Erasmus University Rotterdam
Erasmus School of Economics
Bachelor Thesis Operations Research

A max-min ant system heuristic to solve the hourly learning activity planning problem in personalised learning

Anouk Luijben (540239)

Supervisor:	Danny (Jia Hui) Zhu
Second assessor:	prof. dr. D. Huisman.
Date final version:	2nd July 2023

Contents

Abstract

An increased popularity in personalised learning methods for students has prompted the development of learner-based timetabling methods. However, achieving an exact solution for the Hourly Learning Activity Planning problem, which involves the generating of personalised learning schedules, requires impractical computation times. In this research, the Max-Min Ant System is proposed as a heuristic to solve the problem. The Adaptive Large Neighborhood Search heuristic is used as benchmark to evaluate its performance. We conclude that the Max-Min Ant System outperforms this heuristic with an average optimality gap of 1.7% against 2.2%, although its computation times are slightly longer. In particular, for schools with high learners demand spread or those that have split their classrooms, the Max-Min Ant System emerges as the optimal heuristic.

1 Introduction

Personalised learning (PL) is an educational approach that focuses on students' needs and progress. It is the opposite of the conventional approach of course scheduling, which is based on students all following the same curriculum. PL takes into account differences between individual learners, based on the notion that learning curves, the rate of students' progress, vary from student to student (Kalyuga et al., 2003). Technological advancements in data science and machine learning the last decades have led to numerous timetabling methods and therefore in an increase in the popularity of PL (Shemshack and Spector, 2020). The classic scheduling method, that generates long-term schedules per school class, appears to be a poor fit for PL, hence a short-term and learner-based method provides a solution. The European Political Strategy Center of the European Commission emphasises the importance of PL and mentions it as one of the trends in transforming education (European Commission and European Political Strategy Centre, 2019). According to the commission, it is unlikely that the one-size-fits-all learning keeps working in this modern-day economy that is built around human capital. Instead, individualised learning paths take into account that students approach problems in their own way and acquire knowledge and skills at their own pace.

The corresponding problem, which aims to create a set of activities every hour while satisfying as much demand as possible, is the Hourly Learning Activity Planning Problem (HLAPP). With the generated set of activities a schedule can be made that is personalised for every student. Those schedules can be useful when long-term (e.g. a week) learner demands are known and an initial schedule per student is already created. When rescheduling is needed, due to changed learner demands or availability of teachers and classrooms, the method can update the initial plan.

Wouda et al. (2023) concluded that solving the HLAPP as an integer linear programming model to optimality involved impractical computation times. Therefore, they introduced an efficient adaptive large neighborhood search (ALNS) metaheuristic to solve the problem. The heuristic starts with creating a solution and subsequently destroys and repairs it using several techniques. This process is repeated and guides the method to a high-quality solution. According to Wouda et al. (2023), ALNS ensures that the rescheduling is done within a few minutes with an average optimality gap of 1.6%.

In this paper, we introduce the Max-Min Ant System (MMAS) metaheuristic to solve the HLAPP. This algorithm is an extension of the Ant Colony optimisation technique and originally introduced by Stützle and Hoos (1996). Ant colony optimisation is based on ants searching for food, while communicating with each other with the help of an invisible chemical named pheromone. Every ant starts moving randomly, but after finding food it returns back to the colony while leaving pheromone trails on its way. When encountering a trail, other ants tend to follow it, although those trails eventually evaporate. The evaporation process ensures the possibility for discovering new sources of food. Previous research shows promising results, for example when applied to the traveling salesman problem (Jangra and Kait, 2017). Besides, Stützle and Hoos (2000) stated that the MMAS is currently among the best performing algorithms for solving the traveling sales problem and the quadratic assignment problem.

In order to evaluate the relative performance of the MMAS, the ALNS heuristic and its results are used as benchmark. The two heuristics are compared in general and parameter-specific on a wide-ranging set of experiments using the relative gap with the optimal objective as measure. The MMAS shows superior results in terms of optimality gaps, since the method exhibits an average optimality gap of 1.7% while ALNS of 2.2%. Despite its slightly longer running times compared to ALNS, the MMAS is perfectly suitable for quickly generating school schedules, endorsed by an average computation time of 145 seconds. The MMAS outperforms ALNS in particular for high learner demand spread and for schools that split their classrooms. The performance of the MMAS is most affected by adapting learner demand spread, changing the fulfilment of self-study or splitting the classrooms in a school. We conclude that the MMAS is a more effective heuristic compared to ALNS for solving the HLAPP. Hence, MMAS emerges as an effective heuristic to solve the HLAPP.

The outline of the paper is as follows. In Section 2, we give an overview of the existing literature about school scheduling models and the proposed heuristics. Next, in Section 3, the problem encountered in this research, the Hourly Learning Activity Planning problem, is described and the corresponding notation is introduced. In Section 4, the integer linear programming model is presented, which can be used to solve the problem to optimality. Thereafter, in Section 5 and 6, the methodology for two different heuristics are presented, the Adaptive Large Neighborhood Search and the Max-Min Ant Colony System, respectively. We introduce the experimental design and present the obtained results in Section 7. Finally, in Section 8 the conclusions and suggestions for future research are presented.

2 Literature review

Modeling school timetables is a frequently studied topic in the literature. However, most studies investigate methods for the conventional way of scheduling (e.g. Birbas et al. (1997) and Saviniec and Constantino (2017)), while only a few studies analyse personalised learning based timetabling problems. Pillay (2013) highlights the growing use of heuristics in high school timetabling. Thereby, numerous studies propose a heuristic as a solution to address this issue. For instance, Santiago-Mozos et al. (2005) introduce a two-phase heuristic and test it on a real problem at a Spanish university, demonstrating good results. Kristiansen et al. (2011) develop an integer programming model and solve it with an Explicit Constraint Branching heuristic.

Kannan et al. (2012) use a graph-theoretic approach. The algorithm breaks the problem in sub problems and solves them with heuristics. The mutual drawback of these studies is the assumption of unchanging student demands over the semester or year. However, Wouda et al. (2023) did include changing learner demands and obtained good results by using an Adaptive Large Neighborhood Search (ALNS) as a metaheuristic.

In this paper, we investigate the performance of another heuristic named Max-Min Ant System (MMAS) to create personalised learning schedules. This method is an extension of the Ant Colony Optimisation heuristic, which was initially introduced by Dorigo et al. (1996) and immediately obtained encouraging results for the traveling salesman problem. The heuristic constructs several feasible solutions by iteratively selecting solution components. This component selection process relies on probabilities, which are determined by pheromone levels. Good solutions are rewarded by increasing the pheromone levels of their corresponding components, thereby improving the chance of using the components in next iterations. When applied to scheduling problems, such as the vehicle routing problem, ant colony optimisation also appeared to be a successful method (Bell and McMullen, 2004). Stützle and Hoos (1996) are the inventors of the MMAS, a heuristic that improves the ant colony optimisation by using only the best solutions to spread pheromone and by setting limits to the pheromone levels which avoids premature convergence of the search.

To the best of our knowledge, the MMAS is never used to solve the HLAPP. Applying the MMAS on a scheduling problem is done by Crawford et al. (2014), who address the software project scheduling problem. They obtained high quality solutions in terms of running time and solution quality.

3 The Hourly Learning Activity Planning Problem

In this section, the Hourly Learning Activity Planning Problem (HLAPP) is described with its corresponding variables and parameters, based on Wouda et al. (2023).

3.1 Problem statement

The investigated problem in this study involves of the construction of hourly learning activity plans based on learner demands. Every student provides preferences or demands for different modules, which are used to assign students to activities. An activity is defined as the assignment of a group of learners to a module, classroom and teacher. Activities are categorised as either instruction activities, where a qualified teacher teaches a module to a group of learners who the module demand, or self-study activities, where learners work independently on a module with a teacher present solely for supervision. Each activity is part of a module that defines the precise learning topics, and each module is part of a course.

Not all classrooms are usable for each activity. For example, activities of the course informatics can only be taught in a classroom with computers and these constraints should be included in the model. Moreover, there are differences between teachers' qualifications. We make a distinction between first-degree teachers who are allowed to teach all modules in a course, second-degree teachers who are allowed to teach only the first half of the modules of a course (the less advanced

modules) and third-degree teachers who are only qualified to supervise self-study sessions. First- and seconds-degree teachers are also allowed to supervise self-study sessions.

In this problem, we incorporate the assumption that self-study sessions are less efficient than instruction activities. The fraction that self-study activities satisfy of the learner demands is a subjective parameter that schools can choose for themselves and is called the self-study fulfilment level.

In PL schools, it is likely that group sizes per activity are smaller because learners work on different activities at any time. Current school buildings probably have too large classrooms and therefore we also include the policy of splitting all existing instruction classrooms in two.

3.2 Variable and parameter notation

To model the HLAPP, some sets, parameters and variables need to be defined. The set L includes all learners l who demand certain course modules $m \in M$. The demand of learner l for module m is denoted as $D_{lm} \in \mathbb{R}_{\geq 0}$ and is zero if the learner is ineligible for module m. A teacher $t \in T$ can instruct or supervise an activity in classroom $c \in C$. The objective of the model is to assign learners to their most demanded learning modules. As mentioned in Section 3.1, self-study activities only satisfy a fraction w ($0 \leq w \leq 1$) of the demand D_{lm}. In order to keep track of the learners assigned to self-study activities, we create a module $m_S \in M$ that denotes the self-study module. The demand for this module D_{lm_S} can be calculated as $w \max_{m \in M \setminus m_S} D_{lm}$ because during a self-study session, learners are assumed to work on their most-preferred module.

As noted previously, not every teacher is qualified to teach every course. In the binary qualification matrix Q^T is specified if teacher t can teach module m, which is denoted as $Q^T_{tm} = 1$. In the same way, in the classroom binary qualification matrix $Q^C_{cm} \in \{0, 1\}$ is tracked if a classroom c can be used for a module m.

Every classroom c has its own seating capacity $N_c \in \mathbb{N}$, which is an upper bound on the maximum number of learners that can attend an activity in this classroom. Besides, $\delta^- \in \mathbb{N}$ and $\delta^+ \in \mathbb{N}$ specify the minimum and maximum number of learners needed to schedule an instruction activity respectively. For self-study activities, δ^+ is not imposed. We use a lower bound δ^- to prevent scheduling activities that are too small as this is assumed as an inefficient use of classroom space and teachers.

4 The integer linear programming model

In this section, we introduce the integer linear programming model (ILP) as a method to solve the problem to optimality. Therefore, two decision variables are required in addition to the notation of Section 3.2. The assignment of learner l to module m is modeled with the binary decision variable y_{lm}. Additionally, the binary decision variable x_{mct} indicates if an activity of module m is scheduled in classroom c, taught by teacher t. The integer linear programming model we aim to optimise is the following:

$$\max_{x,y} \quad \sum_{l \in L} \sum_{m \in M} D_{lm} y_{lm} \tag{1}$$

$$
\begin{aligned}
\text{s.t.} \quad & \sum_{l \in L} y_{lm} \leq \sum_{c \in C} \min(\delta^+, N_c) \sum_{t \in T} x_{mct} && \forall m \in M \setminus \{m_S\} && (2)\\
& \sum_{l \in L} y_{lm_S} \leq \sum_{c \in C} N_c \sum_{t \in T} x_{m_S ct} && && (3)\\
& \sum_{l \in L} y_{lm} \geq \delta^- \sum_{c \in C} \sum_{t \in T} x_{mct} && \forall m \in M && (4)\\
& \sum_{m \in M} y_{lm} = 1 && \forall l \in L && (5)\\
& \sum_{m \in M} \sum_{c \in C} x_{mct} \leq 1 && \forall t \in T && (6)\\
& \sum_{m \in M} \sum_{t \in T} x_{mct} \leq 1 && \forall c \in C && (7)\\
& \sum_{c \in C} x_{mct} \leq Q^T_{tm} && \forall t \in T, \forall m \in M && (8)\\
& \sum_{t \in T} x_{mct} \leq Q^C_{cm} && \forall c \in C, \forall m \in M && (9)\\
& y_{lm} \leq \mathbb{1}_{D_{lm} > 0} && \forall l \in L, \forall m \in M && (10)\\
& x_{mct} \in \{0, 1\} && \forall m \in M, \forall c \in C, \forall t \in T && (11)\\
& y_{lm} \in \{0, 1\} && \forall l \in L, \forall m \in M && (12)
\end{aligned}
$$

The objective (1) aims to maximise the assignment of learners to their most demanded modules. Constraints (2) concern the instruction activities and ensure that neither the maximum instruction size nor the classroom capacity is exceeded. Similarly, (3) guarantees the maximum group size is not exceeded for self-study activities, which should only take into account N_c. Constraints (4) make sure the minimum number of learners is reached in every module. Every learner should be assigned to a module, which is ensured by constraints (5). Constraints (6) and (7) guarantee that every teacher and classroom are used for at most one module, respectively. Constraints (8) and (9) use the qualification matrices to ensure that only qualified teachers and classrooms are assigned to a module activity. With the use of constraints (10), learners are only assigned to modules they are eligible to take, that is if $D_{lm} > 0$. Lastly, constraints (11) and (12) impose bounds on the binary variables x and y.

When the model is solved to optimality, a schedule can implicitly be derived with the two-dimensional variable y and the three-dimensional variable x. Due to this latter variable, the problem turns into a variant of the three-dimensional assignment problem, which is classified as *NP*-hard. Therefore, we propose metaheuristics to solve the problem instead.

5 Adaptive Large Neighborhood Search metaheuristic

In this section, the Adaptive Large Neighborhood Search (ALNS) is introduced as metaheuristic to solve the HLAPP. Røpke and Pisinger (2006) originally introduced this heuristic and later expanded upon in their work on vehicle routing problems, describing a general treatment of the metaheuristic (Pisinger and Røpke, 2010). In this research, we follow the applied variant on the HLAPP of Wouda et al. (2023). A description of the pseudo-code of the metaheuristic is provided in Algorithm 1. In short, the algorithm starts with an initial feasible solution and

iterates a predefined maximum number of times. In every iteration, it chooses a destroy and repair operator, which transform the current solution s into a candidate solution s^c. If s^c is a new best solution, the algorithm uses a Local Search procedure to improve the solution further. Finally, the operator selection mechanism is updated. An extensive description of the steps is provided in the subsequent sections.

Algorithm 1 Adaptive Large Neighbourhood Search

```
1: Input: Initial feasible solution s.
2: Output: Best observed solution s*.
3: s* := s, ρ_D := (1, ..., 1), ρ_R := (1, ..., 1).
4: repeat
5:    Select destroy and repair methods d_op ∈ O_D, r_op ∈ O_R using ρ_D and ρ_R.
6:    s_c := r_op(d_op(s))
7:    if s_c is accepted then
8:       s := s_c
9:       if s_c has a better objective value than s* then
10:         s* := Local-Search(s_c)
11:         s := s*
12:      end if
13:      Update ρ_D and ρ_R
14:   end if
15: until maximum number of iterations is exceeded
16: return s*
```

Initial solution: by assigning all learners to self-study activities, the initial solution is generated. Qualified classrooms and teachers (note that every teacher is qualified to supervise a self-study activity) are randomly selected and up to N_c learners are added to the activities. The theoretical situation of insufficiently qualified classrooms is never observed in practice and therefore unlikely to happen.

Destroy operators: similar to Wouda et al. (2023), four different destroy operators are used. These operators remove a predetermined fixed amount of learners $d > 0$ from their current assignments.

1. *Random activity removal:* this operator randomly eliminates complete activities from the solution. The corresponding teacher, classroom and learners are marked unassigned. The operator continues removing activities until at least d learners are removed.

2. *Smallest activity removal:* the smallest activity is defined as the activity with the lowest number of learners assigned to it. This operator keeps removing the smallest activity until at least d learners are removed. With the use of this operator, a classroom and teacher with few learners are marked unassigned, who can subsequently be re-inserted (using a repair operator) in activities with more learners.

3. *Random learner removal:* this operator involves randomly removing a learner from an assignment, after which the learner is marked as unassigned. However, at least δ^- learners should remain in the activity. The operator continues until d learners are removed or until there is no possibility anymore to remove learners without violating constraints about the minimum activity size.

4. *Worst regret learner removal:* this operator removes learners with the worst regret, which is defined as the difference between their best and current assignment. In other words, it removes learners that are least met in their demand. The regret r_{lm} of learner $l \in L$ assigned to module $m \in M$, can be calculated as

$$r_{lm} = \max_{m' \in M} \{D_{lm'}\} - D_{lm} \tag{13}$$

After calculating the regret for every learner, the regrets are listed in decreasing order while keeping track of the corresponding learners. With the use of a skewed distribution (which favours larger regrets), d learners are selected. The distribution is constructed as a (decreasing) triangular for the first d values and subsequently uniformly flat. Then, the probability for a learner l with costs at index $j \in \{1, ..., |L|\}$ to be selected is:

$$Pr(\text{select } l) = \begin{cases} \frac{d-j+1}{\sum_{i=1}^{d} i+|L|-j} & \text{if } j \leq d, \\ \frac{1}{\sum_{i=1}^{d} i+|L|-j} & \text{otherwise.} \end{cases} \tag{14}$$

The operator removes all d learners, unless less than δ^- learners remain in their activities.

Repair operators: the two repair operators insert all unassigned learners back into the solution, ensuring that the resulting solution is feasible.

1. *Break-out activity:* this operator groups unassigned learners by modules they demand, whereafter the modules are listed in decreasing order by aggregating demand. An activity of such a module is scheduled for the grouped learners if there are at least δ^- learners, a qualified classroom and a qualified teacher. The algorithm prefers second-degree to teach second-degree modules. As a result, first-degree teachers can be preserved to teach first-degree modules. Besides, the operator chooses the smallest possible classroom for the new activity. If not all learners fit in the classroom, the largest classroom is chosen and part of the learners is not assigned to the new activity. When creating the new activity, the operator considers all learners in self-study activities and assigns learners to the new activity for which it is an improvement. If no new activities can be created, the remaining unassigned learners are assigned using the *greedy learner insert* operator.

2. *Greedy learner insert:* this operator assigns a randomly selected unassigned learner to his or her most demanded feasible instruction activity. The activity is feasible if there are currently less than $min\{\delta^+, N_c\}$ learners assigned to the activity and the learners demand for the activity is not zero. The learner is assigned to a self-study activity if no feasible instruction activity exist and a new self-study activity is created if necessary. If this is also not possible, a new self-study activity is created by converting the instruction activity with the smallest contribution to the objective into a self-study activity. The operator continues until all unassigned learners are assigned to an activity.

Acceptance criterion: the acceptance criterion is extracted from simulating annealing (SA), similar to Santini et al. (2018) who obtained good results. If the objective value of s^c is higher than s, the solution is accepted. Otherwise, the solution is accepted with the following

probability:

$$\Pr(\text{accept } s^c) = \exp\left\{\frac{f(s^c) - f(s)}{T}\right\}, \tag{15}$$

where $f(\cdot)$ represents the objective value of the solution. The parameter T is the temperature, which decreases in every iteration at a cooling rate $\gamma \in (0, 1)$. The starting value of T is determined at the begin of the heuristic.

Local search: if s is replaced by s^c, the algorithm applies local search to possibly further improve s^c. The operator used for this purpose, the *reinsert learner* operator, determines all improving relocation moves out of self-study and applies them in order of decreasing objective gain. An improving move to self-study is unlikely to exist due to the structure of the repair operators. If a relocation move has become infeasible due to a previous move, the move is skipped. Calling this operator is repeated until no further improving moves are found.

Updating and operator selection: The destroy and repair operator are determined in each iteration with the use of the roulette wheel mechanism (Røpke and Pisinger, 2006). The probability of selecting operator j, assuming there are $k \in \mathbb{N}$ operators each with weight $z_i \geq 0, i \in \{1, ...k\}$, is:

$$\Pr(\text{select } j) = \frac{z_j}{\sum_{i=1}^{k} z_i}.$$

The weights are tracked in two lists: ρ_D and ρ_R for the destroy and repair operators respectively. The initial value of every operators weight is set to 1.

After using an operator j, its performance is determined based on (i) it finds a new best solution, (ii) it improves the current solution, (iii) it does not improve the current solution but the solution is accepted. Each of these performance measures is given a score $\omega_i, i \in \{1, ...3\}$. The weight z_j of operator j is updated as the convex combination $\theta z_j + (1 - \theta)\omega_i$, where the predefined decay parameter $\theta \in [0.1]$ controls the influence of the operator effectiveness on its weight. Both the destroy and the repair operator are updated by the same factor as we cannot differentiate between theirs effect.

6 Max-Min Ant System metaheuristic

The Max-Min Ant System (MMAS) is a metaheuristic that is introduced in Stützle and Hoos (1996) but in general described in Stützle and Hoos (2000). In the MMAS each ant of a predefined number of artificial ants A, constructs a feasible candidate solution s^c. The activities in s^c are constructed by repeatedly selecting a classroom and assigning a module, teacher and learners to it. The assignment of the module and learners are based on pheromone levels, where a higher pheromone level indicates a higher chance of choosing the assignment. The distinguishing aspect of the MMAS compared to other ant colony optimisation heuristics is the use of only the best solution to spread pheromone rather than using all found solutions. The solution with the highest objective $f(\cdot)$ of all ants is saved as s^{it}. This process is repeated for a predefined number of iterations N and if $f(s^{it}) > f(s^{best})$, a new best solution is found and $s^{best} = s^{it}$. Every iteration, the pheromone trails are updated using either s^{it} or s^{best}, depending of the

progress of the algorithm. The pseudo-code for the MMAS is given in Algorithm 2. In the next sections, we elaborate further on the construction of a candidate solution and the initialization and updating procedure of the pheromone levels.

Algorithm 2 Max-Min Ant System

Input: An upper bound for the solution.
Output: Best observed solution s^{best}.
Initialize pheromone trails τ_{mc} and τ_{lm}.
Initialize $f(s^{best}) = 0$
while maximum number of iterations is not reached **do**
  $S \leftarrow \emptyset$
  for each ant **do**
    Construct a candidate solution s^c with τ_{mc} and τ_{lm}
    add s^c to S
  end for
  $s^{it} = argmax\{f(s)|s \in S\}$
  $s^{best} = argmax\{f(s)|s \in \{s^{it}, s^{best}\}\}$
  Update pheromone trails
end while

6.1 Solution construction

In every iteration, an ant constructs a feasible candidate solution s^c according to the following procedure. An ant starts with marking all classrooms, teachers and learners unassigned. Subsequently, instruction activities are added to the solution by repeatedly selecting a classroom, module, teacher and learners. In the module and learners selection, pheromone trails are included which encourage the repetition of good choices. We use separate pheromone trails for module and learners selection due to the possibly varying importance of their retention. The pseudo-code for this process is described in Algorithm 3 and the details of the different steps are as follows.

Algorithm 3 Solution construction

while creating a new instruction activity is possible **do**
  Choose a random instruction classroom $c \in C$
  Choose a module $m \in M \setminus \{m_S\}$
  if no module can be assigned to c **then**
    Return to step 2
  end if
  Choose randomly a qualified teacher $t \in T$
  Choose learners for the module
  Create an instruction activity with c, m, t and the learners and add to the solution
end while
If there are unassigned learners left, create self-study activities and assign those learners to them.
Perform local search on the solution

Choosing a classroom: the available classrooms are limited and therefore in most solutions almost all classrooms will be used. For this reason, we decide to randomly choose a classroom from all unassigned classrooms as first step. Considering that we want to add an instruction activity to the solution, only instruction classrooms can be selected.

Choosing a module: choosing a module depends on the selected classroom as not all classrooms are qualified for each module. Certain classroom module combinations can yield high objectives and thus, preserving these beneficial combinations can improve the algorithm. The preservation of beneficial pairs is accomplished by using pheromone trails.
Once a classroom is selected, all qualified modules (i.e. at least one unassigned teacher is available for the module and the classroom is qualified for the module) are determined. Besides, there must be at least δ^- unassigned learners that prefer the module over self-study. This criterion ensures that the remaining students, who will be assigned to self-study at the end, are likely to prefer it over all created instruction activities. The module-classroom combinations mc are stored in a set $\mathcal{M}$ and in case no such combinations exist, another classroom is chosen. Otherwise, similar to Stützle and Hoos (2000), the probability $p_{mc}(t)$ to choose a module classroom combination in iteration t is defined as:

$$p_{mc}(t) = \frac{[\tau_{mc}(t)]^\alpha \cdot [\eta_{mc}]^\beta}{\sum_{mc \in \mathcal{M}} [\tau_{mc}(t)]^\alpha \cdot [\eta_{mc}]^\beta} \tag{16}$$

where $\tau_{mc}(t)$ is the pheromone level of a combination at iteration t and η_{mc} is a locally available heuristic. In this case, the aggregated demand per module is employed as local heuristic because high demand contributes to a high objective value. α and β determine the relative importance of the pheromone level and the local heuristic, respectively.

Choosing a teacher: in the design of choosing a module, we already include the guarantee of an available teacher. Therefore, we only determine the unassigned and qualified teachers for this module and randomly select a teacher from this group.

Choosing learners: when the classroom, module and teacher are chosen, the learners can be selected. First, all unassigned learners that prefer this module over self-study are determined and the possible learner module combinations are stored in a set $\mathcal{L}$. In the selection of the module is already included that enough learners are available. Then, the probability of choosing a learner module combination is again determined with the definition of Stützle and Hoos (2000):

$$p_{lm}(t) = \frac{[\tau_{lm}(t)]^\zeta \cdot [\eta_{lm}]^\xi}{\sum_{lm \in \mathcal{L}} [\tau_{lm}(t)]^\zeta \cdot [\eta_{lm}]^\xi} \tag{17}$$

Now, the pheromone levels τ_{lm} of learner module combinations are used and η_{lm} is defined as the demand of learner l for module m since higher demand should increase the probability of choosing the learner. The number of learners that are selected is $min(\lambda \cdot N_c, \delta^-)$, where $\lambda \in (0, 1]$ is a fraction that determines how much of the classroom capacity is used. The activity is partly filled because this allows the local search later to add learners.

An ant creates an instruction activity with the chosen classroom, module, teacher and learners, adds it to the solution and marks the classroom, teacher and learners as assigned. The process is repeated until one of the following criteria is reached: (i) no available classroom

can be found that satisfies the requirements of a qualified module and teacher and a sufficient number of learners that prefer the module over self-study, (ii) the aggregated capacity of available classrooms where self-study is allowed exceeds the number of unassigned learners, thereby we ensure that there is always enough capacity left to assign the remaining learners to self-study, or (iii) the number of available teachers falls below the number of self-study classrooms that are needed to assign the remaining learners to self-study. After reaching one of the criteria, self-study activities are created for the remaining unassigned learners with the largest unassigned classrooms and random unassigned teachers.

The solution is improved with the *reinsert learner* local search, which is described in Section 5. This local search remains beneficial in this context again because the classrooms of the instruction activities are partly filled and learners from self-study could benefit from moving to an instruction activity. Besides, moving from an instruction activity to a self-study activity is again unlikely to be preferred since only learners are selected that prefer the module over self-study.

6.2 Pheromone updating

The pheromone levels are updated after iteration t in a similar manner to Stützle and Hoos (2000), although some changes need to be made since we maximise an objective instead of minimise. Because pheromone levels determine the chance of choosing certain solution components, a decent balance between exploration and exploitation is crucial. Exploration is defined as the ability to explore new solution regions, while exploitation is the ability to exploit good solutions and improve them. Insufficient exploration could result in convergence towards a local optimum. On the other hand, absence of exploitation could result in a sub optimal solution.

Updating the pheromone levels in the MMAS is performed by multiplying all levels with a constant factor ρ and using one solution per iteration to spread pheromone. The corresponding rule is:

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \Delta\tau_{ij}^{best} \tag{18}$$

Only the solution of one single ant is used to update the pheromone levels in every iteration. To ensure a good balance between first exploration and second exploitation, the first ϵ ($0 \leq \epsilon \leq N$) iterations $\Delta\tau_{ij}^{best}$ is defined as $\gamma \cdot f(s^{it})$, allowing the algorithm to reward locally optimal solutions. After ϵ iterations, $\Delta\tau_{ij}^{best} = \gamma \cdot f(s^{best})$ to guarantee exploitation. Using the objective value for updating, results in rewarding good solutions since more pheromone is spread for high objective values. The proportional ratio γ, likewise utilized by Lin et al. (2013), ensures the same order of magnitude for τ_{ij} and η_{ij} in Equation 16 and 17.

The constant $\rho \in [0, 1)$ is the trail persistence, which means $1 - \rho$ models the evaporation. If a combination is not used in several consecutive best solutions, the trail decreases exponentially, resulting in "forgetting" bad combinations.

The different pheromone trails for the module classroom combinations and the learner module combinations are updated with a separate evaporation rate. The underlying rationale is that for either module classroom combinations or learner module combinations it may be required to be more prone to being forgotten.

Another measure to avoid stagnation of the search and therefore ensure exploration, is limiting the pheromone levels to an interval $[\tau_{min}, \tau_{max}]$. This way, the influence of the pheromone levels relative to the locally heuristic information is limited and the chance of one combination outperforming all others is lowered. After each iteration, the pheromone levels are verified and set to τ_{min} or τ_{max} if the levels are too low or too high, respectively.

6.3 Pheromone initialization and bounds

The maximum pheromone level τ_{max} in iteration t is asymptotically bounded and calculated the following way, based on the definition of Stützle and Hoos (2000) but adapted for the maximising problem:

$$\tau_{max}(t) = \frac{1}{1-\rho} \cdot \gamma \cdot f(s^{opt}) \tag{19}$$

where s^{opt} is the same solution used for updating the trail, that is s^{it} first and s^{best} after ϵ iterations. As a result, τ_{max} is an estimate of the asymptotically maximum value and fluctuates during the algorithm. Therefore, τ_{max} is updated after every iteration. We aim to initialize all pheromone levels at τ_{max}, since this leads to a high level of exploration at the beginning of the algorithm. However, no solutions are computed at the start of the algorithm. Therefore, we initialize the pheromone levels in such a way that after the first iteration all pheromone levels are automatically set to $\tau_{max}(1)$, the pheromone level computed with the first iteration-best solution. This is accomplished by setting $f(s^{opt})$ to an upper bound of the solution when initializing the pheromone levels, leading to high values of $\tau(0)$. Considering that the pheromone levels are limited to values within the imposed bounds, all pheromone levels are set to $\tau_{max}(1)$ after the first iteration. The upper bound is calculated by summing the maximum demand of each learner. Given the constraint that a learner can only be assigned to one module, the upper bound represents the highest attainable objective in theory.

For determining the minimum value for the pheromone level, we need the probability p_{best} of constructing the optimal solution. For the sake of simplicity, we assume that p_{dec}, the probability of choosing the solution component that leads to the optimal solution, is constant during the process. Let n be the number of times an ant has to choose a component. Then, the formula to compute the p_{best} is:

$$p_{best} = \left(\frac{\tau_{max}}{\tau_{max} + (avg-1) \cdot \tau_{min}}\right)^n \tag{20}$$

with avg the average number of components an ant has to choose from. Hence, the formula to determine τ_{min} is:

$$\tau_{min} = \frac{\tau_{max} \cdot (1 - \sqrt[n]{p_{best}})}{(avg-1) \cdot \sqrt[n]{p_{best}}} \tag{21}$$

For the module classroom pheromone trail, avg is set to half of the total classrooms and n to the total number of classrooms. When setting the values for the learner module combinations, avg is set to half of the learners and n to the total number of learners. Since τ_{min} depends on τ_{max}, the value is also updated after every iteration.

7 Experiments

This section gives an overview of the experiments that are performed to compare three methods to solve the Hourly Learning Activity Planning Problem, namely solving the integer linear programming (ILP) problem to optimality, the Adaptive Large Neighborhood Search (ALNS) heuristic and the Max-Min Ant System (MMAS) heuristic. In Section 7.1, we explain the experimental design and in Section 7.2 we show the results of the parameter tuning. Comparison of the two heuristics in terms of performance and a parameter sensitivity analysis are shown in Section 7.3.

7.1 Experimental design

No standardised benchmarks for personalised learning in the Dutch setting were available, which is the data needed for this research. Hence, benchmark cases are formulated based on expert estimates. In order to investigate the influence of certain parameters and policy decisions, a set of experiments each consisting of 100 instances are constructed. Each experiment contains classrooms, course modules, teachers and learners to represent a six-year secondary education program. Within each experiment we differ between several parameters to investigate their influence.

The different school sizes we consider are medium (M), large (L) and extra large (XL), all having their own characteristics. The M, L and XL size schools contain 800, 1200 and 1600 learners respectively. Besides, in the schools are 80, 120 and 160 teachers, 40, 60 and 80 instruction classrooms and 3, 4 and 6 large self-study classrooms included in M, L and XL schools respectively. The capacity of the classrooms are 32 learners for an instruction classroom and 80 learners for a self-study classroom. These capacities are the same for all school sizes.

A school teaches 12 representative courses, based on expert estimates, each consisting of 48 modules. Therefore, there are 576 instruction modules in total. The required classroom qualifications and the average number of hours learners typically devote, are determined per course. The division of classrooms and teachers over courses is based on the fraction of hours spent on the course by students. For example, for a course that is scheduled 3 hours in a 30 hours schedule, 10% of the teachers is assigned to this course and 10% of the instruction classrooms are assigned the course's room type. We assume that self-study activities can be scheduled in the "regular" classrooms and in the large self-study classrooms.

The minimum number of learners needed for an activity to be scheduled δ^- is set to 5 and the maximum δ^+ to 30 for all experiments. These numbers have been established based on expert estimates and are expected to remain unchanged under the personalised learning criterion.

The qualification distribution of teachers is expressed as $(p; q; 1-p-q)$ for the fraction of first-degree, second-degree and third-degree teachers. Their specifications are given in Section 3.1. The distributions we investigate are $(1; 0; 0)$, representing a school with only first-degree teachers, $(0.5; 0, 5; 0)$, which is commonly used in Dutch secondary schools, and $(0.4; 0.4; 0.2)$, which could be a economically interesting due to the lower costs for third-degree teachers.

The learner demand for each course is randomly generated from a normal distribution with $N(\mu_l, \sigma)$. Nominal progression, which represents the typical pace of learning that a learner is

expected, is used as a basis to compute μ_l. The nominal learning path is 8 modules per year $y_l \in \{0, 1, ..., 5\}$ and therefore $\mu_l = 8y_l + 4$. The σ parameter represents the demand spread between learners. In the experiment, we let the parameter vary between 0, 1, 2 and 3. Higher values of σ means that learner demands are more widely distributed across the modules. From the formula, it can be derived that 95% of learners from the same group work on course modules that are 5 (for $\sigma = 1$), 9 (for $\sigma = 2$), or 13 (for $\sigma = 3$) modules from each other. When $\sigma = 0$, all students from the same group work on the same module. The generated learner demand value is rounded to the nearest integer in $\{1, 2, ..., 48\}$.

The demand value that is generated for the module is drawn from a Exp($\beta = 2$) distribution. For the self-study fulfilment level w, which is the fraction that satisfies a learners demand if the activity is a self-study session, we investigate 0.50 and 0.75.

We use a full factorial design to investigate the effects of the parameters, which results in 144 experiments of 100 instances that are generated. An overview of the parameter setting per experiment is given in Table 2 in Appendix B.

7.2 Tuning

In both the ALNS and MMAS heuristic, parameters need to be tuned.

For the parameters in the ALNS, we rely on the parameter configurations as determined by Wouda et al. (2023) since they obtained good results. The parameters were tuned with the optimisation package SMAC3 (Lindauer, 2022), which uses Bayesian optimisation to tune hyperparameters. 144 tuning instances were given twenty independent runs, which resulted in the following parameter values: $\omega = (21.8, 13.6, 3.8)$, $\theta = 0.8$ and $d = 15\%$. The initial temperature for the acceptance criterion is set such that a solution with an objective up to 5% worse than the initial solution is accepted with a probability of 50%. The number of iteration is set to 25,000.

The parameters for the MMAS are also tuned with SMAC3 (Lindauer, 2022) during a configuration run of four hours. After attempting various methods, the tuned parameter values of only the first tuning instance led to the best results and are used for the experiment runs.

For the process of tuning parameters, it is necessary to set tuning intervals. According to Gentile (2015), setting the number of ants A to 25 is commonly accepted for the MMAS. Nevertheless, a higher value of A logically leads to higher objective values but also to higher running-times, which could result in a disproportionate comparison of two heuristics. Hence the tuning interval of A is set to [0, 20]. The tuned value of A is 19 ants. Gentile (2015) also mentions $\alpha = 1$ and $\beta = 2$ as a rule of thumb in the MMAS. Since ζ and α are equivalent in meaning, both intervals are set to [0.5, 3]. Using the same logic, the intervals of ξ and β are [0.5, 3]. The tuned values of $\alpha = 1.801$ and $\beta = 2.813$ can be explained with the order of magnitude of the pheromone level and aggregated demand. Since the first is higher than the second, α and β counterbalance this. The same reasoning holds when choosing learners, concluding from $\zeta = 1.678$ and $\xi = 2.824$. An interesting observation is that the corresponding parameters for module classroom combinations and learner module combinations are nearly equal. Therefore, a worthwhile possibility for future research is to investigate the option of using solely 2 parameters, that is $\alpha = \zeta$ and $\beta = \xi$. The tuned trail persistence rates $\rho_{mc} = 0.841$ and $\rho_{lm} = 0.624$ indicate

that learner module combinations are more easily forgotten, which is a surprising outcome since learners directly influence the objective and modules only indirectly. The interval of p_{best} is set to [5e-10, 1.0], with a tuned value of 0.044. The lower bound of λ, the fraction of classroom capacity that is initially filled, is set to 0.5. A lower value would choose less than δ^- learners for an activity and no instruction activities would be created. The tuned value is 0.811. The interval of the proportional ratio γ is set to [0.000001, 0.1] since the goal of the ratio is to decrease the magnitude of the objective value in comparison with the locally used heuristic (explained in Section 6.2). The obtained value for γ is 0.042. The interval for the number of iterations N is set to [0,40] because again too many iterations could cause a disproportionate comparison between the heuristics in terms of running time. After tuning, $N = 37$ is determined, while using s^{best} after 35 iterations, that is $\epsilon = 35$. An overview of the tuned parameters, their interpretation, tuning intervals and the configuration values are shown in Table 1.

Table 1: Tuned parameters used for the Max-Min Ant System, including the considered interval and the configuration value

Parameter	Interpretation	Interval	Value
N	number of iterations	[0, 40]	37
A	number of ants	[0, 20]	19
ϵ	number of iterations before s^{best} is used for updating the pheromone	[0, 40]	35
α	relative importance of pheromone level τ_{mc}	$[0.5, 3]$	1.801
β	relative importance of aggregated demand	$[0.5, 3]$	2.813
ζ	relative importance of pheromone level τ_{lm}	$[0.5, 3]$	1.678
ξ	relative importance of individual demand	$[0.5, 3]$	2.824
ρ_{mc}	trail persistence of τ_{mc}	$[0.02, 0.99]$	0.841
ρ_{lm}	trail persistence of τ_{lm}	$[0.02, 0.99]$	0.624
p_{best}	probability of choosing the the solution component that leads to the optimal solution	[5e-10, 0.1]	0.044
λ	fraction of the classroom capacity that is initially filled	$[0.5, 1.0]$	0.811
γ	proportional ratio for the objective value	$[0.000001, 0.1]$	0.042

7.3 Comparison experimental results

The ILP model and both heuristics are implemented in Python 3.9. The first three instances of each experiment are used to establish information about the different methods, parameters and policy choices. The information about the ILP performance is derived from the runs done by Wouda et al. (2023) (for experiment 15, 21, 39 and 93 it was necessary to use a fourth or fifth instance as there was no ILP information made available by Wouda et al. (2023) about the desired instances). A complete overview of the performance results of the three methods is given in Table 3 in Appendix C. In Section 7.3.1, we discuss and compare the general performances of the MMAS. Subsequently, in Section 7.3.2 we analyse the performance of the MMAS compared with ALNS per parameter and policy decision.

7.3.1 Performance comparison

In this research, the relative gap with the optimal objective is used as a measure to compare the heuristic performances. Considering that heuristics try to find the highest possible objective, this seems to be a good measure for comparison. The average computation time of the MMAS exceeds ALNS, with 145 seconds against 96 seconds. Still, its average running time is far below the running time of ILP, which is 2272 seconds, and sufficient for calculating a school schedule. In Figure 1, the three solving methods are compared based on optimality gaps, which are calculated as the relative difference of the optimal objective with the objective of the solving method. For the ILP method, the gap with optimality after 600 seconds is implemented and for the heuristics the objective after the stopping criterion is satisfied. We observe that the ILP method provides small optimality gaps, particularly in experiments with low demand spread. This is likely caused by the extended computation time of the method. Comparing the two heuristics leads to the conclusion that MMAS outperforms ALNS for all demand spreads. On average, the objective of the MMAS deviates 1.66% from the optimal objective value, while ALNS deviates 2.22% on average. Besides, the length of the boxplot boxes indicate that the MMAS has a lower variability than ALNS for all types of demand spread, meaning that the MMAS performs more predictable. In 91 of the 144 experiments, the MMAS outperformed ALNS by obtaining a higher objective value. These results indicate that MMAS is an effective heuristic for solving the HLAPP. Considering the longer running times of the MMAS compared to ALNS, an interesting improvement of this research would involve reducing running times of MMAS (by decreasing the number of iterations and ants) and reevaluating the performance of the heuristics again.

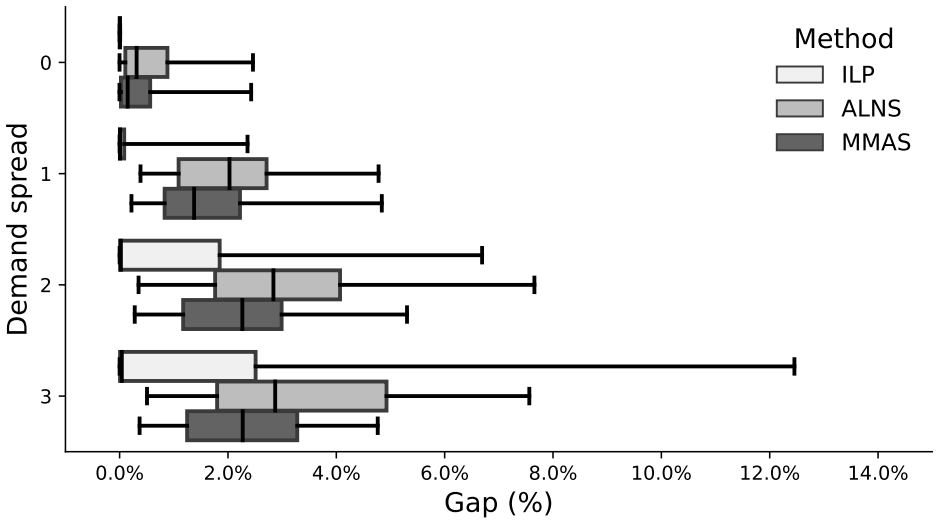


Figure 1: Boxplots to compare the optimality gaps between the ILP, ALNS and MMAS solution methods. For the ILP, the optimality gap after 600 seconds is used

7.3.2 Parameter sensitivity analysis

In this section, we consider the influence of the included parameters on the MMAS and its relative performance compared to ALNS when varying certain parameters. In Figure 2a we

observe that already a modest learner demand spread leads to high optimality gaps for MMAS. The explanation is that lower demand spreads indicate stronger preferences among learners for the same modules, which results in creating activities easier. For $\sigma = 0$ and $\sigma = 1$, ALNS and the MMAS exhibit similar performance, in both cases the optimality gap difference is less than 0.5 percentage point. Analysing $\sigma = 2$ and $\sigma = 3$ leads to the conclusion that the MMAS outperforms ALNS with average optimality gap differences of 0.76 and 1.05 percentage point respectively. In consequence, the MMAS is preferred over ALNS especially for high levels of learner demand spread.

After investigating the influence of the self-study fulfillment parameter w in Figure 2b, we notice that the performances of the MMAS are more influenced by a high percentage of w than those of ALNS. The optimality gap of MMAS changes from 2.34% to 0.97%, while the gaps of ALNS changes from 2.67% to 1.77%. The underlying reason is that ALNS pays less attention to self-study than the MMAS. For example, in only one destroy operator (i.e. the *worst regret learner removal*), the algorithm considers learners' preferences regarding self-study. By comparison, the MMAS devotes considerable focus on the influence of self-study activities on the objective. Since the algorithm only assigns learners to instruction activities that prefer those over self-study, the remaining unassigned learners prefer self-study over all created activities. Considering that a high penalty value increases the impact of self-study activities on the objective, the MMAS tends to favour a high penalty.

Figure 2: Objective value gaps by learner demand spread and self-study penalty, comparing ALNS and MMAS

(a) Learner demand spread

(b) Self-study penalty

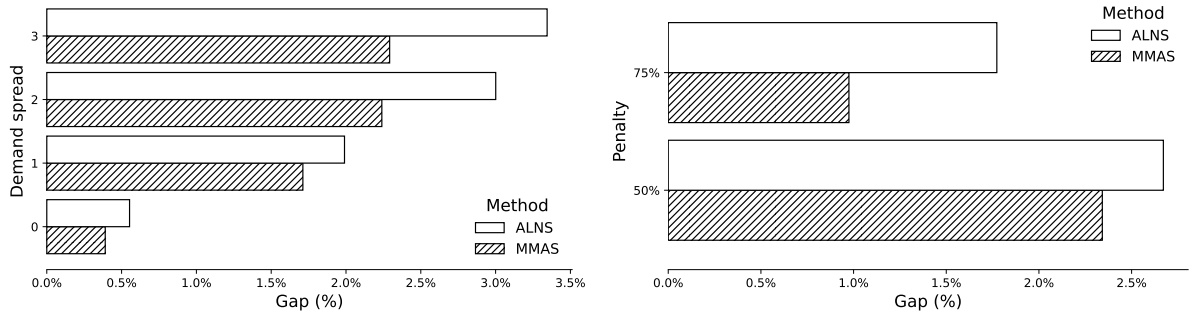
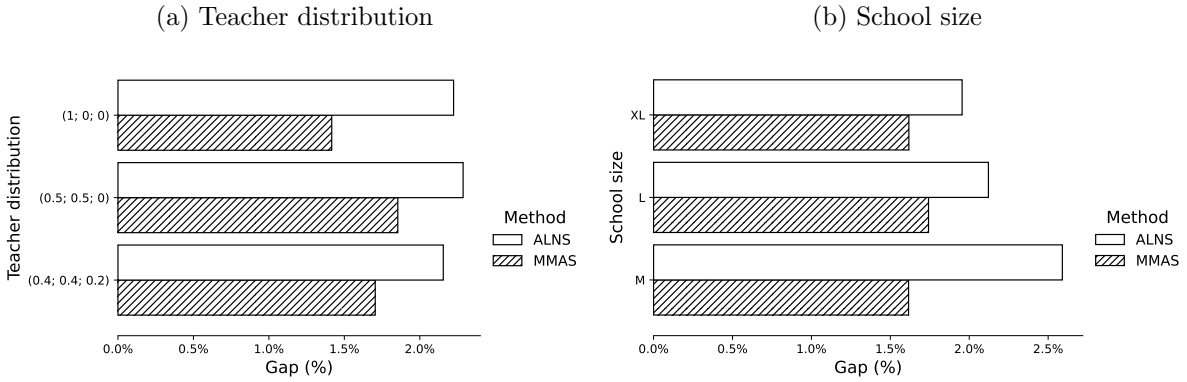


Figure 3 compares the two heuristics based on school specific parameters, namely teacher distribution and school size. The average optimality gaps of the MMAS show less than 0.5 percentage point difference with varying teacher distributions in Figure 3a. We conclude that the method performs consistently for all types of teacher distributions. Comparing the method to ALNS indicates that the MMAS is preferred since their optimality gaps differences are higher than 0.5 percentage points for all types of distributions.

In Figure 3b, we differentiate between the school sizes. The performance of MMAS is not influenced by the school size since 0.12 percentage point is the highest optimality gap difference, that is between XL and L schools. When comparing the MMAS with ALNS, we conclude the optimality gaps of ALNS are the highest for medium size schools, which is in line with the

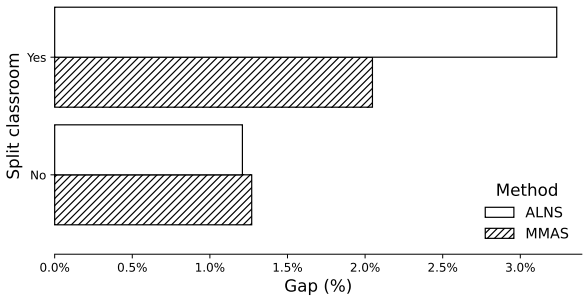
findings of Wouda et al. (2023). The MMAS appears to be a good alternative in this case since its optimality gaps are lower (1.62% for MMAS against 2.59% for ALNS), indicating better solutions. For large and extra large school sizes, MMAS also performs better, although the differences are not considerably large with differences of 0.38 and 0.33 percentage point for L and XL school respectively.

Figure 3: Objective value gaps by teacher distribution and school size, comparing ALNS and MMAS



The last policy we investigate is splitting all the classrooms of a school. Then, the capacity of all classrooms is reduced to 16 learners instead of 32 but this also leads to a duplication of the number of available classrooms. From Figure 4 we conclude that the splitting policy has obviously more effect on the performance of the ALNS compared to MMAS, since its average optimality gap increases from 1.21% to 3.23%. The underlying factor behind this is that ALNS is less concerned about learner demands. For instance, the *random activity removal* and the *smallest activity removal* remove learners without considering their personal demand. With an increased number of classrooms, the chance to choose the "good" classroom to remove is smaller. In conclusion, the MMAS is preferred as heuristic for schools that have implemented classroom splitting, while no evidently superior method emerges for schools who have not.

Figure 4: Objective value gaps by splitting the classrooms, comparing ALNS and MMAS



8 Conclusion

In this research, the Max-Min Ant System (MMAS) metaheuristic is proposed to solve the Hourly Learning Activity Planning Problem (HLAPP). The problem covers constructing a timetable by assigning learners to classrooms, modules and teachers while satisfying as much as demand possible. The MMAS is an innovative heuristic in which several solutions are generated by ants that uses pheromone trails to encourage the repetition of good assignments. To analyse its performance, two methods are introduced. First, an integer linear programming (ILP) model, which is evaluated when solving the model to optimality and for a running time of 600 seconds. Second, the results of the Adaptive Large Neighborhood Search (ALNS) metaheuristic are used as benchmark to determine the relative performance of the MMAS. This algorithm repeats destroying and repairing a solution and thereby advances to a high-quality solution. The methods are compared using a wide-ranging set of experiments, based on expert insight.

We conclude that the computation times of the ILP method are too long to solve the problem and therefore the use of a heuristic is preferred. We observe that the MMAS shows superior performance results since its average optimality gap is 1.7% against 2.2% for the ALNS. The average running times of ALNS are slightly lower but both running times are perfectly suitable for generating a school schedule quickly. The investigation of varying parameters shows that adapting learner demand spread, changing the fulfilment of self-study or splitting the classrooms in a school, results in significantly different performance results for the MMAS. Other examined parameters did not influence the optimality gap more than 0.5 percentage point. In the comparison between the two heuristics, MMAS surpasses ALNS particularly for high levels of learners demand spread and if the policy of splitting classrooms is maintained. Additionally, for high fulfilment of self-study and small school sizes, the MMAS also showed relative better results, although to a lesser extent. Eventually, we conclude that MMAS emerges as a more effective heuristic compared to ALNS. Therefore, the MMAS proves itself as an effective heuristic for solving the HLAPP.

For further research, several options can be considered. In this study, two separate pheromone trails with corresponding parameters are used in the MMAS to model the assignment probabilities when selecting modules and learners. However, it is worthwhile to investigate if one single trail with all possible combinations can improve the heuristic since this would make the heuristic less dependent on the initially chosen classroom and module. Besides, extending the heuristic with a tabu search could improve the algorithm. There are a number of inferior solutions for the problem due the enormous number of combinations possible. Eliminating those solutions could enhance the global optimisation.

References

Bell, J. E. and McMullen, P. R. (2004). Ant colony optimization techniques for the vehicle routing problem. *Advanced Engineering Informatics*, 18(1):41–48.

Birbas, T., Daskalaki, S., and Housos, E. (1997). Timetabling for greek high schools. *Journal of the Operational Research Society*, 48(12):1191–1200.

Crawford, B., Soto, R., Johnson, F., Monfroy, E., and Paredes, F. (2014). A max–min ant system algorithm to solve the software project scheduling problem. *Expert Systems with Applications*, 41(15):6634–6645.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41.

European Commission and European Political Strategy Centre (2019). *10 trends transforming education as we know it*. Publications Office.

Gentile, M. (2015). A theoretical consideration of theparameters of the max min ant system.

Jangra, R. and Kait, R. (2017). Analysis and comparison among ant system; ant colony system and max-min ant system with different parameters setting. *2017 3rd International Conference on Computational Intelligence amp;amp; Communication Technology (CICT)*.

Kalyuga, S., Ayres, P., Chandler, P., and Sweller, J. (2003). The expertise reversal effect. *Educational Psychologist*, 38(1):23–31.

Kannan, A., van den Berg, G., and Kuo, A. (2012). Ischedule to personalize learning. *Interfaces*, 42(5):437–448.

Kristiansen, S., Sørensen, M., and Stidsen, T. R. (2011). Elective course planning. *European Journal of Operational Research*, 215(3):713–720.

Lin, M.-H., Tsai, J.-F., and Lee, L.-Y. (2013). Ant colony optimization for social utility maximization in a multiuser communication system. *Mathematical Problems in Engineering*, 2013:1–8.

Lindauer, Marius, E. (2022). Smac3: A versatile bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23(54):1–9.

Pillay, N. (2013). A survey of school timetabling research. *Annals of Operations Research*, 218(1):261–293.

Pisinger, D. and Røpke, S. (2010). Large neighborhood search. *International Series in Operations Research amp; Management Science*, page 399–419.

Røpke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472.

Santiago-Mozos, R., Salcedo-Sanz, S., DePrado-Cumplido, M., and Bousoño-Calzón, C. (2005). A two-phase heuristic evolutionary algorithm for personalizing course timetables: A case study in a spanish university. *Computers amp; Operations Research*, 32(7):1761–1776.

Santini, A., Ropke, S., and Hvattum, L. M. (2018). A comparison of acceptance criteria for the adaptive large neighbourhood search metaheuristic. *Journal of Heuristics*, 24(5):783–815.

Saviniec, L. and Constantino, A. A. (2017). Effective local search algorithms for high school timetabling problems. *Applied Soft Computing*, 60:363–373.

Shemshack, A. and Spector, J. M. (2020). A systematic literature review of personalized learning

terms. *Smart Learning Environments*, 7(1).

Stützle, T. and Hoos, H. H. (1996). Improving the ant system: A detailed report on the max–min ant system. *FG Intellektik, FB Informatik, TU Darmstadt, Germany, Tech. Rep. AIDA–96–12*.

Stützle, T. and Hoos, H. H. (2000). Max-min ant system. *Future Gener. Comput. Syst.*, 16:889–914.

Wouda, N. A., Aslan, A., and Vis, I. F. (2023). An adaptive large neighbourhood search metaheuristic for hourly learning activity planning in personalised learning. *Computers amp;amp; Operations Research*, 151:106089.

Appendices

A Code description

The code used in this research contains the Adaptive Large Neighborhood Search, the Max-Min Ant System algorithm and files to analyse the algorithms. The used directory of the code, now programmed as ..., should be adapted to the users own directory. Besides, the repository assumes an '/experiments' directory is set-up, and populated with the experimental data, similar to the data of Wouda et al. (2023) (the data can be obtained by downloading the *first revision.zip* file at this website).

A.1 Max-Min Ant System

All files corresponding to the Max-Min Ant System can be found in the *extension* directory.

The general algorithm code for the Max-Min Ant System is in the *antColonyOptimization.py* file. It includes a loop for several experiments and instances and makes use of the tuned values of the variables. The methods to select teacher, modules, classrooms and learners are also programmed in this file. In several iterations, ants generate solutions and their objective values are compared. The code keeps track of the best solution and running times. The files are saved in the map *outputExtension*.

The *classesExtension* directory contains the used dataclasses of the algorithm. Most classes are similar to the code of Wouda et al. (2023), sometimes with little adaptions. In the *Problem* class, all possible learner module combinations and module classroom combinations can be calculated. Besides, a method to compute the upper bound is added. A new created class is *Pheromone*, which keep track of the possible combinations for learners and modules and for modules and classrooms. It saves their pheromone levels in an efficient way. The *MaxPheromone* and *MinPheromone* classes keep track of the maximum and minimum amount of pheromone for the combinations respectively and update the levels. The *Statistics* class is created to keep track of the objective values and run-times of the algorithm. Finally, the *TeacherDictionary* keeps track of teachers and their qualified modules efficiently.

The variables are tuned in the file *tuning_extension.py*. In this code, the proposed intervals of the variables, as described in Table 1, are included. The program runs for four hours and prints the tuned values thereafter.

In *analyse_extension.py*, the results of the algorithm are analysed. This means, the instances are aggregated per experiment. The results are from the *outputExtension* directory and the analyses are saved in the directory *cache_directory*.

The *table_generating.py* and *figure_generating.py* files generate the used tables and figures from this research, respectively. Both use the analysed data from the *cache_directory* directory. The generated tables are printed and the figures are saved in the *figures* directory.

A.2 Adaptive Large Neighborhood Search

The code for the Adaptive Large Neighborhood Search (ALNS) is similar to the code of Wouda et al. (2023), although some changes were made. The code can be found in the *src* directory. In

the following files, modifications were implemented.

In the file *heuristics*, the algorithm for ALNS is located. The parser for the command-line is removed and exchanged for a *main* function. It includes a loop for the experiments and the instances, allowing to do runs consecutively. Inside the loop, the data location and result location are defined and the problem is set. The output of the results of the ALNS is in saved in the *experiments* directory and named 'heuristic'. Another modification is that the algorithm initializes the stop criterion for every loop iteration.

In the *analyse* file is also a loop included. This file is used to analyse the files of the ALNS and the ILP per experiment. The files are saved in the *cache* directory, which must be set-up by the user beforehand.

The *Problem* class is also changed. Since the problem must be set for every instance and experiment again, the cache of the class must be initialized multiple times again. This is done by initializing cache for the methods every time a new problem is set. This way, the class remains using cached properties which provides computational advantage.

B Experiments

Table 2: Parameter levels per experiment

Experiment	w	σ	Size	Split?	Qualifications
1	50%	0	M	Yes	(1, 0, 0)
2	50%	0	L	Yes	(1, 0, 0)
3	50%	0	XL	Yes	(1, 0, 0)
4	75%	0	M	Yes	(1, 0, 0)
5	75%	0	L	Yes	(1, 0, 0)
6	75%	0	XL	Yes	(1, 0, 0)
7	50%	1	M	Yes	(1, 0, 0)
8	50%	1	L	Yes	(1, 0, 0)
9	50%	1	XL	Yes	(1, 0, 0)
10	75%	1	M	Yes	(1, 0, 0)
11	75%	1	L	Yes	(1, 0, 0)
12	75%	1	XL	Yes	(1, 0, 0)
13	50%	2	M	Yes	(1, 0, 0)
14	50%	2	L	Yes	(1, 0, 0)
15	50%	2	XL	Yes	(1, 0, 0)
16	75%	2	M	Yes	(1, 0, 0)
17	75%	2	L	Yes	(1, 0, 0)
18	75%	2	XL	Yes	(1, 0, 0)
19	50%	3	M	Yes	(1, 0, 0)
20	50%	3	L	Yes	(1, 0, 0)
21	50%	3	XL	Yes	(1, 0, 0)
22	75%	3	M	Yes	(1, 0, 0)
23	75%	3	L	Yes	(1, 0, 0)
24	75%	3	XL	Yes	(1, 0, 0)
25	50%	0	M	Yes	(0.5, 0.5, 0)
26	50%	0	L	Yes	(0.5, 0.5, 0)
27	50%	0	XL	Yes	(0.5, 0.5, 0)
28	75%	0	M	Yes	(0.5, 0.5, 0)
29	75%	0	L	Yes	(0.5, 0.5, 0)
30	75%	0	XL	Yes	(0.5, 0.5, 0)
31	50%	1	M	Yes	(0.5, 0.5, 0)
32	50%	1	L	Yes	(0.5, 0.5, 0)
33	50%	1	XL	Yes	(0.5, 0.5, 0)
34	75%	1	M	Yes	(0.5, 0.5, 0)
35	75%	1	L	Yes	(0.5, 0.5, 0)
36	75%	1	XL	Yes	(0.5, 0.5, 0)
37	50%	2	M	Yes	(0.5, 0.5, 0)
38	50%	2	L	Yes	(0.5, 0.5, 0)
39	50%	2	XL	Yes	(0.5, 0.5, 0)
40	75%	2	M	Yes	(0.5, 0.5, 0)
41	75%	2	L	Yes	(0.5, 0.5, 0)
42	75%	2	XL	Yes	(0.5, 0.5, 0)
43	50%	3	M	Yes	(0.5, 0.5, 0)
44	50%	3	L	Yes	(0.5, 0.5, 0)
45	50%	3	XL	Yes	(0.5, 0.5, 0)
46	75%	3	M	Yes	(0.5, 0.5, 0)
47	75%	3	L	Yes	(0.5, 0.5, 0)
48	75%	3	XL	Yes	(0.5, 0.5, 0)
49	50%	0	M	Yes	(0.4, 0.4, 0.2)
50	50%	0	L	Yes	(0.4, 0.4, 0.2)
51	50%	0	XL	Yes	(0.4, 0.4, 0.2)
52	75%	0	M	Yes	(0.4, 0.4, 0.2)

Experiment	w	σ	Size	Split?	Qualifications
53	75%	0	L	Yes	(0.4, 0.4, 0.2)
54	75%	0	XL	Yes	(0.4, 0.4, 0.2)
55	50%	1	M	Yes	(0.4, 0.4, 0.2)
56	50%	1	L	Yes	(0.4, 0.4, 0.2)
57	50%	1	XL	Yes	(0.4, 0.4, 0.2)
58	75%	1	M	Yes	(0.4, 0.4, 0.2)
59	75%	1	L	Yes	(0.4, 0.4, 0.2)
60	75%	1	XL	Yes	(0.4, 0.4, 0.2)
61	50%	2	M	Yes	(0.4, 0.4, 0.2)
62	50%	2	L	Yes	(0.4, 0.4, 0.2)
63	50%	2	XL	Yes	(0.4, 0.4, 0.2)
64	75%	2	M	Yes	(0.4, 0.4, 0.2)
65	75%	2	L	Yes	(0.4, 0.4, 0.2)
66	75%	2	XL	Yes	(0.4, 0.4, 0.2)
67	50%	3	M	Yes	(0.4, 0.4, 0.2)
68	50%	3	L	Yes	(0.4, 0.4, 0.2)
69	50%	3	XL	Yes	(0.4, 0.4, 0.2)
70	75%	3	M	Yes	(0.4, 0.4, 0.2)
71	75%	3	L	Yes	(0.4, 0.4, 0.2)
72	75%	3	XL	Yes	(0.4, 0.4, 0.2)
73	50%	0	M	No	(1, 0, 0)
74	50%	0	L	No	(1, 0, 0)
75	50%	0	XL	No	(1, 0, 0)
76	75%	0	M	No	(1, 0, 0)
77	75%	0	L	No	(1, 0, 0)
78	75%	0	XL	No	(1, 0, 0)
79	50%	1	M	No	(1, 0, 0)
80	50%	1	L	No	(1, 0, 0)
81	50%	1	XL	No	(1, 0, 0)
82	75%	1	M	No	(1, 0, 0)
83	75%	1	L	No	(1, 0, 0)
84	75%	1	XL	No	(1, 0, 0)
85	50%	2	M	No	(1, 0, 0)
86	50%	2	L	No	(1, 0, 0)
87	50%	2	XL	No	(1, 0, 0)
88	75%	2	M	No	(1, 0, 0)
89	75%	2	L	No	(1, 0, 0)
90	75%	2	XL	No	(1, 0, 0)
91	50%	3	M	No	(1, 0, 0)
92	50%	3	L	No	(1, 0, 0)
93	50%	3	XL	No	(1, 0, 0)
94	75%	3	M	No	(1, 0, 0)
95	75%	3	L	No	(1, 0, 0)
96	75%	3	XL	No	(1, 0, 0)
97	50%	0	M	No	(0.5, 0.5, 0)
98	50%	0	L	No	(0.5, 0.5, 0)
99	50%	0	XL	No	(0.5, 0.5, 0)
100	75%	0	M	No	(0.5, 0.5, 0)
101	75%	0	L	No	(0.5, 0.5, 0)
102	75%	0	XL	No	(0.5, 0.5, 0)
103	50%	1	M	No	(0.5, 0.5, 0)
104	50%	1	L	No	(0.5, 0.5, 0)
105	50%	1	XL	No	(0.5, 0.5, 0)
106	75%	1	M	No	(0.5, 0.5, 0)
107	75%	1	L	No	(0.5, 0.5, 0)
108	75%	1	XL	No	(0.5, 0.5, 0)
109	50%	2	M	No	(0.5, 0.5, 0)

Experiment	w	σ	Size	Split?	Qualifications
110	50%	2	L	No	(0.5, 0.5, 0)
111	50%	2	XL	No	(0.5, 0.5, 0)
112	75%	2	M	No	(0.5, 0.5, 0)
113	75%	2	L	No	(0.5, 0.5, 0)
114	75%	2	XL	No	(0.5, 0.5, 0)
115	50%	3	M	No	(0.5, 0.5, 0)
116	50%	3	L	No	(0.5, 0.5, 0)
117	50%	3	XL	No	(0.5, 0.5, 0)
118	75%	3	M	No	(0.5, 0.5, 0)
119	75%	3	L	No	(0.5, 0.5, 0)
120	75%	3	XL	No	(0.5, 0.5, 0)
121	50%	0	M	No	(0.4, 0.4, 0.2)
122	50%	0	L	No	(0.4, 0.4, 0.2)
123	50%	0	XL	No	(0.4, 0.4, 0.2)
124	75%	0	M	No	(0.4, 0.4, 0.2)
125	75%	0	L	No	(0.4, 0.4, 0.2)
126	75%	0	XL	No	(0.4, 0.4, 0.2)
127	50%	1	M	No	(0.4, 0.4, 0.2)
128	50%	1	L	No	(0.4, 0.4, 0.2)
129	50%	1	XL	No	(0.4, 0.4, 0.2)
130	75%	1	M	No	(0.4, 0.4, 0.2)
131	75%	1	L	No	(0.4, 0.4, 0.2)
132	75%	1	XL	No	(0.4, 0.4, 0.2)
133	50%	2	M	No	(0.4, 0.4, 0.2)
134	50%	2	L	No	(0.4, 0.4, 0.2)
135	50%	2	XL	No	(0.4, 0.4, 0.2)
136	75%	2	M	No	(0.4, 0.4, 0.2)
137	75%	2	L	No	(0.4, 0.4, 0.2)
138	75%	2	XL	No	(0.4, 0.4, 0.2)
139	50%	3	M	No	(0.4, 0.4, 0.2)
140	50%	3	L	No	(0.4, 0.4, 0.2)
141	50%	3	XL	No	(0.4, 0.4, 0.2)
142	75%	3	M	No	(0.4, 0.4, 0.2)
143	75%	3	L	No	(0.4, 0.4, 0.2)
144	75%	3	XL	No	(0.4, 0.4, 0.2)

C Heuristics performance

Table 3: Comparison of the ILP performance with the ALNS heuristic and the MMAS heuristic. The *Obj.* columns show the average objective values of the three methods, determined with three instances of each experiment. The *Run* columns represent the running times of the ILP until optimality and for the ALNS and MMAS until the stopping criterion is satisfied. For ILP, the *Avg.* and *Max.* columns show the average and maximum gap after 600 seconds respectively. For the two heuristics, the *Avg.*, *Max.* and *Min.* columns show the relative objective gap, when comparing the objectives to the optimal objective.

	ILP				ALNS					MMAS				
Exp	Obj.	Run	Avg.	Max.	Obj.	Run	Avg.	Max.	Min.	Obj.	Run	Avg.	Max.	Min.
1	4749.10	28	0.00%	0.01%	4731.49	62	0.37%	0.21%	0.62%	4730.27	97.16	0.40%	0.21%	0.52%
2	7341.47	56	0.01%	0.01%	7248.95	90	1.28%	1.19%	1.32%	7322.62	170.80	0.26%	0.21%	0.32%
3	9885.60	169	0.00%	0.00%	9743.91	122	1.46%	1.12%	1.68%	9852.14	275.81	0.34%	0.25%	0.39%

	ILP				ALNS					MMAS				
Exp	Obj.	Run	Avg.	Max.	Obj.	Run	Avg.	Max.	Min.	Obj.	Run	Avg.	Max.	Min.
4	4896.61	25	0.00%	0.01%	4889.27	59	0.15%	0.11%	0.19%	4894.99	82.85	0.03%	0.01%	0.06%
5	7335.74	47	0.00%	0.00%	7298.31	88	0.51%	0.51%	0.52%	7330.06	152.96	0.08%	0.06%	0.11%
6	9931.40	83	0.01%	0.01%	9855.50	125	0.77%	0.60%	0.97%	9926.06	237.85	0.05%	0.04%	0.07%
7	4009.26	783	0.13%	0.37%	3873.64	69	3.50%	3.23%	4.02%	3924.19	100.64	2.17%	1.78%	2.70%
8	6572.08	2369	0.66%	1.19%	6379.13	102	3.02%	2.78%	3.25%	6389.56	179.45	2.86%	2.73%	2.98%
9	9095.60	2291	0.41%	0.60%	8861.89	136	2.64%	2.38%	2.95%	8833.89	279.71	2.96%	2.87%	3.05%
10	4432.53	139	0.01%	0.01%	4319.65	65	2.62%	1.94%	3.21%	4384.14	88.67	1.11%	0.92%	1.37%
11	6897.25	1004	0.01%	0.03%	6729.05	106	2.50%	2.02%	2.91%	6825.40	159.68	1.06%	0.50%	1.40%
12	9233.98	309	0.01%	0.01%	9060.61	149	1.91%	1.81%	2.11%	9132.72	246.54	1.11%	1.04%	1.15%
13	3622.72	4472	1.18%	1.40%	3440.36	70	5.29%	4.43%	6.34%	3518.27	104.93	2.97%	2.71%	3.39%
14	5808.76	5653	3.20%	4.24%	5533.77	111	4.96%	4.17%	6.18%	5643.11	195.77	2.94%	2.67%	3.18%
15	8183.71	17494	5.86%	6.28%	7841.62	198	4.37%	3.80%	5.23%	7926.21	305.16	3.25%	3.02%	3.39%
16	4156.17	515	0.02%	0.04%	3947.00	58	5.30%	4.78%	5.68%	4065.60	92.14	2.23%	2.01%	2.38%
17	6478.19	1113	0.03%	0.04%	6264.25	101	3.42%	3.21%	3.69%	6400.00	165.95	1.22%	1.05%	1.41%
18	8967.14	1009	1.84%	2.75%	8713.02	152	2.92%	2.73%	3.09%	8859.54	265.06	1.22%	1.06%	1.35%
19	3452.68	4303	2.06%	2.19%	3254.77	68	6.08%	6.05%	6.10%	3364.97	105.40	2.61%	2.45%	2.82%
20	5643.51	13028	5.70%	5.87%	5343.31	112	5.62%	4.93%	6.24%	5509.02	211.88	2.44%	2.22%	2.70%
21	7857.91	33205	10.66%	12.34%	7540.51	212	4.22%	3.31%	5.00%	7671.20	320.85	2.43%	2.20%	2.59%
22	4128.93	363	0.00%	0.00%	3857.59	54	7.04%	6.61%	7.56%	4010.85	90.85	2.95%	2.65%	3.26%
23	6294.43	899	0.18%	0.48%	6050.90	94	4.03%	3.83%	4.30%	6217.72	160.96	1.23%	1.03%	1.39%
24	8637.68	8106	3.42%	3.86%	8367.05	144	3.24%	2.63%	3.83%	8561.30	274.12	0.89%	0.75%	1.06%
25	4805.10	27	0.00%	0.00%	4774.74	62	0.64%	0.45%	0.87%	4751.00	92.38	1.14%	1.08%	1.21%
26	7291.23	79	0.01%	0.01%	7202.09	89	1.24%	1.05%	1.42%	7213.34	170.01	1.08%	0.93%	1.27%
27	9907.64	92	0.01%	0.01%	9762.33	121	1.49%	1.26%	1.67%	9837.51	269.29	0.71%	0.61%	0.81%
28	4892.27	28	0.00%	0.01%	4885.71	59	0.13%	0.10%	0.20%	4887.86	82.48	0.09%	0.06%	0.13%
29	7404.10	51	0.01%	0.01%	7374.19	88	0.41%	0.39%	0.43%	7382.79	150.08	0.29%	0.24%	0.32%
30	9922.47	103	0.01%	0.01%	9855.42	124	0.68%	0.59%	0.80%	9895.49	238.26	0.27%	0.23%	0.30%
31	4045.82	604	0.13%	0.30%	3866.70	69	4.63%	4.35%	4.78%	3905.66	99.12	3.59%	3.28%	3.92%
32	6427.72	1507	0.51%	0.87%	6207.56	103	3.55%	3.20%	4.07%	6144.08	177.14	4.62%	4.50%	4.84%
33	9061.64	1735	0.80%	2.36%	8778.31	139	3.23%	2.35%	4.09%	8673.73	275.18	4.47%	4.25%	4.79%
34	4322.18	135	0.01%	0.01%	4204.05	65	2.81%	2.70%	2.90%	4250.99	88.63	1.67%	1.45%	1.79%
35	6834.61	388	0.01%	0.01%	6677.17	105	2.36%	2.17%	2.63%	6716.01	159.02	1.77%	1.62%	2.00%
36	9262.91	337	0.01%	0.01%	9072.59	149	2.10%	2.03%	2.23%	9113.75	245.95	1.64%	1.54%	1.75%
37	3634.56	1998	0.99%	1.73%	3388.12	69	7.27%	6.90%	7.66%	3501.67	105.05	3.79%	3.64%	4.04%
38	5802.38	3222	3.19%	5.36%	5499.78	110	5.50%	4.22%	7.23%	5528.98	194.00	4.94%	4.66%	5.19%
39	8127.71	13374	3.98%	4.58%	7801.54	202	4.18%	3.70%	4.96%	7750.68	301.77	4.87%	4.52%	5.30%
40	4163.47	296	0.01%	0.01%	3948.64	58	5.44%	4.60%	6.00%	4068.71	91.45	2.33%	2.26%	2.46%
41	6389.48	735	0.07%	0.17%	6193.02	99	3.18%	2.56%	3.61%	6297.86	163.21	1.45%	1.33%	1.53%
42	8806.23	1597	1.48%	1.89%	8532.16	150	3.21%	3.07%	3.30%	8653.07	259.78	1.77%	1.63%	1.85%
43	3431.07	2974	1.70%	2.26%	3256.22	68	5.37%	4.93%	5.68%	3320.86	104.19	3.32%	2.76%	3.75%
44	5601.71	15260	4.70%	5.47%	5297.08	110	5.76%	4.43%	6.60%	5388.08	207.99	3.97%	3.75%	4.33%
45	7910.41	20131	8.91%	9.70%	7552.17	157	4.74%	4.51%	4.94%	7566.43	317.08	4.55%	4.26%	4.76%
46	4048.49	217	0.01%	0.01%	3798.64	56	6.58%	6.12%	7.04%	3919.53	88.66	3.29%	3.28%	3.30%
47	6359.29	700	0.01%	0.03%	6111.16	95	4.07%	3.22%	4.60%	6251.55	165.21	1.73%	1.42%	1.97%
48	8706.85	1148	1.21%	1.54%	8457.79	144	2.95%	2.71%	3.38%	8576.43	263.17	1.52%	1.39%	1.71%
49	4694.49	31	0.01%	0.01%	4641.29	62	1.14%	0.82%	1.49%	4605.02	90.01	1.94%	1.64%	2.10%
50	7351.26	59	0.01%	0.01%	7220.07	93	1.82%	1.60%	1.97%	7191.16	166.25	2.23%	2.10%	2.43%
51	9729.10	159	0.01%	0.01%	9515.41	130	2.25%	2.00%	2.46%	9609.31	268.67	1.25%	0.96%	1.40%
52	4847.10	28	0.01%	0.01%	4837.01	58	0.21%	0.06%	0.30%	4821.83	80.00	0.52%	0.29%	0.73%
53	7189.56	45	0.00%	0.01%	7128.71	90	0.85%	0.78%	0.99%	7142.67	145.73	0.66%	0.57%	0.78%

	ILP				ALNS					MMAS				
Exp	Obj.	Run	Avg.	Max.	Obj.	Run	Avg.	Max.	Min.	Obj.	Run	Avg.	Max.	Min.
54	9772.28	115	0.01%	0.01%	9665.60	127	1.10%	1.03%	1.21%	9714.48	229.03	0.60%	0.49%	0.67%
55	3820.56	209	0.00%	0.01%	3728.36	67	2.49%	1.35%	3.21%	3717.10	96.55	2.79%	2.45%	3.16%
56	6098.11	673	0.04%	0.11%	5973.53	100	2.09%	1.86%	2.44%	5912.69	172.74	3.14%	2.94%	3.43%
57	8668.89	560	0.02%	0.03%	8458.97	133	2.48%	2.15%	2.71%	8353.80	271.37	3.77%	3.58%	3.91%
58	4283.18	61	0.01%	0.01%	4169.09	63	2.73%	2.02%	3.29%	4245.83	84.22	0.88%	0.82%	0.91%
59	6749.47	193	0.01%	0.01%	6602.20	102	2.23%	2.21%	2.26%	6663.85	152.40	1.29%	1.09%	1.39%
60	9065.04	193	0.00%	0.01%	8921.69	144	1.61%	1.39%	1.74%	8952.84	237.44	1.25%	1.22%	1.28%
61	3425.95	506	0.27%	0.80%	3288.66	69	4.17%	4.05%	4.40%	3327.08	100.89	2.97%	2.72%	3.33%
62	5450.43	1028	0.47%	0.81%	5264.35	107	3.53%	3.29%	3.78%	5279.51	188.89	3.24%	2.84%	3.48%
63	7779.36	2654	2.35%	3.19%	7468.48	147	4.16%	3.42%	4.59%	7537.47	291.07	3.21%	2.81%	3.44%
64	4121.77	187	0.01%	0.01%	3913.45	57	5.32%	5.08%	5.46%	4035.94	84.79	2.13%	1.84%	2.38%
65	6482.76	425	0.01%	0.01%	6247.54	99	3.76%	3.64%	3.86%	6403.38	155.38	1.24%	1.16%	1.28%
66	8654.35	753	0.01%	0.02%	8440.87	149	2.53%	2.43%	2.70%	8542.02	248.74	1.31%	1.16%	1.46%
67	3378.47	245	0.00%	0.01%	3164.75	66	6.75%	6.42%	7.03%	3264.43	102.39	3.49%	2.87%	3.93%
68	5338.90	2071	1.12%	1.51%	5097.64	109	4.73%	4.42%	5.04%	5173.91	193.13	3.19%	3.15%	3.24%
69	7565.09	7319	4.09%	6.54%	7214.95	152	4.86%	3.75%	5.76%	7329.47	305.40	3.21%	2.89%	3.44%
70	4099.38	270	0.01%	0.01%	3870.97	56	5.90%	5.16%	6.51%	4004.54	86.47	2.37%	1.85%	2.83%
71	6343.46	2992	0.18%	0.27%	6081.88	94	4.30%	4.09%	4.64%	6244.43	159.05	1.59%	1.44%	1.74%
72	8577.49	509	0.01%	0.02%	8317.01	142	3.13%	2.99%	3.36%	8451.75	251.38	1.49%	1.27%	1.62%
73	4364.12	72	0.00%	0.01%	4345.22	56	0.43%	0.34%	0.55%	4359.66	65.52	0.10%	0.02%	0.15%
74	7149.39	30	0.00%	0.01%	7140.65	81	0.12%	0.00%	0.20%	7117.63	111.97	0.45%	0.35%	0.56%
75	9648.08	61	0.00%	0.01%	9627.32	107	0.21%	0.12%	0.27%	9644.56	168.42	0.04%	0.01%	0.08%
76	4527.97	29	0.00%	0.01%	4516.71	55	0.25%	0.19%	0.32%	4525.65	61.82	0.05%	0.00%	0.08%
77	7189.87	42	0.00%	0.00%	7189.34	76	0.01%	0.00%	0.02%	7189.74	102.31	0.00%	-0.00%	0.01%
78	9816.09	62	0.00%	0.00%	9807.70	100	0.09%	0.05%	0.11%	9816.10	150.49	-0.00%	-0.00%	0.00%
79	3435.61	468	0.12%	0.37%	3371.53	60	1.90%	1.73%	2.03%	3371.66	70.03	1.90%	1.63%	2.12%
80	5578.46	1001	0.64%	0.74%	5441.48	90	2.52%	2.16%	3.24%	5490.52	124.42	1.61%	1.25%	2.15%
81	7996.11	753	0.17%	0.42%	7884.72	119	1.41%	1.20%	1.58%	7957.54	191.95	0.48%	0.39%	0.54%
82	4032.57	74	0.01%	0.01%	4006.55	59	0.65%	0.56%	0.74%	4007.98	56.05	0.62%	0.27%	0.99%
83	6340.58	184	0.01%	0.01%	6302.07	93	0.61%	0.42%	0.76%	6279.87	98.83	0.97%	0.83%	1.22%
84	8793.85	467	0.01%	0.02%	8739.25	128	0.62%	0.49%	0.78%	8727.24	164.51	0.76%	0.69%	0.83%
85	3034.93	281	0.00%	0.01%	2977.78	63	1.92%	1.77%	2.16%	2967.43	66.27	2.27%	1.75%	2.85%
86	4930.07	4182	2.04%	2.77%	4809.87	104	2.50%	1.82%	2.87%	4778.75	118.71	3.17%	2.84%	3.57%
87	6977.08	11775	3.70%	4.22%	6790.64	149	2.75%	2.56%	2.91%	6802.56	195.62	2.57%	2.14%	2.98%
88	3944.24	117	0.01%	0.01%	3878.76	54	1.69%	1.61%	1.80%	3918.96	51.36	0.65%	0.61%	0.71%
89	5985.58	286	0.01%	0.01%	5947.55	91	0.64%	0.55%	0.72%	5942.40	90.29	0.73%	0.49%	0.87%
90	8161.11	269	0.00%	0.00%	8106.61	136	0.67%	0.54%	0.89%	8078.18	148.28	1.03%	0.92%	1.22%
91	3055.85	247	0.01%	0.01%	3005.93	62	1.66%	1.34%	2.12%	2989.24	63.60	2.23%	2.14%	2.29%
92	4773.80	2101	2.56%	4.18%	4666.97	99	2.29%	2.08%	2.44%	4634.34	116.92	3.02%	2.48%	3.29%
93	6657.48	26143	4.99%	6.12%	6478.88	180	2.76%	2.54%	2.88%	6431.29	197.18	3.52%	2.96%	4.21%
94	3946.04	279	0.01%	0.02%	3858.16	50	2.28%	2.15%	2.49%	3911.16	49.73	0.89%	0.78%	1.03%
95	5920.63	254	0.01%	0.01%	5869.48	89	0.87%	0.55%	1.05%	5864.11	87.78	0.96%	0.95%	0.97%
96	8108.35	325	0.00%	0.01%	8053.98	131	0.68%	0.51%	0.79%	8013.96	145.30	1.18%	0.95%	1.31%
97	4389.15	43	0.00%	0.01%	4366.00	55	0.53%	0.16%	0.94%	4381.69	65.10	0.17%	0.11%	0.21%
98	7178.44	73	0.00%	0.01%	7174.51	81	0.05%	0.04%	0.07%	7160.87	111.05	0.25%	0.06%	0.35%
99	9549.44	64	0.00%	0.00%	9529.23	103	0.21%	0.20%	0.23%	9545.74	167.23	0.04%	0.02%	0.06%
100	4630.84	24	0.00%	0.01%	4614.01	54	0.37%	0.21%	0.48%	4629.84	61.50	0.02%	0.02%	0.03%
101	7359.44	31	0.01%	0.01%	7359.04	74	0.01%	-0.00%	0.02%	7359.45	100.84	-0.00%	-0.00%	0.00%
102	9721.76	52	0.00%	0.01%	9719.14	99	0.03%	0.00%	0.05%	9721.75	150.47	0.00%	-0.00%	0.00%
103	3447.71	466	0.09%	0.25%	3363.98	61	2.49%	1.98%	3.22%	3377.69	69.51	2.08%	1.63%	2.73%

	ILP				ALNS					MMAS				
Exp	Obj.	Run	Avg.	Max.	Obj.	Run	Avg.	Max.	Min.	Obj.	Run	Avg.	Max.	Min.
104	5627.86	763	0.45%	0.68%	5510.74	89	2.13%	1.76%	2.34%	5542.39	123.99	1.54%	1.33%	1.69%
105	7869.06	796	0.20%	0.32%	7774.91	119	1.21%	0.80%	1.55%	7836.54	189.30	0.41%	0.22%	0.54%
106	4094.97	72	0.01%	0.01%	4076.74	59	0.45%	0.40%	0.49%	4071.60	55.45	0.57%	0.47%	0.70%
107	6429.25	330	0.02%	0.04%	6388.50	93	0.64%	0.56%	0.69%	6363.19	98.12	1.04%	0.81%	1.26%
108	8642.73	394	0.01%	0.01%	8574.60	129	0.80%	0.74%	0.88%	8577.51	161.31	0.76%	0.72%	0.83%
109	3104.25	305	0.01%	0.01%	3050.48	63	1.76%	1.68%	1.83%	3033.91	64.93	2.32%	1.94%	2.72%
110	4936.31	3915	1.95%	2.42%	4829.22	95	2.22%	1.89%	2.41%	4763.97	118.91	3.62%	2.83%	4.49%
111	6998.90	22473	4.96%	6.69%	6825.15	128	2.55%	2.29%	2.77%	6822.17	199.63	2.59%	2.14%	3.03%
112	3926.87	68	0.00%	0.01%	3863.42	51	1.64%	1.51%	1.82%	3894.88	51.03	0.82%	0.72%	0.89%
113	6065.60	209	0.01%	0.01%	6023.91	86	0.69%	0.56%	0.92%	6010.57	89.72	0.92%	0.85%	1.03%
114	8213.20	287	0.01%	0.01%	8165.60	130	0.58%	0.44%	0.69%	8128.15	146.81	1.05%	0.98%	1.18%
115	3007.18	308	0.03%	0.09%	2942.74	61	2.19%	1.71%	2.60%	2933.68	64.08	2.51%	1.87%	3.19%
116	4744.33	2124	1.77%	2.47%	4658.93	96	1.83%	1.40%	2.08%	4599.19	118.21	3.16%	2.86%	3.72%
117	6738.27	17395	4.88%	6.05%	6564.28	128	2.65%	2.41%	3.11%	6500.37	194.92	3.66%	3.47%	3.86%
118	3879.82	7165	0.03%	0.04%	3809.42	49	1.85%	1.25%	2.63%	3858.40	50.20	0.55%	0.47%	0.61%
119	5993.35	232	0.01%	0.01%	5949.43	80	0.74%	0.67%	0.81%	5931.87	87.51	1.04%	0.96%	1.12%
120	8126.78	364	0.00%	0.01%	8074.14	124	0.65%	0.58%	0.76%	8045.30	146.08	1.01%	0.93%	1.18%
121	4414.96	66	0.00%	0.01%	4393.40	53	0.50%	0.17%	0.95%	4409.45	65.05	0.13%	0.06%	0.19%
122	7165.23	33	0.00%	0.00%	7156.98	76	0.12%	0.03%	0.23%	7111.75	110.10	0.75%	0.66%	0.90%
123	9574.51	55	0.00%	0.01%	9546.88	100	0.29%	0.18%	0.44%	9567.46	167.55	0.07%	0.02%	0.15%
124	4626.42	24	0.00%	0.01%	4618.35	51	0.17%	0.04%	0.26%	4624.12	61.57	0.05%	0.00%	0.11%
125	7184.48	34	0.01%	0.01%	7184.51	71	-0.00%	-0.00%	-0.00%	7184.09	100.55	0.01%	-0.00%	0.02%
126	9791.84	70	0.00%	0.01%	9788.13	96	0.04%	0.00%	0.06%	9791.09	150.08	0.01%	-0.00%	0.03%
127	3424.06	491	0.19%	0.56%	3355.91	58	2.04%	1.39%	2.92%	3351.76	69.15	2.16%	1.64%	2.75%
128	5620.04	730	0.53%	1.58%	5495.40	85	2.27%	1.89%	2.53%	5542.25	122.81	1.42%	0.86%	1.94%
129	7906.55	962	0.27%	0.58%	7782.53	114	1.60%	1.35%	2.03%	7857.36	187.31	0.63%	0.47%	0.76%
130	4087.31	72	0.00%	0.01%	4068.73	57	0.46%	0.39%	0.50%	4054.84	55.97	0.80%	0.68%	0.95%
131	6277.29	198	0.00%	0.00%	6232.53	88	0.72%	0.40%	1.16%	6217.13	98.76	0.97%	0.88%	1.05%
132	8727.65	704	0.03%	0.08%	8674.27	122	0.62%	0.54%	0.69%	8660.29	159.37	0.78%	0.61%	0.94%
133	3082.98	273	0.01%	0.01%	3022.29	60	2.01%	1.38%	2.75%	3003.54	65.00	2.65%	2.27%	2.89%
134	4936.30	2889	0.98%	1.58%	4820.96	90	2.39%	1.94%	2.78%	4787.55	118.41	3.11%	2.79%	3.46%
135	7005.19	7403	2.61%	3.15%	6802.84	122	2.98%	2.86%	3.18%	6800.44	218.31	3.02%	2.81%	3.38%
136	3919.05	78	0.01%	0.01%	3866.45	51	1.36%	1.06%	1.74%	3892.13	51.21	0.69%	0.28%	0.92%
137	6111.60	174	0.01%	0.01%	6081.95	87	0.49%	0.35%	0.64%	6045.34	91.73	1.10%	1.06%	1.13%
138	8198.21	423	0.01%	0.01%	8144.45	128	0.66%	0.46%	0.99%	8095.42	146.30	1.27%	1.12%	1.43%
139	2996.26	171	0.01%	0.01%	2944.13	60	1.78%	1.26%	2.47%	2935.30	63.79	2.08%	2.07%	2.09%
140	4757.47	1275	1.14%	1.87%	4648.11	94	2.35%	1.83%	2.86%	4563.65	115.65	4.25%	4.00%	4.42%
141	6682.17	16735	8.23%	12.46%	6499.09	127	2.82%	2.61%	2.99%	6451.22	193.88	3.58%	3.18%	3.98%
142	3934.95	704	0.02%	0.05%	3858.66	48	1.98%	1.74%	2.38%	3916.46	49.18	0.47%	0.37%	0.54%
143	5951.61	204	0.01%	0.01%	5898.95	81	0.89%	0.58%	1.13%	5887.38	86.74	1.09%	0.90%	1.29%
144	8148.55	415	0.01%	0.01%	8087.43	162	0.76%	0.51%	0.92%	8065.04	144.35	1.03%	0.75%	1.35%