A Lagrangian Relaxation-based Heuristic to Solve Large Extended Graph Partitioning Problems

Oliver G. Czibula[1,2], Hanyu Gu[2], and Yakov Zinder[2]

[1] Ausgrid, 570 George Street Sydney NSW 2000, Australia
[2] School of Mathematical and Physical Sciences, University of Technology Sydney
15 Broadway Ultimo NSW 2007, Australia

Abstract. The paper is concerned with the planning of training sessions in large organisations requiring periodic retraining of their staff. The allocation of students must take into account student preferences as well as the desired composition of study groups. The paper presents a bicriteria Quadratic Multiple Knapsack formulation of the considered practical problem, and a novel solution procedure based on Lagrangian relaxation. The paper presents the results of computational experiments aimed at testing the optimisation procedure on real world data originating from Australia's largest electricity distributor. Results are compared and validated against a Genetic Algorithm based matheuristic.

1 Introduction

This paper is concerned with the optimisation of class formation at large organisations, typically with thousands of workers of different types, that require periodic retraining of their staff. Finding good solutions to this problem is important as it allows more effective training sessions to be provided.

This research is motivated by the problem of providing training to workers at Ausgrid, Australia's largest electricity distributor. Due to the multitude of hazards that exist when working with high voltages at heights or in confined spaces, Ausgrid is required by Australian law to deliver regular safety, technical, and professional training to all its employees who work on or near the electricity network. Ausgrid provides regular training to thousands of employees, contractors, and third parties. Many of these people have very different learning outcomes from courses, different learning styles, different levels of education or English proficiency, and different levels of technical proficiency for certain tasks.

Consider an example where field workers and upper management are undertaking a particular course: while the core material would remain the same for both groups, if they are taught in separate classes the trainers can take the opportunity to better tailor the delivery to the specific needs of their group, allowing for more productive training sessions. It is often not possible to run segregated classes due to the scarcity of training resources and the associated cost of delivering additional classes, therefore some blending of different student types is often necessary. Ideally, differing student types should be combined into a single class only when they have a high compatibility with one another.

In similar problems where students are preferentially assigned to classes, it is customary to give each student-class pair a weight (or cost). Most of the training provided by Ausgrid has a limited period in which it is valid, and therefore courses should be periodically retrained. Workers are only permitted to work in roles for which they have up-to-date training. The date at which a worker's training expires is considered their due date for that course. If there are a number of scheduled classes for a given course that a worker requires, we can designate a cost of assigning the worker to any one of those classes: classes that run on or before their due date have low or zero cost, while classes that run after the due date have a high cost that increases the later the class is scheduled.

The considered problem has a bi-criteria objective: we wish to find an assignment of students to classes that minimises incompatibility between student types within classes, and that minimises the assignment cost of individual students to classes. Each class has a minimum and maximum number of students it can hold, and of student types it can be assigned.

Johnson et al. [10] discuss the problem of partitioning the vertices of a graph $G(V, E)$ with nonnegative weights w_v, $v \in V$ and costs c_e, $e \in E$, into K disjoint clusters partitioning V, such that the sum of the weights in each cluster is bounded between $w_{\min}$ and $w_{\max}$ and the sum of the costs within each cluster is minimised. This problem is known as the graph partitioning problem, and is known to be NP-hard [4], [7]. Johnson et al. proposed a column generation approach, and tested the approach on graphs with between 30 and 61 nodes, and between 47 and 187 edges. For the 12 test cases the authors considered, their proposed approach provided integer solutions for all but two, and for those, solutions obtained by a branch-and-bound scheme were very close to the fractional solutions provided by column generation.

The problem considered in this paper can be modelled as a generalised graph partitioning problem. Each student j is represented by a vertex $v_j \in V$; the preference between students i and j is represented by an undirected edge $e = (v_i, v_j) \in E$ with edge cost c_e. The problem is to find a partition $\varGamma = \{W_1, W_2, \dots, W_N\}$ of V that solves

$$\text{Minimise:} \quad \alpha \sum_{i=1}^{N} \sum_{e \in E(W_i)} c_e + \beta \sum_{i=1}^{N} \sum_{v_k \in W_i} w^i_{v_k} \tag{1}$$

$$\text{Subject To:} \quad w_{min} \le |W_i| \le w_{max} \quad i = 1, \dots, k \tag{2}$$

where N is the number of classes, W_i is the set of students assigned to class i, $E(W_i) = \{(v_k, v_l) | v_k \in W_i,\ v_l \in W_i\}$, $w^i_{v_k}$ is cost of assigning student k to class i, α and β are weights for the assigning cost and preference cost respectively. A special case of the problem considered in this paper, in which $w^1_{v_k} = w^2_{v_k} = \ldots = w^N_{v_k}$, is equivalent to the graph partitioning problem. Therefore our problem is also NP-hard.

Chopra and Rao [3] discuss several forms of the graph partitioning problem as well as IP models for each. The authors do not assume a complete graph, allowing them to take advantage of the graph structure when clustering. They also discuss several valid inequalities and facet-defining inequalities for the GPP.

The considered problem can also be modelled as an extended Quadratic Multiple Knapsack Problem (QMKP) with additional constraints. The QMKP is a generalization and combination of the well-known multiple knapsack problem and the quadratic knapsack problem. The QMKP received little attention in the literature until recently, and most solution approaches are based on metaheuristics [2], [6], [9]. The main contributions of this paper include: ***(i)*** formulation of the considered practical problem as an extended bicriteria QMKP. ***(ii)*** design of a Lagrangian relaxation (LR) based, fast heuristic capable of solving large, real-world instances.

Caprara et al. [1] discuss an LR-based approach to solving the QMKP exactly, whereby a tighter upper bound (for their maximisation objective) is computed using the subgradient method in linear expected time. The presented approach is able to solve instances with up to 400 binary variables exactly. The authors note that the presented approach can also be used to solve Max Clique problems almost as fast as with the cutting plane approaches available at the time.

Julstrom [11] discusses greedy, genetic, and greedy genetic algorithms for the QMKP. The author presents two greedy heuristics that build solutions by choosing objects according to their value densities, and two genetic algorithm (GA) heuristics. One GA is a standard implementation, whereas the other is extended with greedy techniques that probabilistically favour objects of high value density. The four algorithms are tested on 20 problem instances, and the extended GA is reported to perform best on all but one test case.

The remainder of this paper is organised as follows: Section 2 introduces a quadratic programming formulation and its linearisation for the considered problem; Section 3 introduces a Lagrangian relaxation formulation for the quadratic programming model; Section 4 discusses the proposed LR-based heuristic technique; Section 5 presents a GA matheuristic for the problem; Section 6 presents the computational results of the proposed heuristic on a number of industry-inspired test cases; and Section 7 discusses our conclusions and outlines possibilities for future research.

2 Quadratic Programming Formulation

Let $\mathcal{N} = \{1, \cdots, N\}$, $\mathcal{M} = \{1, \cdots, M\}$, and $\mathcal{K} = \{1, \cdots, K\}$ be the set of classes available, the set of students to be assigned, and the set of student types respectively. Denote the cost of assigning students $j \in \mathcal{M}$ to class $i \in \mathcal{N}$ by $c_{i,j}$, the cost of pairing student types $k \in \mathcal{K}$ and $l \in \mathcal{K}$ together in the same class by $b_{k,l}$. Each student has exactly one type. The set of students who are of type k is represented by T_k, $k \in \mathcal{K}$. Each student must be assigned to exactly one class, but not all classes need to be run. Each class $i \in \mathcal{N}$ that is run must contain at least a_i and at most b_i students, and at least p_i and at most q_i student types.

Students or student types cannot be assigned to any class that is not run. In this problem, we assume there are more scheduled classes than needed, i.e. a feasible solution always exists. The following Quadratic Program describes the problem:

$$\text{(QP) Minimise:} \quad \alpha \sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{l=1}^{K} b_{k,l} Y_{i,k} Y_{i,l} + \beta \sum_{i=1}^{N} \sum_{j=1}^{M} c_{i,j} X_{i,j} \tag{3}$$

$$\text{Subject To:} \quad \sum_{i=1}^{N} X_{i,j} = 1 \quad j = 1, \ldots, M \tag{4}$$

$$a_i Z_i \leq \sum_{j=1}^{M} X_{i,j} \leq b_i Z_i \quad i = 1, \ldots, N \tag{5}$$

$$p_i Z_i \leq \sum_{k=1}^{K} Y_{i,k} \leq q_i Z_i \quad i = 1, \ldots, N \tag{6}$$

$$X_{i,j} \leq Y_{i,k} \quad i = 1, \ldots, N; k = 1, \ldots, K; j \in T_k \tag{7}$$

$$X_{i,j} \in \{0, 1\} \quad i = 1, \ldots, N; j = 1, \ldots, M \tag{8}$$

$$Y_{i,k} \in \{0, 1\} \quad i = 1, \ldots, N; k = 1, \ldots, K \tag{9}$$

$$Z_i \in \{0, 1\} \quad i = 1, \ldots, N \tag{10}$$

where the binary variable $X_{i,j}$ is defined to be 1 if student j is assigned to class i, or 0 otherwise; the binary variable $Y_{i,k}$ is defined to be 1 if student type k is assigned to class i, or 0 otherwise; The binary variable Z_i is defined to be 1 if class i is run, or 0 otherwise.

In the objective function (3), the quadratic term represents the cost of pairing student types together, and the linear term represents the cost of assigning students to classes, weighted by coefficients α and β, respectively.

The constraints (4) express the requirement that each student be assigned to exactly one class. The constraints (5) and (6) express the requirement that each running class has between a_i and b_i students and between p_i and q_i student types, respectively, if the class is run, or zero otherwise. The constraints (7) express the requirement that a student may only be assigned to a class if that student's type has also been assigned to that class.

It is possible to linearise the quadratic term in (3) by introducing $\hat{Y}_{i,k,l} = Y_{i,k} Y_{i,l}$ together with constraints:

$$\hat{Y}_{i,k,l} \leq Y_{i,k} \quad i = 1, \ldots, N; k = 1, \ldots, K; l = 1, \ldots, K \tag{11}$$

$$\hat{Y}_{i,k,l} \leq Y_{i,l} \quad i = 1, \ldots, N; k = 1, \ldots, K; l = 1, \ldots, K \tag{12}$$

$$\hat{Y}_{i,k,l} \geq Y_{i,k} + Y_{i,l} - 1 \quad i = 1, \ldots, N; k = 1, \ldots, K; l = 1, \ldots, K \tag{13}$$

to give the linearised model:

$$\text{(LQP) Minimise: } \alpha \sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{l=1}^{K} b_{k,l} \hat{Y}_{i,k,l} + \beta \sum_{i=1}^{N} \sum_{j=1}^{M} c_{i,j} X_{i,j} \tag{14}$$

$$\text{Subject To:} \quad (4) - (13) \tag{15}$$

$$\hat{Y}_{i,k,l} \in \{0,1\} \quad i = 1,\dots,N; k = 1,\dots,K; l = 1,\dots,K \tag{16}$$

The (QP) model has $N(K+M+1)$ variables and $M+2N+MNK$ constraints, whereas the (LQP) model has $N(K+M^2+M+1)$ variables and $M+2N+MNK+3NK^2$ constraints.

The time required to find an optimal solution to the (QP) and (LQP) models grows rapidly, where even some small test cases with just a few dozen students can take hours to solve. As the problem instances we hope to solve are significantly larger than this, we propose to use a heuristic approach.

3 Lagrangian Relaxation

To improve the convergence of the subgradient algorithm, the equalities (4) are first converted into inequalities:

$$\sum_{i=1}^{N} X_{i,j} \geq 1 \quad j = 1,\dots,M \tag{17}$$

By moving the constraints (4) into the objective function, we obtain the Lagrangian relaxation model:

$$\text{(QR) Minimise: } \alpha \sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{l=1}^{K} b_{k,l} Y_{i,k} Y_{i,l} + \beta \sum_{i=1}^{N} \sum_{j=1}^{M} c_{i,j} X_{i,j} + \sum_{j=1}^{M} \lambda_j (1 - \sum_{i=1}^{N} X_{i,j}) \tag{18}$$

$$\text{Subject To:} \quad (5) - (10) \tag{19}$$

where $\lambda_j \geq 0$, $j = 1,\dots,M$, is the Lagrangian multiplier corresponding to the cost of not assigning student j to any class. The quadratic term in (18) can be linearised in the same way as with (3).

In (QP), only constraints (4) couple together the N sub-problems of assigning students to *a particular* class i. In (QR), these constraints are moved to the objective function, therefore it is possible to express (QR) as N smaller, class-specific sub-problems (QR$_i$):

$$\text{(QR}_i\text{) Minimise: } \alpha \sum_{k=1}^{K} \sum_{l=1}^{K} b_{k,l} Y_{i,k} Y_{i,l} + \beta \sum_{j=1}^{M} c_{i,j} X_{i,j} - \sum_{j=1}^{M} \lambda_j X_{i,j} \tag{20}$$

$$\text{Subject To:} \quad a_i Z_i \leq \sum_{j=1}^{M} X_{i,j} \leq b_i Z_i \tag{21}$$

$$p_i Z_i \leq \sum_{k=1}^{K} Y_{i,k} \leq q_i Z_i \tag{22}$$

$$X_{i,j} \leq Y_{i,k} \quad k = 1, \dots, K; j \in T_k \tag{23}$$

$$X_{i,j} \in \{0,1\} \quad j = 1, \dots, M \tag{24}$$

$$Y_{i,k} \in \{0,1\} \quad k = 1, \dots, K \tag{25}$$

$$Z_i \in \{0,1\} \tag{26}$$

where the N combined solutions to (QR$_i$) for $1 \leq i \leq N$ forms the solution to (QR).

Solving the Lagrangian relaxation (QR) provides a lower bound to the optimal objective value of (QP). We wish to find the tightest possible lower bound for (QP) by solving the Lagrangian dual problem, the solution of which are the optimal Lagrangian multipliers λ^* that give the largest objective value of (QR) [8], [5]. A typical way to numerically approximate λ^* is by using the iterative subgradient algorithm.

In the subgradient algorithm, λ is iteratively updated with the relationship

$$\lambda^{k+1} = \lambda^k + \frac{s^k \cdot \epsilon_k(\eta^* - \eta^k)}{||s^k||^2} \tag{27}$$

where λ^k is the value of the Lagrangian multipliers λ at the kth iteration; η^* is an upper bound of the optimal objective value of the Lagrangian dual problem, and η^k is the objective value of (QR) at the kth iteration; and ϵ_k is a positive scaling factor, typically with the initial value of $\epsilon_0 = 2$ and halved whenever the objective value hasn't been improved in certain number of iterations; s^k is a subgradient of the Lagrangian dual at iteration k given by

$$s_j^k = 1 - \sum_{i=1}^{N} X_{i,j}^k \quad j = 1, \dots, M \tag{28}$$

where X^k is from the solution of (QR) at the kth iteration.

4 Lagrangian Heuristic

In our proposed heuristic, we solve the Lagrangian dual using the subgradient algorithm. In doing so, we generate many solutions in this iterative process, and we record the following characteristics about them:

- The number of times class i is running (ξ_i), and the number of times it is not ($\bar{\xi}_i$).
- The number of times type k is assigned to class i ($\phi_{i,k}$), and the number of times it is not ($\bar{\phi}_{i,k}$).

– The number of times students j_1 and j_2 are assigned to the same class (ψ_{j_1,j_2}), and the number of times they are not ($\bar{\psi}_{j_1,j_2}$).

Based on these values, one or more assumptions are made about the final solution with some probability. The higher the value, the greater the probability the assumption will be made. Each value has a corresponding counter-value, such as the number of times class i is running (ξ_i), and the number of times class i is not running ($\bar{\xi_i}$). An assumption can be made according to the value/counter-value that has greater magnitude, for example if $\xi_i = 7649$ and $\bar{\xi_i} = 1008$ then, since $\xi_i > \bar{\xi_i}$, we would be making the assumption that class i will be run, with some probability. The probability with which we make an assumption according to some value v (or counter-value $\bar{v}$) is $\frac{v}{v+\bar{v}}^2$ (or $\frac{\bar{v}}{v+\bar{v}}^2$), so if $\xi_i = 7649$ and $\bar{\xi_i} = 1008$ then with probability $(\frac{7649}{7649+1008})^2 \approx 0.7807$ we will make the assumption that class i will run in the final solution.

Each kind of imposed assumption reduces the number of feasible solutions of the model being solved. In the case of the first kind of assumption, the Z_i variable is fixed to 1, if class i is assumed running, or if class i is assumed not running then the Z_i and the related $X_{i,j}$ and $Y_{i,k}$ variables, $1 \leq j \leq M$ and $1 \leq k \leq K$, can be all fixed to zero. For the assumption that type k is or is not assigned to class i, the constraints (7) can be omitted for $j \in T_k$, and all the related $X_{i,j}$ variables are also fixed to zero if type k is assumed to be not assigned to class i. For the assumption that students j_1 and j_2 should be assigned together, we introduce the additional constraints $X_{i,j_1} = X_{i,j_2}$ for $1 \leq i \leq N$, and for the assumption that they should be assigned separately, we introduce the constraints $X_{i,j_1} + X_{i,j_2} \leq 1$ for $1 \leq i \leq N$.

Once an assumption is made, the model is updated and the Lagrangian dual is once again solved using the subgradient algorithm. With each new assumption the size of the model becomes smaller or the number of feasible solutions is reduced. Once the model is sufficiently small, the optimal solution, subject to the assumptions, can be found using a commercial solver.

The initial value of $\lambda = \lambda^0$ is likely to be quite distant from λ^*. It is expected that there would be an initial period of convergence from λ^0 towards the neighbourhood of λ^*, followed by a period of convergence within this neighbourhood. Since these early values of λ^k are likely to produce fairly poor solutions, we treat this early period of convergence towards the neighbourhood of λ^* as a "burn-in stage", and do not record these solutions.

To calculate an upper bound for the Lagrangian dual in (27), we relax constraints (7) from the (QP) model. The resulting capacitated assignment problem can be solved quickly, and a repair heuristic is used to make the solution feasible with respect to the original problem. The objective value of this feasible solution is used as η^* in the updating rule (27) of the subgradient algorithm.

Our LR-based heuristic is described as follows:

LR-based Heuristic:

Step 1. Initialise an empty set of assumptions `A`.
Step 2. Construct the (QP) model, together with assumptions `A`.

  i. If the size of the model is sufficiently small to solve in practically acceptable time, attempt to solve it:
     (a) If a solution with sufficiently small relative IP gap can found within a short time limit, terminate the algorithm returning the optimal solution to the (QP) subject to assumptions `A`.
     (b) If no solution with sufficiently small relative IP gap can be found within a short time limit, continue to Step 3.
     (c) If no feasible solution exists, determine which assumptions are causing the infeasibility, remove them, and continue to Step 3.
  ii. If the size of the model is expected to be too large to solve in practically acceptable time, continue to Step 3.

Step 3. Run `LD-Subroutine` and retrieve all the values of ξ, $\bar{\xi}$, ϕ, $\bar{\phi}$, ψ, and $\bar{\psi}$.
Step 4. For each of the value and counter-value pairs $\{v, \bar{v}\}$ obtained in Step 3, add an assumption to `A` with probability $\frac{v}{v+\bar{v}}^2$ (or $\frac{\bar{v}}{v+\bar{v}}^2$) and then return to Step 2.

LD-Subroutine:

Step 1. Initialise the λ vector to its starting value λ^0, $\epsilon_0 := 2$, determine an estimate for η^*, initialise the iteration counter $k := 0$, and initialise all ξ, $\bar{\xi}$, ϕ, $\bar{\phi}$, ψ, and $\bar{\psi}$ values to zero.
Step 2. Construct the N (QR$_i$) models according to the input data and the set of assumptions `A`, for $1 \leq i \leq N$.
Step 3. Solve the N (QR$_i$) models, and update λ according to (27).
Step 4. If $\epsilon_k \leq \frac{1}{2}$, update the values of ξ, $\bar{\xi}$, ϕ, $\bar{\phi}$, ψ, and $\bar{\psi}$ according to observations about solution X^k.
Step 4. If the η^k has not been improved in the last $10 \times N$ iterations, then $\epsilon_{k+1} := \frac{1}{2}\epsilon_k$.
Step 5. If $\epsilon_k < 0.1$ then terminate `LD-Subroutine` and return the values of ξ, $\bar{\xi}$, ϕ, $\bar{\phi}$, ψ, and $\bar{\psi}$; otherwise set $k := k + 1$ and return to Step 2.

5 Genetic Algorithm Based Matheuristic

Although Genetic Algorithms (GA) are often used in solving the QKP and QMKP [13], [14], [9], these publications on GA are not directly applicable to our problem due to the existence of lower bounds on class sizes. Moreover, the existence of lower and upper bounds on class sizes renders classical operations of crossover and mutation inefficient, i.e. both operations produce too many infeasible solutions. The computational experiments indicated that the common approach of introducing penalty for infeasibility does not improve the performance of the classical version of GA. These observations lead to the development of a matheuristic, presented in this paper, which is an amalgamation of GA and IP. This matheuristic was compared with the LR-based approach described above.

In the developed version of GA, feasible solutions in the initial population are be generated using a two-step procedure. First, we decide which classes will be run, and how many students will be in each class by solving a straightforward

IP. For all test cases, including all cases where CPLEX failed to solve the original QP problem, this IP was solved in under a second. Next, we randomly assign students to classes according to the numbers obtained in the previous step.

The crossover operator, which addresses the challenge imposed by the existence of lower and upper bounds on class sizes, is defined as follows. As with conventional crossover operators, the designed crossover operates on two solutions, referred to as parents. The results of the designed crossover is a single solution, referred to as a child. As with the procedure that generates solutions for the initial population, the crossover operator involves two stages. First, an IP is solved that determines which classes should be run in the child, and the number of students in each class. In this IP, those classes that are run in both parents must run in the child, and those classes that are not run in either parent will not run in the child.

Next, students are assigned to classes in numbers specified by the IP. In this stage, first, students are assigned starting with students who are assigned to the same class in both parents. Each of these students is assigned to the that class in the child as well. The remaining students are assigned one at a time. If at least one class chosen for the student in the parents is available for this student in the child, then the student is assigned to one of these classes. If there are two such classes, the actual class is chosen at random. Students who were not allocated are then randomly allocated to the classes in the child solution according numbers specified by the IP.

6 Computational results

Using Ausgrid's training data as a template, we generated a series of random test cases[3]. Each test case had between 100 and 500 students, with between 4 and 12 student types, and between 56 and 98 classes.

We used IBM ILOG CPLEX 12.5.0.0 64-bit on an Intel i7-4790K quad-core 4.00Ghz system with 16GB of RAM, running Windows 7 Professional. Our code was written in C# 4.0, and interacted with CPLEX using the IBM ILOG Concert API. We used default CPLEX settings, except we increased the maximum allowed memory usage to the total amount of free physical memory. Since the proposed heuristic is probabilistic, we applied it to each of the test cases 10 times. The pseudorandom number generator we used was the MT19937 Mersenne Twister [12]. For the weighted objective function, we used $\alpha = \beta = 1$. We also applied the GA matheuristic to each of the test cases 10 times for the same amount of time that the LR-based heuristic used on average.

Table 1 shows the results of the computational experiments. The tables shows the test case (Case), the number of students (Std), the number of classes (Cls), the number of student types (Typ), the number of variables in the QP (Vars), the minimum (tMin), average (tAvg), and maximum (tMax) solution time for the LR-based heuristic, the minimum (lrMin) and maximum (lrMax) objective

[3] All test cases and solution files are available on request.

value obtain by applying the LR-based heuristic, and the minimum (gaMin) and maximum (gaMax) objective value obtain by applying the GA matheuristic. Since the two approaches tested are quite different, time is reported in CPU time. It is clear from the results that the LR-based heuristic outperformed the GA matheuristic.

In Figure 1, the horizontal axis depicts the 90 test cases. The vertical axis gives the time taken, in seconds. The graph shows the range of solution times across the 10 runs of the LR-based heuristic. The solid line shows the average time across the 10 runs of the heuristic. The largest value was about 40 minutes, however the overall average was only about 272 seconds. In contrast, the attempts to obtain exact solutions by solving the corresponding quadratic programming problem using CPLEX failed in most cases given a six hour limit.

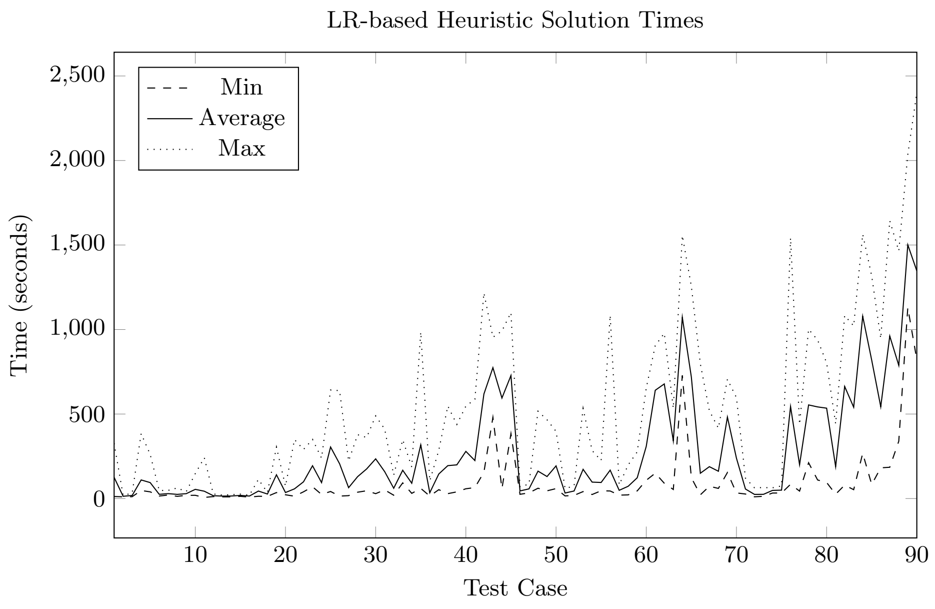

Fig. 1. The time taken for the LR-based heuristic to produce a solution.

7 Conclusions

In this paper we presented a heuristic solution approach, based on the Lagrangian relaxation, for the problem of assigning students to classes. This problem arises in large organisations that require training and retraining of staff. The objective function reflects the preference of assigning certain groups of students to the same class, which often occurs in practice. The proposed heuristic was tested by computational experiments on a number of randomly generated test cases,

Case	Std	Cls	Typ	Vars	tMin	tAvg	tMax	lrMin	lrMax	gaMin	gaMax
1	300	56	3	62776	12.1	125.4	327.1	1700.5	14101.5	7979	13904.5
2	300	66	3	73986	10.3	13.4	18.4	1483.5	3009	20452	26405.5
4	300	88	3	98648	47.5	110.1	383.8	1526.5	6277	12868.5	20827.5
5	300	98	3	109858	39.7	94.4	261.1	1328.5	3012	19947.5	21808.5
7	291	66	4	73392	20.8	29	48.4	1883.5	2405	13872	20788
8	291	78	4	86736	13.1	24.7	62.5	2160	2739.5	16303.5	22527.5
10	291	98	4	108976	19.2	54.4	147.6	1904	6210	14350.5	22959.5
11	206	56	4	57512	6.5	43.4	241.5	1437	17346.5	5930.5	10101
13	206	78	4	80106	9.8	14.6	20.5	1443.5	3482	12167	16141
14	206	88	4	90376	10	15	23	1236	12347	13018.5	16949
16	300	56	6	62776	8.6	14.7	19.4	1597	7193.5	21518.5	31016
17	300	66	6	73986	13.5	44.3	113.4	1574.5	3314	11135.5	21437
19	300	88	6	98648	35.4	140.5	303.9	1341	3251	10727	18855
20	300	98	6	109858	22.3	35.6	73.2	1136.5	3634.5	20977	25647
22	400	66	5	80586	39.9	99	290.6	2181	5061	13182.5	28905
23	400	78	5	95238	69.8	194.1	349.9	2078	2665.5	11723.5	25204
25	400	98	5	119658	41.8	304.8	642.1	1665	7581	11361.5	26601
26	500	56	6	73976	14.4	205.8	638.5	3004.5	16429	16148	38892.5
28	500	78	6	103038	37.6	129.6	377.8	3117	4130.5	28708	43548.5
29	500	88	6	116248	46.1	175.2	361.3	2389.5	7289	23430	45908.5
31	132	56	7	53368	51.2	159.1	411	1454.5	1654.5	4324.5	7100
32	132	66	7	62898	19.7	61.1	138.6	1383	2549.5	5418	6608
34	132	88	7	83864	30.7	90.7	186.4	1518.5	2106	4081	6520
35	132	98	7	93394	60	317.4	985.1	1292	1890	4297.5	6680
37	400	66	5	80586	52.5	145.4	285.9	1889	3319.5	11636.5	24165
38	400	78	5	95238	29	194.7	547.5	2262	3835.5	23962	26256.5
40	400	98	5	119658	57.9	279.1	546.7	1639	5139.5	11082.5	25742
41	500	56	8	73976	64.9	224.9	587	3786.5	11623	14888.5	37871.5
43	500	78	8	103038	480.3	774.3	954.3	3095.5	7006.5	13194	29130.5
44	500	88	8	116248	58.3	594.7	991.5	2574	3868.5	10993.5	32387.5
46	300	56	6	62776	25.1	45.1	69.6	2274.5	18798	9859	21706.5
47	300	66	6	73986	33.6	57.2	85.3	2183.5	8768	10534	19975.5
49	300	88	6	98648	45.7	130.2	466	2308	7697.5	9481.5	19685
50	300	98	6	109858	58.9	193.2	397.4	1778.5	3082	10332	18988
52	168	66	6	65274	19	44.7	74.1	1637	2614	6321.5	8415
53	168	78	6	77142	41.2	172.9	532.7	1605.5	12674	6406	8189
55	168	98	6	96922	46.1	95.1	224.9	1404.5	4050.5	7505.5	9456
56	500	56	5	73976	44.7	168	1084.7	2825	13742	15216.5	40438
58	500	78	5	103038	21.7	72.7	199.1	2550.5	3835	32732.5	44762.5
59	500	88	5	116248	45.3	122.9	276.3	2465.5	13736.5	36664.5	47615.5
61	300	56	7	62776	149.9	640.6	904.9	2286	3305.5	7636.5	12557
62	300	66	7	73986	90.6	678.1	978	2312.5	2786.5	10760.5	14360
64	300	88	7	98648	728	1071.9	1556.5	2215	2666.5	9325	12500.5
65	300	98	7	109858	123.3	719.3	1256.1	1999.5	4367.5	9902.5	16164
67	400	66	8	80586	72.6	188.6	519.3	2105.5	3174.5	16810.5	24442.5
68	400	78	8	95238	60	161.5	416.2	1992.5	3147.5	14011	26611.5
70	400	98	8	119658	33.3	239.1	597.3	1584.5	2056.5	11480.5	27258
71	132	56	7	53368	26.8	56.4	120.8	1237	9558	5303	7409.5
73	132	78	7	74334	13.4	25	65	1230.5	1910.5	6667.5	7684.5
74	132	88	7	83864	32.6	48.2	63.6	1296	1665	5611.5	6721
76	300	56	9	62776	84.4	544.1	1542.1	2189	13773	6696	11942.5
77	300	66	9	73986	44.1	200.3	438.4	1982.5	2979	10571.5	16390
79	300	88	9	98648	109.9	542.2	938	2254	3810.5	7113	15953.5
80	300	98	9	109858	95	534.7	802.9	1958.5	3169.5	8544	12726
82	400	66	11	80586	79.5	663.1	1082.2	2577	6110	10912	16338
83	400	78	11	95238	52.5	540.4	1021.6	2681.5	3984	10771.5	23656.5
84	400	88	11	107448	267.5	1078.4	1563.9	2512.5	5764.5	10662	14386.5
85	400	98	11	119658	83.4	818.2	1316.3	2404.5	5749	10067.5	23096
86	500	56	11	73976	181.8	543.3	946.7	3735	17272.5	16703	17796
87	500	66	11	87186	185.1	961.3	1643.4	3624	8932.5	14413.5	20936
88	500	78	11	103038	339.4	788	1465	3709.5	7593	13059.5	30584.5
89	500	88	11	116248	1130.2	1501	2035.3	3185	16049.5	10998	21648.5
90	500	98	11	129458	821.2	1347.1	2401	2724.5	9361	10298.5	29047

Table 1. The results of the computational experiments for many of the test cases.

based on data supplied by Ausgrid. The proposed heuristic was able to provide solutions to all test cases in a practically acceptable time. In contrast, the straight forward quadratic programming based approach failed in most cases with a time limit of six hours. In both cases, CPLEX was used as the solver. The Lagrangian relaxation-based heuristic was also compared with a specifically designed matheuristic based on Genetic Algorithms. This comparison indicated the superiority of the Lagrangian relaxation-based heuristic. The integer programming components of the matheuristic were also solved with CPLEX.

The proposed Lagrangian relaxation-based heuristic includes a number of parameters, and future research can be focussed on an investigation of the influence of these parameters on the performance of the entire procedure.

References

1. Caprara, A., Pisinger, D., Toth, P.: Exact solution of the quadratic knapsack problem. INFORMS Journal on Computing 11(2), 125–137 (1999)
2. Chen, Y., Hao, J.K.: Iterated responsive threshold search for the quadratic multiple knapsack problem. Ann Oper Res 226, 101–131 (2015)
3. Chopra, S., Rao, M.R.: The partition problem. Mathematical Programming 59(1-3), 87–115 (1993)
4. Dahlhaus, E., Johnson, D.S., Papadimitriou, C.H., Seymour, P.D., Yannakakis, M.: The complexity of multiway cuts. In: Proceedings of the twenty-fourth annual ACM symposium on Theory of computing. pp. 241–251. ACM (1992)
5. Fisher, M.L.: The lagrangian relaxation method for solving integer programming problems. Management science 50(12_supplement), 1861–1871 (2004)
6. García-Martínez, C., Rodriguez, F., Lozano, M.: Tabu-enhanced iterated greedy algorithm: A case study in the quadratic multiple knapsack problem. European Journal of Operational Research 232, 454–463 (2014)
7. Garey, M.R., Johnson, D.S.: Computers and intractability: a guide to np-completeness (1979)
8. Guignard, M.: Lagrangean relaxation. Top 11(2), 151–200 (2003)
9. Hiley, A., Julstrom, B.A.: The quadratic multiple knapsack problem and three heuristic approaches to it. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation. pp. 547–552. ACM (2006)
10. Johnson, E.L., Mehrotra, A., Nemhauser, G.L.: Min-cut clustering. Mathematical programming 62(1-3), 133–151 (1993)
11. Julstrom, B.A.: Greedy, genetic, and greedy genetic algorithms for the quadratic knapsack problem. In: Proceedings of the 7th annual conference on Genetic and evolutionary computation. pp. 607–614. ACM (2005)
12. Matsumoto, M., Nishimura, T.: Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Transactions on Modeling and Computer Simulation (TOMACS) 8(1), 3–30 (1998)
13. Saraç, T., Sipahioglu, A.: A genetic algorithm for the quadratic multiple knapsack problem. In: Advances in Brain, Vision, and Artificial Intelligence, pp. 490–498. Springer (2007)
14. Singh, A., Baghel, A.S.: A new grouping genetic algorithm for the quadratic multiple knapsack problem. In: Evolutionary Computation in Combinatorial Optimization, pp. 210–218. Springer (2007)