalised University Course Timetabling Problem (UCTP) is the task of generating a workable university timetable by assigning lectures to discrete locations in time and space, subject to various constraints. It is a well studied problem in combinatorial optimisation and known to be computationally hard [19]. This study works with the standard curriculum-based formulation proposed by the International Timetabling Competition (ITC) 2007 Track 3 under the popular UD2 configuration [4,9]. While it is noted in [3] that all (unique) instances of this benchmark but 3 have been solved to optimality, this does not diminish its usefulness. The formulation remains challenging for optimisers running on short-to-medium timeouts, while prior knowledge of the optimal values helps to contextualise results. The reader is directed to the sources above for an in-depth description of the problem and constraints, which are modelled on the real world timetabling problem of the University of Udine. In brief, feasible timetable solutions cannot violate any of five given hard constraints **h1** ... **h5**. These ensure that all lectures are assigned, pre-designated unavailable periods are avoided, as are clashes between lectures. The quality of a feasible solution is determined by violations of four soft constraints, **s1** ... **s4**, which relate to room capacity, minimum working days, curriculum compactness and room consistency respectively.

P. Legrand et al. (Eds.): EA 2022, LNCS 14091, pp. 133–144, 2023.
https://doi.org/10.1007/978-3-031-42616-2_10

We use the following notation to refer to entities in the benchmark instances: $\mathcal{L}$ is the set of lectures $\{l_1 \dots l_\gamma\}$, d_i a day of the week, t_i a timeslot within a day, $p_i = t \times d$ a period (or timeslot within a week), r_i a room. Adopting the terminology used in [16], a room/period pair is referred to as a *place*. This study proposes a parameterless many-objective optimiser based on the non-dominated sorting genetic algorithm III (NSGA-III) [6] and a constructive heuristic. The motivation is to evolve a set of solutions that approximate the Pareto front, thereby giving a decision maker a set of high quality timetables to select from. For efficiency, our approach incorporates δ-evaluators, as suggested by [12]. Phase 1 of the approach aims to find feasible starting solutions, which are then used to initialise the genetic algorithm in Phase 2. Here, the 4 soft constraint violation scores are cast as separate objectives to be minimised concurrently.

Section 2 provides some background work before Sect. 3 details the methodology and optimiser development. Section 4 describes the experiments and results. Sections 5 and 6 feature a discussion and conclusions respectively.

2 Related Work

While results have been published by many authors for the ITC2007 benchmark (see the Benchmark Analysis section of [13] for an incomplete list), the majority treat the problem as a single-objective minimisation, as prescribed by the original competition rules. The original Track 3 competition included five finalists [17], Z. Lu et al, [2,10] and [5], from which the multi-phase constraint-based solver of [17] was declared the winner. In the intervening years, the current best known single-objective results have been achieved by [1] and [15]. The former employed a hybrid genetic algorithm with Tabu Search, whose movement through the search space was determined by a sequence of large neighbourhood operators. The latter embedded an Adaptive Large Neighbourhood Search within a Simulated Annealing framework. The best known results are reproduced here for context.

It is noted in [14] that this single-objective approach predominates in educational scheduling generally, despite the existence of often numerous and conflicting objectives. The authors consider a 3–objective professional training scheduling problem with some similarities to the UCTP, comparing NSGA-II with NSGA-III. The former was found to be superior on all metrics except speed. However, the parameter values were tuned only for NSGA-II, and our problem has a higher-dimensional objective space which may be tackled better by NSGA-III. Other differences between the UCTP and the problem in [14] must be noted too, such as its timescale (repeating week-long blocks rather than months or years), requirement to assign all events, and lack of precedence constraints.

A more direct comparison may be made with [11], in which the many-objective nature of the UCTP and ITC2007 benchmark was considered. A trajectory search was carried out by selecting a small number of lectures and reassigning them. Various acceptance criteria were relied upon for the new evaluations. In both of the two approaches proposed, decision maker preferences were

assumed *a priori* and implied by the cost function. This was defined as either the standard weighted sum of violations or the Chebyshev distance to a reference point (the origin). Using the latter resulted in a more even spread of scores across individual objectives.

To the best of our knowledge, there are as yet no published results for the benchmark that attempt to approximate the Pareto set in the absence of decision maker preferences. The following section outlines the development and reasoning behind the different components of our system.

3 Methodology

Encoding: Our system is built in MATLAB and incorporates modules from the platEMO optimisation suite [20]. Its first task is re-encoding the problem instances, by converting each problem from its original `.ctt` file format to a 2-D indexed cell array data structure.

Solutions to the problem — the timetables themselves — must also be encoded. This is a design choice with serious implications for the efficacy of any evolutionary algorithm used. The proposed solution encoding represents each assignment using the 3-tuple: $\langle d_i, t_i, r_i \rangle$, where d_i and t_i are the day and timeslot respectively and the element-wise length of a complete chromosome is $3 \times |\mathcal{L}|$. Disadvantages of using a 3-tuple include the larger data structure and higher time complexity involved, as well as the potential for epistatic effects caused by interactions between elements within tuples. More favourably, the induced search landscape grants connectivity between days, timeslots and rooms as individual entities, allowing for the design of more nuanced and effective genetic operators. Each element within a gene resonates with a particular soft constraint. For example, perturbing d_i affects the number of unique days that course lectures are held on, and therefore the violation score of **s2**. Compliance with **h1** (all lectures must be assigned) is also ensured by the 1:1 lecture:gene ratio.

Initialisation: The initialisation constitutes Phase 1 of a two-phase optimisation, with the aim being to produce a population of solutions that is as close to fully feasible as practicable. To this end, two broad categories of constructive heuristics have been proposed in the literature [18]. Static heuristics require lectures to be sorted by some metric, where this fixed ordering then determines the sequence of assignments. Dynamic heuristics involve recalculating the metric values after each assignment, thus providing greater adaptive potential. In both cases, the chosen metric is intended as a measure of 'difficulty to assign'.

The static heuristics Largest Enrolment (LE) and Largest Degree (LD) and the dynamic heuristic Saturation Degree (SD) were tested on the ITC2007 benchmark. LE relies on the number of enrolled students for its metric. Lectures with a larger number of students take priority. LD, as described for the generic case in [18], uses the number of potential clashes a lecture has with other lectures resulting from commonality of students. Since explicit student sectioning is not a feature of the ITC2007 benchmark, the metric is defined analogously as: The sum total of lectures that have either a curriculum or a teacher in common with

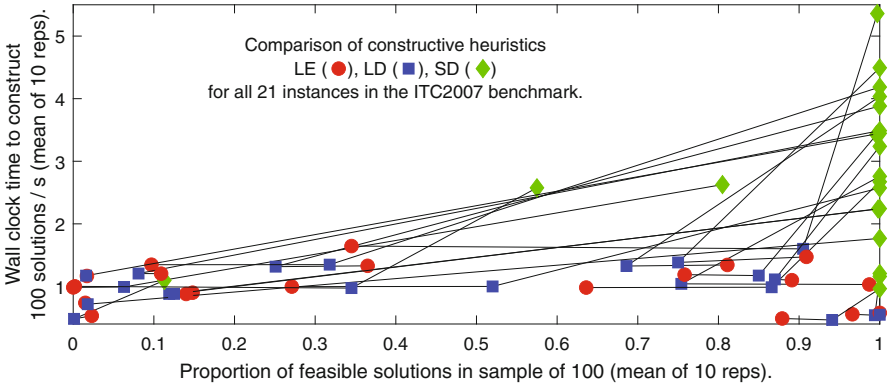

Fig. 1. Performance comparison of 3 constructive heuristics. Lines connect results for common instances.

the lecture being assessed. Priority is given to lectures with higher numbers of potential clashes in this respect. The metric for SD is the number of available feasible places, i.e. those that would not result in a hard constraint violation at the point of assignment. The lecture with the lowest value at each decision point is chosen for assignment. Across all heuristics, ties are broken at random.

Once a lecture has been chosen on the basis of its metric value, a place is randomly selected from the set of feasible places currently available to that lecture. If no feasible place exists, an infeasible place (excluding unavailable periods) is chosen at random instead. A secondary, period-based heuristic is suggested in [18] as an optional, more discriminatory, alternative to random sampling. Our system neglects to include this with the following justification: Any infeasible solutions that may have been constructed in Phase 1 are quickly bred out of the population by the inherent hard constraint handling mechanism. The extra expense of a period-based heuristic was therefore found to outweigh the marginal gains in feasibility rate.

In testing LE, LD and SD, 10 independent repetitions were carried out for each problem instance. In each repetition, 100 timetable solutions were constructed. This number was chosen to reflect the order of magnitude of a typical population. The primary quality measures to consider are the proportion of solutions that are feasible, and the relative speed of obtaining them. As with all experiments in this study, the computation was performed on a 12-core Ryzen9 with 32GB RAM, base clock speed 3.8GHz. The wall clock speed shown here resulted from using a single core and no parallelisation. Figure 1 shows the results for the three heuristics over the 21 instances.

SD achieves superior feasibility rates for every instance, while being computationally dearer. At the scale of a population size of 100, this additional time cost amounts to no more than a few seconds. More pertinently, all SD rates are 0.99 or higher, with the exception of the 3 instances `comp02` (0.80), `comp05` (0.11) and `comp19` (0.58). Across the infeasible solutions constructed for these 3 problems,

the mean distances to feasibility, given as a vector of the hard constraints (**h2 h3**, **h4**, **h5**), were (0.3, 3.1, 0.0, 0.2), (0.2, 18.6, 0.0, 0.2) and (0.5, 3.8, 0.0, 0.2) respectively. These show that in the minority of cases where SD fails to achieve a near-perfect feasibility rate, the expected violations of hard constraints in the infeasible subset are nonetheless low. In particular, **h4** is zero in all cases.

Besides feasibility and speed, there may be other factors to consider when assessing the quality of an initial population generated by a constructive heuristic. The percentage of unique individuals in the sample is one example. In the aforementioned tests, 100% was achieved across all instances and all heuristics on this measure. Additionally, it may be worth considering some measure of dispersion or dissimilarity between individuals. A suitably diverse starting population may be important in terms of the exploratory power of the optimiser.

Algorithm: NSGA-III is a successful evolutionary algorithm that supports many-objective optimisation with constraints [6]. It is an extension to the popular NSGA-II algorithm, which was originally conceived for lower-dimensional objective spaces [7]. As the ITC2007 problem has 4 objectives to optimise, NSGA-III serves as an appropriate base for Phase 2 of our system.

Selection and constraint handling: Alongside the initialised population, the SD heuristic implementation returns an array of feasibility flags, toggled during construction. The property `con` holds the flag associated with each solution, with a `true` value indicating at least one violation of a hard constraint. For the first generation only, scores for the four soft constraint objectives are then calculated in full. 2-way tournament selection is used to select a mating pool. Randomly paired candidate solutions are first compared on their `con` property, with the lower value indicating the winner. Feasible solutions are thereby given priority. Should the `con` values be equal, the sum of the objective scores is used as a tie-breaking fitness measure.

Genetic operators: For a real-valued encoding, NSGA-III traditionally uses simulated binary crossover (SBX) and polynomial mutation as its genetic operators. For this discrete problem, adaptations were first made to both genetic operators to ensure the preservation of integrality in decision space. Further investigation determined that, with no meaningful ordering apparent for entities such as days or periods, traditional polynomial mutation is not necessarily well suited for this problem domain. Similarly, standard SBX carries the risk of

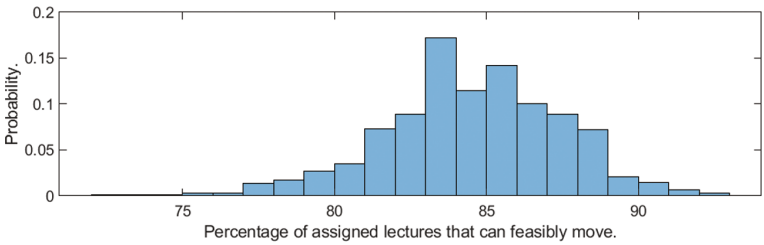

Fig. 2. A histogram of the percentage of assigned lectures with at least one feasible move available, for `comp12`. The sample set is 1000 feasible solutions constructed by SD.

degenerating timetables by recombining promising subsets in an injudicious way, thereby worsening the overall solution quality. In complex, combinatorial problems such as timetabling, a successful crossover operator requires domain-specific knowledge and can be computationally expensive. The proposed approach therefore dispenses with crossover entirely and is instead wholly reliant on a guided mutator. In developing this mutator, the following test was conducted:

1000 feasible solutions were constructed using SD. For each assigned lecture of each solution, a check was made on the number of places it could be re-assigned to without violating the overall solution feasibility. For some assignments, there were no feasibility-preserving moves available. The histogram in Fig. 2 shows an example (for `comp12`) of the distribution of percentages of assigned lectures, over the 1000 solution sample, with at least one such available move.

For all problems tested, the distributions demonstrate that the expected chance of an available feasible move is generally high. The optimiser can be guided, therefore, by imbuing the initial mutator, known as `MuPF`, with a preference for feasible moves where they exist. After randomly selecting one lecture, l_i, to be mutated, another random selection is made from the set of feasible moves available to that lecture. If this set is found to be empty, `MuPF` defaults the assignment to any random place.

Using this mutator, a test run was performed on `comp01` with a population size of 364 over 550 generations. Over the course of this run, the minimum values of objectives (**s1**, **s2**, **s3**, **s4**) improved from (1599, 15, 88, 66) to (537, 0, 6, 28) respectively. Further tests emphasised the large relative contribution that **s1** often makes to a scalarised objective score. An enhancement to the mutator, in which sufficient room capacity is considered, was proposed specifically to target this objective. Algorithm 1 outlines `MuPFPR`.

An initial indicative plot comparing `MuPF` and `MuPFPR` is given in Fig. 3. A run on `comp01` was carried out with a function evaluation budget of 2 million. The

Algorithm 1: Preference for feasibility, preference for room (`MuPFPR`) mutation operator

Inputs: One starting solution
Output: One mutated solution
Randomly select a lecture, l_i, to mutate
Identify the set of places, `feasMoves`(l_i), to which l_i can be re-assigned without violating the feasibility of the solution
if $feasMoves(l_i) = \emptyset$ **then**
    Re-assign l_i to a new randomly chosen place in any room with sufficiently high capacity and excluding unavailable periods
else
    if $feasMoves(l_i) \cap sufficientRooms(l_i) = \emptyset$ **then**
        Re-assign lecture i to a place randomly chosen from `feasMoves`(l_i)
    else
        Re-assign l_i to a place randomly chosen from the given non-empty intersection

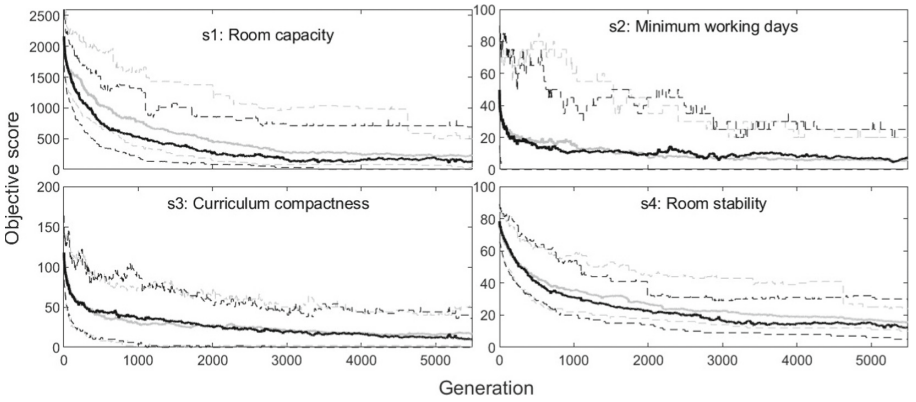

Fig. 3. A comparison of mutator `MuPF` (grey) and `MuPFPR` (black) for a single rep of `comp01` with 2 million function evaluations. Traces shown are the min, mean and max objective scores over each generation.

extra room-related guidance provided by `MuPFPR`, shown as a black trace, helped drive the convergence rate for **s1** objective in the top left tile, at no detriment to the remaining objectives.

Incorporated into the mutation process is an implicit feasibility checker. A violation flag, `conMutation`, is toggled if and only if $\texttt{feasMoves}(i) = \emptyset$. The returned `con` property for that child is generally given by (`conParent` $\vee$ `conMutation`) — except in the case when the parent solution is infeasible and the mutation is feasible. Here, the feasibility of the child is unknown and a full evaluation of the hard constraints must be called. The rarity of this outcome ensures that, in practise, the hard constraint evaluators seldom need to be executed at all — an example of a time-saving partial evaluation. The following section details how δ-evaluations are used to make similar savings when calculating the soft constraint objectives.

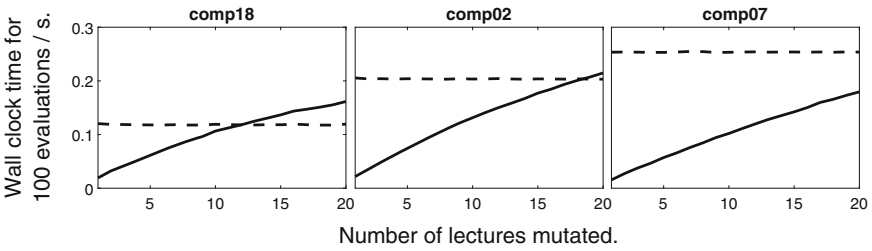

Fig. 4. A comparison of the time complexity (mean of 10 reps) for the combined δ-evaluators (solid line) vs. full (dashed line), for a small (`comp01`), medium (`comp02`) and large (`comp07`) sized problem and a variable number of mutations.

δ-evaluations: The process by which a δ-evaluation negates the need for a full evaluation on the soft constraint objectives is as follows: The ID of the lecture to be perturbed is recorded. The value contributed to the parent objective score by the assignment of this lecture is calculated. This value is subtracted from the objective score of that parent, which is known *a priori* from the previous generation. Lastly, the contribution of the new assignment in the child solution is added. Objective **s1** is best suited for a fast δ implementation, due to the fact that the value contributed by an individual lecture is independent of those from other lectures. For the remainder of the objectives, interactions between the lecture being perturbed and various other lectures must also be accounted for. Specifically, those from the same course (for **s2** and **s4**), or those with a common curriculum (**s3**). Combined over 4 objectives, the δ-evaluators nonetheless offer a sizeable time saving over their full counterparts, as illustrated in Fig. 4. While the run time of a full evaluator scales with the number of lectures, the δ run time scales with the number of mutations — due to the resulting combinatorial interactions. Under a single lecture mutation, the δ-evaluator gives the largest time savings, by multiples of 6.3, 10.7 and 13.2 for the respective problems shown.

Non-dominated sorting: NSGA-III relies initially on the dominance relation on objective scores to sort a concatenated parent/offspring population into non-dominated fronts. The efficient non-dominated sort with sequential search (ENS-SS) is used [21]. The hard constraint handling procedure mandates that any solution with a `con` flag value `true` is automatically dominated by all feasible solutions, regardless of the quality of its objective vector. The only way, therefore, in which such a solution can be admitted into the next generation is if the cardinality of the feasible solution set is less than the active population size. This in turn implies the following about Phase 2: If a given generation is fully feasible, all subsequent generations are also fully feasible. To promote diversity, NSGA-III also associates solutions with rays passing through a set of `popSize` uniformly distributed points on the 4-dimensional unit hyperplane. The normal-boundary intersection method with two layers is used to obtain these coordinates. `popSize` is a geometrically constrained approximation to the desired, user-input population size, `setPopSize`.

4 Experimental Design and Results

Each run of the optimiser was allocated to a single core of the Ryzen9 machine, as per the original ITC2007 stipulation. Parallelisation was used only across independent runs. In the absence of the original CPU benchmarking program, termination was after 600 s wall clock time, which was the limit intended by the competition, and `setPopSize` = 100. For each problem in a subset of 10 tested, 30 repetitions were carried out by varying the random seed. An external passive archive, implementing the ND-Tree structure [3,8], was constructed using the complete search history. The purpose was to update and store the set of non-dominated solutions found over the course of the search. The results are reported in terms of the following performance metrics: The best scalarised

score found (using the original ITC2007 weighted penalty scheme). The size, at termination, of unique solutions in the non-dominated archive (both in decision and objective space, as the mapping is many-to-one). A Monte Carlo estimate of the hypervolume indicator, for which theoretical upper bounds on the maximum objective scores were used as the reference point coordinates.

Table 1 shows our results and statistics, alongside results from [1,11] and [15]. Figure 5 illustrates the spread of non-dominated solutions achieved by a single rep in 3-D objective space, for 3 problems in which the **s1** dimension has successfully been collapsed to zero.

Table 1. Results from 30 independent reps. b_s is the best scalarised solution score found over all reps, while b_s(**s1**, **s2**, **s3**, **s4**) gives the objective scores that make it up (averaged over the unique objective vectors whose sum is b_s). $\mathcal{A}$ is the final archive of non-dominated solutions, where sets of unique vectors in objective or decision space are distinguished by subscripts $_o$ and $_d$ respectively. Cardinalities for both are given as median values. $hv(\mathcal{A}_o)$ is the (mean) hypervolume of $\mathcal{A}_o$, while HV_{ref} is the reference point used. The best scalarised results from the two approaches in [11] are given as G1 (Threshold Accepting with 1% threshold) and G2 (reference point based). Finally, BK denotes the best known single-objective scores to date within the time limit, achieved by either [1]* or [15]† or both.

Instance	Proposed approach						Others		
	b_s	b_s(**s1**, **s2**, **s3**, **s4**)	$\lvert\mathcal{A}_o\rvert$	$\lvert\mathcal{A}_d\rvert$	$hv(\mathcal{A})$	HV_{ref}	G1	G2	BK
comp01	11	(4, 0, 4, 3)	11	7492	0.959	(3606, 360, 294, 124)	5	10	5*†
comp03	162	(0, 52.5, 92, 17.5)	17	850	0.831	(11160, 720, 1536, 179)	115	154	68†
comp04	92	(0, 6.7, 65.3, 20)	17	482	0.853	(8151, 665, 1130, 207)	67	90	35*†
comp06	167	(0, 15, 104, 48)	16	233	0.777	(10632, 990, 1668, 253)	94	159	30*
comp08	108	(0, 0, 74, 34)	14	301	0.810	(7711, 700, 1166, 238)	75	120	37*
comp09	158	(0, 40, 94, 24)	24	623	0.821	(9269, 720, 1492, 203)	153	197	100†
comp11	0	(0, 0, 0, 0)	2	45453	0.981	(3196, 335, 500, 103)	0	0	0*†
comp13	131	(0, 30, 84, 17)	20	390	0.832	(10668, 670, 1292, 226)	101	133	59*†
comp14	125	(0, 20, 90, 15)	19	1289	0.866	(7138, 830, 1392, 190)	88	120	51†
comp18	116	(0, 30, 78, 8)	45	1373	0.884	(2638, 455, 954, 91)	n/a	n/a	64†

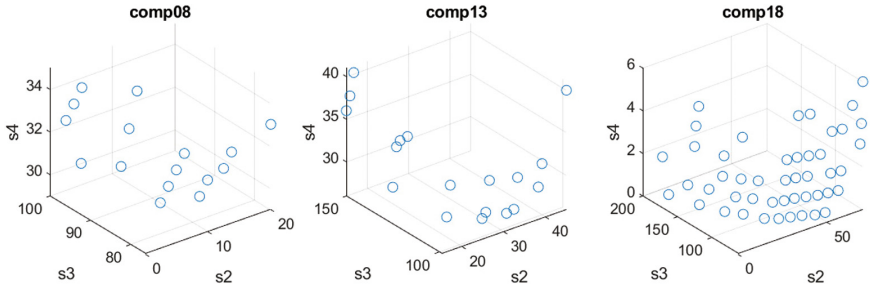

Fig. 5. Non-dominated solution sets in (**s2**,**s3**,**s4**)-space, found during single runs for 3 problems in which the fourth objective, **s1**, was optimised to zero.

5 Discussion

The strategy for speeding up (or by-passing) calculation of objective scores was successful in yielding inexpensive evaluations. However, this only partially mitigated against the cost of non-dominated sort. The algorithm unsurprisingly had a lower execution rate for function calls than many single-objective solvers. Comparing its performance on an equal function evaluation budget rather than a time budget would be enlightening, as the gradients in Fig. 3 suggest further gains are available. Despite this, scalarised results are seen to approach those of single-objective solvers on some problems which is encouraging — `comp11` in particular was solved to optimality. With regard to the individual objective scores, the targeted operator `MuPFPR` was capable of rapidly optimising **s1** to zero across the board (except for `comp01` where the value of **s1** in the optimal solution is known to be 4). These gains were not made at the expense of other objectives however, which showed improvement without exception during the runs. This suggests that additional bespoke operators, targeted at these objectives, may be a promising next step in striving to closer approximate the true Pareto front. A comparison with the reference point based approach of [11] (G2), shows competitive or improved scalarised scores, although this claim is weakened by the CPU benchmarking discrepancy. A major point of differentiation though is that our approach returns a population per run, rather than a single solution, in a comparable timescale. The approach appears relatively problem-agnostic, in contrast to [12] whose results show high variance across problems. Most importantly, it works on the assumption of *a posteriori* decision maker preferences. Different areas and extremes of the Pareto front are therefore explored simultaneously and a well-spread set of non-dominated solutions can be provided, as shown in Fig. 5. The hypervolume indicator values in Table 1 also evidence this, with all 10 problems, bar `comp06`, achieving a mean of 0.82 or higher. As lower absolute objective scores are achieved, the cardinality $|\mathcal{A}_o|$ naturally tends to decrease, as in `comp01` (median 11) and `comp11` (2). This can be explained by the proximity of the front to the origin and consequent sparsity of distinct points on the 4-D integer lattice. The observation $|\mathcal{A}_d| \gg |\mathcal{A}_o|$ also interestingly highlights the extent to which multiple designs map to a common objective point.

6 Conclusions and Further Work

In a departure from the single-objective treatment of the ITC2007 timetabling problem, we propose a two-phase, many-objective optimiser based on NSGA-III in which hard constraints are handled procedurally and soft constraints are cast as objectives. It is effectively paramaterless, save for `setPopSize` and termination criteria which are pragmatic user choices. The time cost associated with many-objective algorithms is mitigated by prudent use of δ-evaluators. A simple mutation operator reduces the otherwise large violation contributions caused by over-filling rooms (constraint **s1**) to zero wherever possible. Selection and non-dominated sorting ensure convergence of the other objectives as well as feasibility of solutions, while a quick start is guaranteed by the SD constructive heuristic.

Further work will focus on increasing the convergence speed of the remaining 3 objectives by widening the pool of targeted operators. If the mutator is considered as a neighbourhood, a more systematic exploration may be possible. Figure 2 gives an intuition about the size of such a neighbourhood. An adaptive element may be added to Phase 2 to select from such a pool based on the state of the current population or trajectory of the evolution. Alternatively, objectives that reach optimality may be aggregated with `con` so that any solutions sub-optimal in this objective will thereafter be automatically dominated. Further analysis will also help characterise the trade-offs between the objectives. By their definitions, **s1**/**s4** and **s2**/**s3** represent the two pairs with the greatest potential to conflict. The large cardinalities of the decision space solution sets suggests that genotype diversity could also play a useful role in the selection process.

References

1. Abdullah, S., Turabieh, H.: On the use of multi neighbourhood structures within a tabu-based memetic approach to university timetabling problems. Inf. Sci. **191**, 146–168 (2012)
2. Atsuta, M., Nonobe, K., Ibaraki, T.: ITC-2007 Track 2: An approach using general CSP solver. In: Proceedings of the Practice and Theory of Automated Timetabling (2007)
3. Bagger, N.C.F., Sørensen, M., Stidsen, T.R.: Dantzig-Wolfe decomposition of the daily course pattern formulation for curriculum-based course timetabling. Eur. J. Oper. Res. **272**(2), 430–446 (2019)
4. Bonutti, A., De Cesco, F., Di Gaspero, L., Schaerf, A.: Benchmarking curriculum-based course timetabling: formulations, data formats, instances, validation, visualization, and results. Ann. Oper. Res. **194**(1), 59–70 (2012)
5. Clark, M., Henz, M., Love, B.: QuikFix a repair-based timetable solver. In: 7th International Conference on the Practice and Theory of Automated Timetabling, PATAT 2008 (2008)
6. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point based non-dominated sorting approach, part i: solving problems with box constraints. IEEE Trans. Evol. Comput. **18**(4), 577–601 (2013)
7. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
8. Fieldsend, J.E.: Data structures for non-dominated sets: implementations and empirical assessment of two decades of advances. In: GECCO 2020 - Proceedings of the 2020 Genetic and Evolutionary Computation Conference, pp. 489–497 (2020)
9. di Gaspero, L., Schaerf, A., McCollum, B.: The second international timetabling competition: curriculum-based course timetabling (Track 3). In: Proceedings of the 1st International Workshop on Scheduling a Scheduling Competition (2007)
10. Geiger, M.J.: An application of the threshold accepting metaheuristic for curriculum based course timetabling. In: Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling (PATAT) (2008)

11. Geiger, M.J.: Multi-criteria curriculum-based course timetabling—a comparison of a weighted sum and a reference point based approach. In: Ehrgott, M., Fonseca, C.M., Gandibleux, X., Hao, J.-K., Sevaux, M. (eds.) EMO 2009. LNCS, vol. 5467, pp. 290–304. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01020-0_25
12. Geiger, M.J.: Applying the threshold accepting metaheuristic to curriculum based course timetabling. Ann. Oper. Res. **194**(1), 189–202 (2012)
13. Gozali, A.A., Fujimura, S.: Solving university course timetabling problem using multi-depth genetic algorithm. SHS Web Conf. **77**, 01001 (2020)
14. Hafsa, M., Wattebled, P., Jacques, J., Jourdan, L.: A Multi-objective evolutionary approach to professional course timetabling: a real-world case study. In: 2021 IEEE Congress on Evolutionary Computation, pp. 997–1004 (2021)
15. Kiefer, A., Hartl, R.F., Schnell, A.: Adaptive large neighborhood search for the curriculum-based course timetabling problem. Ann. Oper. Res. **252**(2), 255–282 (2017)
16. Lewis, R., Paechter, B., Rossi-Doria, O.: Metaheuristics for university course timetabling. Stud. Comput. Intell. **49**, 237–272 (2007)
17. Müller, T.: ITC2007 solver description: a hybrid approach. Ann. Oper. Res. **172**(1), 429–446 (2009)
18. Pillay, N., Özcan, E.: Automated generation of constructive ordering heuristics for educational timetabling. Ann. Oper. Res. **275**(1), 181–208 (2019)
19. Rossi-Doria, O., et al.: A comparison of the performance of different metaheuristics on the timetabling problem. Pract. Theory Autom. Timetabling IV **2740**, 329–351 (2003)
20. Tian, Y., Cheng, R., Zhang, X., Jin, Y.: PlatEMO: A MATLAB platform for evolutionary multi-objective optimization. IEEE Comput. Intell. Mag. **12**(4), 73–87 (2017)
21. Zhang, X., Tian, Y., Cheng, R., Jin, Y.: An efficient approach to nondominated sorting for evolutionary multiobjective optimization. IEEE Trans. Evol. Comput. **19**(2), 201–213 (2015)