Ann Oper Res (2014) 218:147–163
DOI 10.1007/s10479-013-1358-8

The generalized balanced academic curriculum problem with heterogeneous classes

Sara Ceschia · Luca Di Gaspero · Andrea Schaerf

Published online: 5 April 2013

Abstract We propose an extension of the *Generalized Balanced Academic Curriculum Problem* (GBACP), a relevant planning problem arising in many universities. The problem consists of assigning courses to teaching terms and years, satisfying a set of precedence constraints and balancing students' load among terms. Differently from the original GBACP formulation, in our case, the same course can be assigned to different years for different *curricula* (i.e., the predetermined sets of courses from which a student can choose), leading to a more complex solution space.

The problem is tackled by both Integer Programming (IP) methods and combinations of metaheuristics based on local search. The experimental analysis shows that the best results are obtained by means of a two-stage metaheuristic that first computes a solution for the underlying GBACP and then refines it by searching in the extended solution space.

Keywords Timetabling · Simulated Annealing · Academic Planning · Mixed integer quadratic programming

1 Introduction

In many universities, one of the problems that the academic administration faces while designing (or modifying) degree programs is the determination of the *curricula*, i.e., the lists of mandatory/optional courses from which students can build their personal study plan. After the determination of curricula, a planning problem arises, which concerns the assignment of individual courses to teaching periods (i.e., a pair term/academic year) within the studying

S. Ceschia · L. Di Gaspero (✉) · A. Schaerf
DIEGM, University of Udine, via delle Scienze 208, 33100, Udine, Italy
e-mail: luca.digaspero@uniud.it

S. Ceschia
e-mail: sara.ceschia@uniud.it

A. Schaerf
e-mail: schaerf@uniud.it

horizon. In general, the assignment must satisfy a set of precedence constraints between courses (e.g., prerequisites) and it should balance students' load among terms.

This problem is called *Balanced Academic Curriculum Problem* (BACP) and it has been first modeled by Castro and Manzano (2001). In its initial version, the balancing of the students' load was achieved by minimizing the maximum of the course load per student over all terms. This problem formulation has been added as *problem 30* in CSPLib (Gent and Walsh 1999), and three benchmark instances have been made available. The problem has been solved by Castro and Manzano (2001) using constraint programming (CP).

The BAC problem, in the CSPLib formulation, has been also tackled by Hnich et al. (2002) and Castro et al. (2007), using CP and Integer Programming (IP) techniques, and by Lambert et al. (2006), using a hybrid technique composed by genetic algorithms and constraint propagation.

Monette et al. (2007) perform an extensive study on BACP by experimenting with 720 instances of size up to 200 courses and varying characteristics built by means of a random instance generator. One of their contributions consists of criticizing the min-max optimization criterion, based on the original BACP formulation and introducing an optimization criterion based on (linear and quadratic) deviations from the average load, which are more adequate in practice.

An additional point of the CSPLib formulation that seems quite unrealistic is the implicit assumption that a student takes all delivered courses without any personal choice. In order to overcome this limitation, the formulation has been extended in Di Gaspero and Schaerf (2008) by allowing several curricula with shared courses among them. In this formulation, called GBACP (G for Generalized), a *curriculum* is a set of courses that represents a possible complete selection of a student, and courses have to be balanced and limited in number for each single curriculum. Moreover, the new formulation includes professors' preferences for teaching in specific terms, which are often present in real situations.

Di Gaspero and Schaerf (2008) also introduced six new instances, called UD1–UD6, obtained from real data from the University of Udine. The new instances are much larger than the CSPLib ones and they exhibit different structure, since they come from very different cases. Moreover, they turn out to be much harder to solve than the CSPLib ones, and their optimal cost remains unknown. In Di Gaspero and Schaerf (2008) the problem was tackled by local search (LS), which was easily able to find the optimal solution for the CSPLib instances and gave the first upper bounds of the objective values for the new instances.

The GBACP has been further investigated by Chiarandini et al. (2011) who propose two solution techniques: the first one is based on IP, while the second is an improvement of the local search algorithm proposed by Di Gaspero and Schaerf (2008). In addition, four new instances, called UD7–UD10 (again from University of Udine), have been considered.

The IP procedure by Chiarandini et al. (2011) provided many high-quality lower bounds and also the best known solution for one instance (UD6), whereas the LS outperformed both IP and the previous LS of Di Gaspero and Schaerf (2008) for the other instances (except one, UD4, for which all three methods find the same value).

All data on GBACP has been made available on the web at http://satt.diegm.uniud.it/projects/gbac/ together with a program that validates solutions. The same website reports also the best solutions of the present work.

In this work we propose a further extension of the problem, obtained by relaxing one implicit assumption. Namely, in the GBACP formulation it is assumed that a given course should be taken by all students in the same academic year. In principle, however, students can enroll to a given course in different academic years of their study plan or, from a different point of view, a course can be taught in different years for different curricula. Therefore,

a more reasonable situation is to schedule courses in teaching terms regardless of the academic year and to assign each course/curriculum pair to a specific academic year.

Since in this case classes can be composed by heterogeneous students (w.r.t. the academic year in which they are enrolled) we call this extension the *Generalized Balanced Academic Curriculum Problem* with *Heterogeneous Classes* (GBACP-HC). An important observation is that a high heterogeneity of students in a class is not desirable for didactic reasons, therefore we will try to minimize it by adding a cost component to the objective function.

We design two search methods based on IP and LS, respectively. The two methods are analyzed and compared experimentally on the available instances.

The paper is organized as follows. In Sect. 2 we introduce the formulation of GBACP-HC. Section 3 illustrates the IP-model and Sect. 4 explains the LS solvers for the problem. The experimental analysis is described in Sect. 5. Finally, in Sect. 6 we draw conclusions and discuss future work.

2 GBACP-HC formulation

The GBACP-HC problem consists of the following entities:

Courses: let $\mathcal{C}$ be the set of *courses* to be taught within the university program under consideration. Each course $c \in \mathcal{C}$ has a number of *credits* $r_c \in \mathbb{Z}^+$, which are a measure of its workload.

Curricula: a student can select the courses of his/her study plan among a set of alternatives and a complete selection of a student is represented by a *curriculum*. Let $Q \subseteq \mathcal{C}$ denote a curriculum (i.e., the subset of courses composing a study plan), and Ω be the collection of all curricula.

Periods: the planning horizon is divided into *academic years* $\mathcal{Y} = \{0, \ldots, Y-1\}$ and each of them is subdivided into *terms* $\mathcal{T} = \{0, \ldots, T-1\}$. The whole planning horizon is organized in a set $\mathcal{P}$ of consecutive teaching periods, $\mathcal{P} = \{0, \ldots, P-1\}$, with $P = Y \cdot T$. The t-th term of each year is the set of periods $\{t + y \cdot T | y = 0, \ldots, Y-1\}$, whereas the terms of the y-th year is the set of periods $\{t + y \cdot T | t = 0, \ldots, T-1\}$.

For example, a three-year degree organized in four terms per year has $P = 12$ periods, i.e., $\mathcal{P} = \{0, \ldots, 11\}$, and the first terms of each year is the set $\{0, 4, 8\}$, while the terms of the second years are the set $\{4, 5, 6, 7\}$.

In the original GBACP formulation, courses are assigned to periods. For GBACP-HC, instead, each course c is assigned to a term; in addition, each course/curriculum pair (c, Q), with $c \in Q$, is assigned to an academic year. As a result, the course is assigned to different periods in the various curricula, but it is taught once in the year in a single specific term.

In the following, we combine the assignment information of c and (c, Q), and we will consider each pair (c, Q) as an individual course c'_Q, which has to be assigned to a specific period within the time horizon. In this setting, we must impose the constraint that the courses c'_{Q_1}, c'_{Q_2} arising from c and belonging to different curricula Q_1 and Q_2 must be assigned to periods corresponding to the same term but possibly in different years.

The set of generated courses will be denoted by $\mathcal{C}'$ and a generic one is denoted by c'. The cardinality of $\mathcal{C}'$ is $\sum_{Q \in \Omega} |Q|$. Constraints defined on individual courses $c \in \mathcal{C}$ are naturally extended to the generated courses $c'_Q \in \mathcal{C}'$.

As customary, constraints are split into *hard* ones, that must be satisfied in all solutions, and *soft* ones, that can be violated and contribute to the objective function. The GBACP-HC counts the following hard constraints:

H1. Course limits: for each curriculum $Q \in \Omega$, there are a minimum and a maximum number of courses, denoted by $\underline{\mu}$ and $\overline{\mu}$ respectively, that can be assigned to each period $p \in \mathcal{P}$.

H2. Prerequisites: some courses must be planned earlier than others, since the first ones are prerequisites for the second ones. A precedence relation is expressed by means of a directed acyclic graph $G = (\mathcal{C}, E)$, where $\mathcal{C}$ is the set of courses and an edge $(c_1, c_2) \in E$ states that c_1 is a prerequisite of c_2. Therefore, the courses generated from c_1 must be assigned to periods that precede the ones assigned to the courses generated from c_2. We denote by $G' = (\mathcal{C}', E')$ the version of the graph that considers the generated courses.

H3. Period limits: for each course c there are a minimum and a maximum period $\underline{\pi}_c$ and $\overline{\pi}_c$ that can be assigned to it. These values are set beforehand, and they can be tightened by reasoning on the precedence graph.

Regarding the constraint H1, the original formulation by Castro and Manzano (2001) includes limits in terms of both number of courses and total credits. However, the quadratic penalty of credit load balancing makes the presence of hard limits in the number of credits superfluous in our formulation, because large discrepancies are already prevented. Therefore, we decided to remove the constraint on credits. Conversely, the limits on the number of courses are maintained because they help in avoiding an excessive fragmentation of courses, resulting in a large number of final exams (independently of credits), which is undesirable for the students.

The soft constraints are the following ones:

S1. Load balancing: the distribution of credits assigned to each curriculum among the teaching periods must be balanced (ideally all periods of the same curriculum should have the same number of credits). We denote with α_Q the ideal distribution of credit, defined as $\alpha_Q = \sum_{c \in Q} r_c / |\mathcal{P}|$ and we denote the deviation of the actual (integer-valued) load x from ideal load α_Q with $\delta_Q(x)$.
A natural definition of $\delta_Q(x)$ would be $\delta_Q(x) = |x - \alpha_Q|$, however in case of fractional values of α_Q such a function would have a cost greater than zero also in balanced solutions. Therefore, in order to better capture the intuition that perfectly balanced solution have cost zero, we compute the deviations using ceiling and floor operators as:

$$\delta_Q(x) = \begin{cases} x - \lceil \alpha_Q \rceil & \text{if } x \geq \lceil \alpha_Q \rceil \\ \lfloor \alpha_Q \rfloor - x & \text{if } x < \lfloor \alpha_Q \rfloor \end{cases}$$

We thus use a penalty that is quadratic in the $\delta_Q(x)$ variables.

S2. Preferences: professors can express options for their teaching periods. Specifically, a professor can indicate some terms as undesirable for teaching a specific course. We denote with $U \subseteq \mathcal{C} \times \mathcal{P}$ the set of these undesirable assignments. Any assignment of a course c'_Q generated from c and some curriculum Q to a period p with $(c, p) \in U$ determines a preference violation.

S3. Heterogeneity: since the same course can be assigned to different years in different curricula we consider a measure of how much *heterogeneous* are the students attending it. Among the different possibilities, we select the difference between the maximum and the minimum academic year in which the course is taught as the measure of heterogeneity.

A solution of the GBAC-HC problem is an assignment of terms to courses and of academic years to course/curriculum pairs such that all hard constraints (Course limits, Prerequisites and Period limits) are satisfied and the violation of soft constraints (Load Balancing, Preferences, and Heterogeneity) is minimized.

3 IP model for GBACP-HC

In order to provide an Integer Programming formulation of the GBACP-HC we consider the vector $\bar{x}$ of $|\mathcal{C}'| \times |\mathcal{P}|$ binary variables $x_{c',p}$, which are equal to one if the generated course c' is assigned to the teaching period p.

To model the problem the constraints imposed on the $x_{c',p}$ variables are the following.

$$\sum_{y=0}^{Y-1} x_{c'_{Q_1},t+y\cdot T} = \sum_{y=0}^{Y-1} x_{c'_{Q_2},t+y\cdot T} \qquad \forall t \in \mathcal{T};\ Q_1, Q_2 \in \Omega;\ c \in Q_1 \cap Q_2 \tag{1}$$

$$\sum_{p\in\mathcal{P}} x_{c',p} = 1 \qquad \forall c' \in \mathcal{C}' \tag{2}$$

$$\underline{\mu} \le \sum_{c\in Q} x_{c'_Q,p} \le \overline{\mu} \qquad \forall Q \in \Omega;\ p \in \mathcal{P} \tag{3}$$

$$\sum_{s=0}^{p-1} x_{c'_1,s} \ge x_{c'_2,p} \qquad \forall \left(c'_1, c'_2\right) \in E'^*;\ p \in \mathcal{P} \tag{4}$$

$$\sum_{p\in\mathcal{P}} (p+1) x_{c'_2,p} - \sum_{p\in\mathcal{P}} (p+1) x_{c'_1,p} \ge 1 \qquad \forall \left(c'_1, c'_2\right) \in E'^* \tag{5}$$

$$x_{c',p} = 0 \qquad \forall c' \in \mathcal{C}';\ p < \underline{\pi}_{c'} \vee p > \overline{\pi}_{c'} \tag{6}$$

$$x_{c',p} \in \{0,1\} \qquad \forall c' \in \mathcal{C}';\ p \in \mathcal{P} \tag{7}$$

Constraints (1) bind the variables representing the same course in different curricula to be assigned to the same term (while allowing them to be assigned to different years). Constraints (2) require that each generated course is assigned to one and only one period, while constraints (3) set the limits on the number of courses assigned to each period in every curriculum. The precedence between courses is enforced by constraints (4)–(5). Notice that in constraint (5) the periods are shifted by 1 (i.e., we multiply the $x_{c',p}$ variables by $p+1$ instead of p) in both summations to avoid a degenerate constraint for period $p = 0$. It is possible to see that one of the two sets of constraints is redundant, however we added it to the model to obtain a further pruning of the search tree. Moreover, notice that in order to restrict the search space we consider also implied constraints that are obtained by the transitive closure $G'^* = (\mathcal{C}', E'^*)$ of the precedence relation. Finally, constraints (6) take care of assignments outside the period limits.

The objective function to be minimized is a weighted sum of the three soft components representing Load balancing, Preferences, and Heterogeneity: $F(\bar{x}) = w_1 \cdot f_1(\bar{x}) + w_2 \cdot f_2(\bar{x}) + w_3 \cdot f_3(\bar{x})$. The component f_2 is defined directly on the $\bar{x}$ variables, whereas components f_1 and f_3 make use of auxiliary variables $\bar{v}$ and $\bar{z}$, whose definitions will be provided later. The three components are defined as follows:

$$f_1(\bar{x}) = \sum_{Q\in\Omega} \sum_{p\in\mathcal{P}} \left(z^+_{Q,p} + z^-_{Q,p}\right)^2 \tag{8}$$

$$f_2(\bar{x}) = \sum_{(c,p)\in U} x_{c,p} \tag{9}$$

$$f_3(\bar{x}) = \sum_{c\in\mathcal{C}} (\overline{v}_c - \underline{v}_c) \tag{10}$$

Namely, component (8) computes the squared deviation from the ideal workload, component (9) accounts for preference violations and component (10) measures the heterogeneity of classes.

Notice that component (8) is quadratic. This is not an issue, because CPLEX directly handles quadratic models (MIQP) within its solver. Obviously, removing the quadratic terms results in a solver speed-up, however we are interested in finding a solution to the original problem model and not of a relaxation of it. An analysis of the linearization of the quadratic term appears in the work by Chiarandini et al. (2011) on the GBACP formulation, and there is no significant difference in behavior for the linear model w.r.t. to the quadratic one.

According to the original GBACP formulation, the weights of the first two objective components are set as $w_1 = 1$, $w_2 = 5$.

For the heterogeneity component, given that its importance can be subjective, we experiment with two different settings. In the first one, we set $w_3 = 1$ so that its level of importance is the same as the balancing component. In the second setting, instead, we considerably lower the impact of heterogeneity and we set $w_3 = 1/5$. For $w_3 = 1/5$ in order to exploit the faster integer arithmetics, all weights have been multiplied by 5 and divided back only in the final output.

Moving to the auxiliary variables, the $z^+_{Q,p}$ and $z^-_{Q,p}$ are integer variables that measure the positive and negative deviations from ideal workload, i.e. the two subcases in the definition of $\delta_Q(\cdot)$. They are defined as follows:

$$z^+_{Q,p} = \max\left\{0, \sum_{c\in Q} r_c \cdot x_{c'_Q,p} - \lceil \alpha_Q \rceil\right\} \qquad \forall Q \in \Omega;\ p \in \mathcal{P} \tag{11}$$

$$z^-_{Q,p} = \max\left\{0, \lfloor \alpha_Q \rfloor - \sum_{c\in Q} r_c \cdot x_{c'_Q,p}\right\} \qquad \forall Q \in \Omega;\ p \in \mathcal{P} \tag{12}$$

Moreover, $\upsilon_{c,y}$ are indicator variables that have value 1 if there exists any curriculum Q in which course c is taught in year y, while $\underline{\upsilon}_c$ and $\overline{\upsilon}_c$ are integer variables whose value is the minimum and maximum year in which course c is taught.

$$\upsilon_{c,y} = \max\{x_{c'_Q,y\cdot T+t} | \exists Q, c \in Q; t = 0,\ldots,T-1\} \qquad \forall c \in \mathcal{C};\ y \in \mathcal{Y} \tag{13}$$

$$\underline{\upsilon}_c = \min\big\{(y+1)\cdot \upsilon_{c,y} - 1 \big| y = 0,\ldots,Y-1\big\} \qquad \forall c \in \mathcal{C} \tag{14}$$

$$\overline{\upsilon}_c = \max\big\{(y+1)\cdot \upsilon_{c,y} - 1 \big| y = 0,\ldots,Y-1\big\} \qquad \forall c \in \mathcal{C} \tag{15}$$

To summarize, the full model comprises $|\mathcal{C}'| \times |\mathcal{P}| + |\mathcal{C}| \times |\mathcal{Y}|$ binary variables[1] ($x_{c',p}$, and $\upsilon_{c,y}$) and $2 \times |\Omega| \times |\mathcal{P}| + 2 \times |\mathcal{C}|$ integer variables ($z^+_{Q,p}$, $z^-_{Q,p}$, $\underline{\upsilon}_c$, and $\overline{\upsilon}_c$). Moreover, there are $(|\mathcal{T}| \times |\Omega| \times O(|\mathcal{C}|)) + (|\mathcal{C}'|) + (|\mathcal{C}'| \times O(|\mathcal{P}|)) + (2 \times |\Omega| \times |\mathcal{P}|) + (|\mathcal{C}| \times |Y|) + (2 \times |\mathcal{C}|)$ equality constraints (constraints (1), (2), (6), (11)–(12), (13), (14)–(15)) and $(|\Omega| \times |\mathcal{P}|) + (O(|\mathcal{C}'|^2) \times |\mathcal{P}|) + (O(|\mathcal{C}'|^2))$ inequality constraints (constraints (3), (4), (5)).

4 Local search for GBACP-HC

We describe our local search technique in three stages by specifying the different components of our solution method at increasing levels of abstraction. Namely we present first the basic local search components, then the metaheuristic strategy and, finally, the metaheuristic instantiations and the higher level control strategy for combining the basic metaheuristics.

[1] In this sum implicit binary variables, such those for modeling the min/max operators, are not included.

4.1 Basic local search entities

The basic building blocks for specifying a local search algorithm are the *search space*, the *neighborhood relation*, and the *cost function*. For GBACP-HC we employ different versions of these entities, each of them dealing with a different aspect of the problem.

4.1.1 Search spaces

In the local search procedure for GBACP (Chiarandini et al. 2011), the search space consists of all assignments of (not generated) courses to periods, i.e., pairs term/year. It is easy to see that this search space is also a (restricted) search space for GBACP-HC, in which the same year is assigned for all generated course variables of any curricula. We denote this search space by $\mathcal{S}$.

The full search space for the GBACP-HC problem is composed of all the possible assignments of a term and a year for each course/curriculum pair or, in other words, for each of the generated courses $c' \in \mathcal{C}'$. This search space will be denoted by $\mathcal{S}'$. Obviously, the GBACP search space is a subset of the complete GBACP-HC one, i.e.:

$$\mathcal{S} \subseteq \mathcal{S}'$$

In both search spaces, we do not allow solutions that violate the period limits $\underline{\pi}_c$, $\overline{\pi}_c$ (H3). All other assignments are included in the search space, even if they violate course limits (H1) and prerequisites (H2). Given that the two latter constraints are hard constraints, their violations constitutes the so-called *distance to feasibility* measure. Indeed, as done also in McCollum et al. (2010), the quality of the solution is evaluated with a hierarchical function composed by two components: the *distance to feasibility*, which measures the violations of the hard constraints, and the *objective function*, which rates the satisfaction of the soft constraints. It is worth remarking that in our case all final solutions have distance to feasibility equal to zero, so that this *softening* process is only part of the search method.

The initial solution for LS is generated in a totally random way but fulfilling the H3 constraint: Each course is assigned to a random term and to a random year uniformly chosen from its assignable range.

4.1.2 Neighborhood relations

The period change and period swap moves employed for GBACP (denoted by $\mathcal{N}_P$) can be employed for GBACP-HC on the restricted search space $\mathcal{S}$. However, the peculiarities of the problem led us to the definition of two more specific neighborhood relations, which allow a finer control on the problem features and work on the complete search space $\mathcal{S}'$.

The first neighborhood, denoted by $\mathcal{N}_T$ is composed by moves that either change the *term* of one course or swap the *term* of two courses. The other neighborhood, denoted by $\mathcal{N}_Y$ is composed by the moves that either change the *year* of one pair course/curriculum or swap the *year* of two courses in a given curriculum.

Notice that all these neighborhoods are specifically designed for their specific search space. For example, it is not meaningful to use the $\mathcal{N}_P$ neighborhood relation on the $\mathcal{S}'$ space, since it is applicable only for those states that are $\mathcal{S}$ solutions. Similarly, the $\mathcal{N}_T$ and $\mathcal{N}_Y$ neighborhoods could not be applied to the search space $\mathcal{S}$. Conversely, it is possible to apply singularly the $\mathcal{N}_T$ or $\mathcal{N}_Y$ on the $\mathcal{S}'$ space, even though the search space is not connected under just one of them.

```
procedure SimulatedAnnealing(Search Space S, Neighborhood 𝒩, Cost Function F,
                             Initial Temperature T_0, Minimum Temperature T_min,
                             Cooling Rate α, Neighbors Sampled N);
begin
  i := 0;
  s_0 := InitialSolution(S);
  T := T_0; s_best := s_0;
  while (T > T_min) do
  begin
    n := 0;
    while (n < N) do
      m := DrawRandomMove(s_i, 𝒩);
      ΔF := F(s ⊙ m) − F(s);
      if (ΔF ≤ 0 or U(0, 1) < e^{−ΔF/T})
      then s_{i+1} := s_i ⊙ m;
        if (F(s_{i+1}) < F(s_best))
        then s_best := s_{i+1};
      else s_{i+1} := s_i;
      i := i + 1; n := n + 1;
    end;
    T := α · T;
  end;
  return s_best;
end;
```

Fig. 1 The Simulated Annealing local search procedure

4.1.3 Cost function

The cost function is the weighted sum of the three components introduced in Eqs. (8)–(10) plus the distance to feasibility of the hard constraints allowed in the search space, weighted by a penalty $W = 1000$.

4.2 Simulated Annealing metaheuristic

Once the basic components have been selected, we have to specify the strategy for guiding LS. In this work we select the Simulated Annealing metaheuristic, which has exhibited a good behavior in the GBACP case.

Simulated Annealing was proposed by Kirkpatrick et al. (1983) and Černý (1985) and extensively studied by Aarts and Korst (1989) and van Laarhoven and Aarts (1987) among other researchers. The method got that name after an analogy with a simulated controlled cooling of a collection of hot vibrating atoms. The idea is based on accepting non-improving moves with probability that decreases with time.

We outline the procedure as it has been implemented in our solver. The pseudocode of the algorithm is provided in Fig. 1.

The process starts by creating a random initial state s_0. The main procedure consists of a loop that randomly generates at each iteration a neighbor of the current solution. Given a move m, we denote with ΔF the difference in the cost function between the new solution and the current one, i.e., $\Delta F = F(s \odot m) - F(s)$. If $\Delta F \leq 0$ the new solution is accepted

and becomes the current one. If the value of cost function of the current solution is lower than the one of the best solution so far found, the best solution is updated to the current one. Conversely, if $\Delta F > 0$ the new solution is accepted with probability $e^{-\Delta F/T}$, where T is a parameter, called the *temperature*.

The temperature T is initially set to an appropriately high value T_0. After a fixed number of iterations N, the temperature is decreased by the *cooling rate* α, so that at each cooling step n, $T_n = \alpha \times T_{n-1}$. In general α is in the range $0.8 < \alpha < 1$.

The procedure stops when the temperature reaches a *low-temperature level* T_{min}, that is when no solution that increases the cost function is accepted anymore. In this case we say that the system is *frozen*.

4.3 Metaheuristic instantiations

Given the basic components outlined in the previous sections, we have different choices for instantiating a family of metaheuristic solvers. The most straightforward possibility is to consider the full-fledged Simulated Annealing solver equipped with the $\mathcal{S}'$ search space and the neighborhood composed by the set union of $\mathcal{N}_T$ and $\mathcal{N}_Y$ moves. This solver deals with the complete search space representation under a neighborhood that allows a complete connection. We denote this solver with $\mathsf{SA}(\mathcal{S}', \mathcal{N}_T \oplus \mathcal{N}_Y)$.

On the basis of our experience on similar problems, we consider also two other solvers obtained by a sequential composition of a solver on the GBACP search space, denoted by $\mathsf{SA}(\mathcal{S}, \mathcal{N}_P)$, with a solver for GBACP-HC working on $\mathcal{S}'$. The second component can be either the above one or another partial solver on the complete search space $\mathcal{S}'$, which deals with the neighborhood $\mathcal{N}_Y$ only.

The sequential combination of two solvers, denoted in the following by the symbol $\triangleright$, employs the final solution found by the first solver as the initial solution of the second one.

To summarize, we experiment and compare among themselves three Simulated Annealing solvers: $\mathsf{SA}(\mathcal{S}', \mathcal{N}_T \oplus \mathcal{N}_Y)$, $\mathsf{SA}(\mathcal{S}, \mathcal{N}_P) \triangleright \mathsf{SA}(\mathcal{S}', \mathcal{N}_T \oplus \mathcal{N}_Y)$, and $\mathsf{SA}(\mathcal{S}, \mathcal{N}_P) \triangleright \mathsf{SA}(\mathcal{S}', \mathcal{N}_Y)$.

5 Experimental analysis

5.1 Benchmark instances

We test our algorithms on ten instances that have been extracted from the database containing historical data at the School of Engineering of University of Udine. All instances are available from the web at http://satt.diegm.uniud.it/projects/gbac/, together with a format description, our best solutions and the C++ source code of the validator that certifies their scores. Table 1 summarizes the main features of the instances in terms of number of periods, number of courses, number of curricula, average number of courses per curriculum, average number of courses per curriculum per period, number of prerequisite constraints, number of preference constraints.

5.2 Implementations and general settings

The Integer Programming model described in Sect. 3 have been implemented in C++ using IBM Ilog CPLEX 12.2. The local search algorithms have been implemented in C++ language, exploiting the EASYLOCAL++ framework (Di Gaspero and Schaerf 2003).

All experiments have been performed on a 2.66 GHz quad-core PC with 4 GB RAM, running Ubuntu Linux x86_64 (rel. 10.04). The local search software has been compiled

Table 1 The Benchmark Instances used in the experimentation

Instance	Periods (Years × Terms)	Courses	Curricula	Courses per Curriculum	Courses per Curriculum per Period	Prereq.	Pref.
UD1	9 (3 × 3)	307	37	34.62	3.847	1383	270
UD2	6 (2 × 3)	268	20	27.8	4.633	174	158
UD3	9 (3 × 3)	236	31	29.81	3.312	1092	198
UD4	6 (2 × 3)	139	16	25.69	4.281	188	80
UD5	6 (3 × 2)	282	31	34.32	5.72	397	162
UD6	4 (2 × 2)	264	20	27.15	6.787	70	110
UD7	9 (3 × 3)	302	37	33.89	3.766	1550	249
UD8	6 (2 × 3)	208	19	22.58	3.763	149	120
UD9	9 (3 × 3)	303	37	34.08	3.787	1541	255
UD10	6 (2 × 3)	188	15	25.07	4.178	214	110

using the GNU C++ compiler (v. 4.1.2). For the purpose of making the comparison easier, neither the IP solver nor the LS one took advantage of multi-threading capabilities of the testing platform.

All experiments have been replicated for the two settings of the weights of the heterogeneity component f_3, namely $w_3 = 1$ and $w_3 = 1/5$. As already said, in order to keep all the values integer, for $w_3 = 1/5$ we multiplied the weights by 5 and then divided back the cost of the final solution.

5.3 Integer programming

The main purpose of the experiments with the IP solver was to make an exploratory analysis of the benefits gained by allowing heterogeneous classes and to set some upper bounds on the solution values. Since also for the GBACP case the running time of the IP solver was set to be an order of magnitude higher than those of LS, we decided to allow it a run time limit of 7200 seconds. The IP solver is deterministic, thus it was invoked once for each instance. We report its results for the two weight settings in Table 2 along with the reference values of the best known GBACP solution. The best results from the comparison of the three solvers are highlighted in boldface in the table. Unfortunately, the results were very poor and the IP solver was not able to find any feasible solution in 5 instances within the allowed running time (denoted by a dash in the table) and it was able to improve the cost value on just one instance.

To overcome the problem of the combinatorial growth of the search tree due to the explosion of variables of the model, we decided to test also a restricted IP model which binds the term assignment variables to the values of the best known solution for the GBAC problem at hand, as taken from the GBACP website. Namely, in the restricted model the following constraints are added to the IP formulation:

$$x_{c',p} = 0 \quad \forall c' \in \mathcal{C}';\ p \in \left\{ t + y \cdot T \;\middle|\; t = 0, \ldots, T-1 \wedge t \neq \mathsf{term}(c),\ y = 0, \ldots, Y-1 \right\} \tag{16}$$

where $\mathsf{term}(c)$ is the term assigned to course c in the corresponding best known solution of GBACP. Consequently, in this model the only degree of freedom is to change the year assigned to each generated course.

Table 2 Solutions of the CPLEX solver for the IP full model (time limit 7200 s)

Instances	IP				IP				Best GBACP		
	$w_1 = 1, w_2 = 5, w_3 = 1$				$w_1 = 1, w_2 = 5, w_3 = 1/5$						
	f_1	f_2	f_3	F	f_1	f_2	f_3	F	f_1	f_2	F
UD1	–	–	–	–	–	–	–	–	248	2	**258**
UD2	228	3	7	250	314	2	49	333.8	146	0	**146**
UD3	–	–	–	–	–	–	–	–	160	0	**160**
UD4	324	0	11	**335**	396	4	24	420.8	396	0	396
UD5	–	–	–	–	–	–	–	–	201	2	**211**
UD6	81	2	25	116	52	2	23	66.6	52	0	**52**
UD7	–	–	–	–	–	–	–	–	184	1	**189**
UD8	113	1	14	132	243	4	33	269.6	42	0	**42**
UD9	–	–	–	–	–	–	–	–	192	2	**202**
UD10	107	1	14	126	328	2	18	341.6	44	0	**44**

Table 3 Solutions of the CPLEX solver for the IP restricted model (time limit 3600 s)

Instances	IP_Y				IP_Y				Best GBACP		
	$w_1 = 1, w_2 = 5, w_3 = 1$				$w_1 = 1, w_2 = 5, w_3 = 1/5$						
	f_1	f_2	f_3	F	f_1	f_2	f_3	F	f_1	f_2	F
UD1	263	2	13	286	294	2	18	307.6	248	2	**258**
UD2	135	0	2	137	135	0	4	**135.8**	146	0	146
UD3	176	0	8	184	196	0	18	199.6	160	0	**160**
UD4	–	–	–	–	396	0	1	396.2	396	0	396
UD5	–	–	–	–	291	2	127	326.4	201	2	**211**
UD6	45	0	2	47	45	0	2	**45.4**	52	0	52
UD7	192	1	1	198	227	1	26	237.2	184	1	**189**
UD8	35	0	2	37	34	0	4	**34.8**	42	0	42
UD9	190	2	0	**200**	194	2	15	207.0	192	2	202
UD10	44	0	0	**44**	46	0	5	47.0	44	0	**44**

The results of the CPLEX solver on this model within a time limit of 3600 seconds are reported in Table 3. Differently from the previous case, the restricted IP solver was able to improve the solution cost in many cases for both the weight settings. This result justifies the introduction of the GBACP-HC model.

Even though we were able to improve the cost of the best known solutions for GBACP, the performances of the IP solver are quite far from those of LS, which will be detailed in the next section.

5.4 Local search

The experiments on the three Simulated Annealing solvers were conducted using the following methodology: First a preliminary tuning phase has been performed with the purpose of identifying the best settings of the SA parameters along all the benchmark instances, and

then we compare our tuned solvers with previous results of Chiarandini et al. (2011) on the GBACP.

The SA procedure shown in Fig. 1 has four parameters: start temperature T_0, stop temperature T_{min}, cooling rate α, and number of neighbors sampled at each temperature N. In order to give to all parameter configurations the same amount of computational time, we let the three parameters T_0, T_{min}, and α vary and we compute N in such a way to have exactly the same number of total iterations. In detail, calling I the fixed total number of iterations, we compute N from the following formula

$$N = -I / \log_{\alpha}(T_0 / T_{min})$$

The total number of iterations is set to $I = 3 \cdot 10^8$ which results in a running time of about 360 seconds on the computational architecture already described, thus the settings is similar to the one imposed by Chiarandini et al. (2011).

For the composed solvers the running time is split between the two components. In detail, the first solver is granted 60 seconds and the second one 300 seconds. This splitting is given according to preliminary experiments, that show that 60 seconds are enough to get to a good solution for the GBAC problem.

Instead of directly fixing a value for the final temperature we have decided to specify a range of variation ρ for the temperatures, so that the final temperature T_{min} will be equal to T_0/ρ.

Preliminary experiments show that α is not significant. This is not surprising, because in our setting N is computed from the other parameters, and therefore α only determines the entity of the single step in the temperature and not the actual slope of the cooling trajectory, which is determined by ρ. We therefore set α to the fixed value 0.9999.

For the remaining two parameters (T_0 and ρ), we employed a F-Race procedure (Birattari et al. 2002) for selecting the best configuration among a set of candidate ones. The configurations have been designed by means of *Nearly Orthogonal Latin Hypercubes* (*NOLH*) (Cioppa and Lucas 2007), which allow us to cover the design space using a limited number of configurations. To generate the actual configurations we use the NOLH spreadsheet made available by Sanchez (2005), using the design with 17 points. In detail, we set the intervals of the parameters to the following values (based on preliminary runs): $T_0 \in [10^1, 10^3]$ and $\rho \in [10^2, 10^4]$.

As for the F-Race we use the canonical value of 0.05 as the significance level in the tests. The transformation of results in ranks prescribed by the procedure guarantees that in the statistical test procedure the aggregation of results over the instances is not influenced by the differences in the values of the cost function, that depend on the instance.

In Table 4 we report, for each Simulated Annealing solver, the settings of the parameters that achieved the best results on both the combinations of weights.

We then analyze and compare the solvers against the results of Chiarandini et al. (2011): we run each solver for 50 times on each instance and for each weight settings, and we record the values of the objective function components at the end of the run.

For space reasons we show the results of this experiment on only two instances, namely UD5 and UD6, in Figs. 2 and 3. These instances exhibit two typical behaviors of the solver, which are also confirmed on the remaining instances.

The results are reported in form of box-and-whiskers plots presenting the distributions of the objective components values obtained by each solver on the two weight settings (identified by the different values of w_3).

From the pictures we can see that in both cases the solvers are in general able to achieve lower values of the load balancing component (f_1) w.r.t. the best solver for the original

Table 4 Settings of the Simulated Annealing parameters for the three local search solvers

Weights	Solver	T_0	ρ
$w_1 = 1, w_2 = 5, w_3 = 1$	$\mathsf{SA}(\mathcal{S}', \mathcal{N}_T \oplus \mathcal{N}_Y)$	31.62	134.89
	$\mathsf{SA}(\mathcal{S}, \mathcal{N}_P) \triangleright \mathsf{SA}(\mathcal{S}', \mathcal{N}_T \oplus \mathcal{N}_Y)$	13.48	316.22
	$\mathsf{SA}(\mathcal{S}, \mathcal{N}_P) \triangleright \mathsf{SA}(\mathcal{S}', \mathcal{N}_Y)$	10	2398.8
$w_1 = 1, w_2 = 5, w_3 = 1/5$	$\mathsf{SA}(\mathcal{S}', \mathcal{N}_T \oplus \mathcal{N}_Y)$	316.22	7585.77
	$\mathsf{SA}(\mathcal{S}, \mathcal{N}_P) \triangleright \mathsf{SA}(\mathcal{S}', \mathcal{N}_T \oplus \mathcal{N}_Y)$	13.48	316.22
	$\mathsf{SA}(\mathcal{S}, \mathcal{N}_P) \triangleright \mathsf{SA}(\mathcal{S}', \mathcal{N}_Y)$	10	2398.8

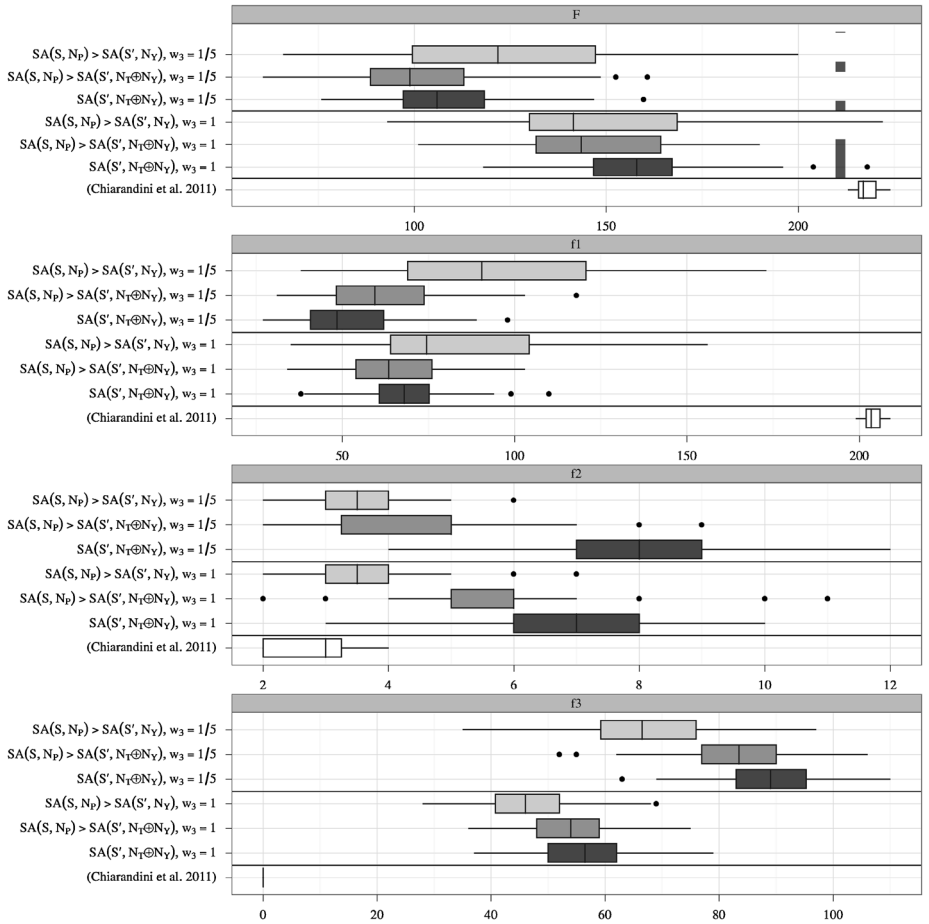

Fig. 2 Distributions of the objective function components on 50 runs: instance UD5

GBACP formulation. This is more evident in the case of the instance UD5 (Fig. 2), in which all the solvers on both weight settings are able to effectively trade load balancing for heterogeneity while breaking only a small amount of preferences.

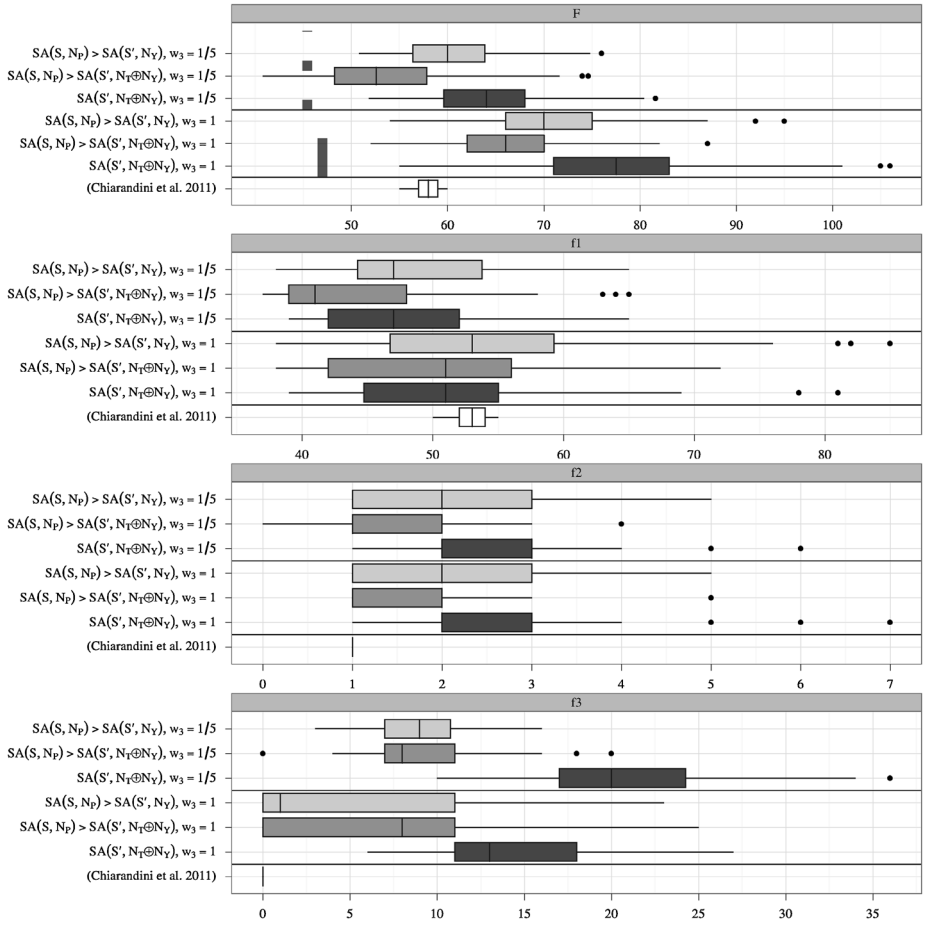

Fig. 3 Distributions of the objective function components on 50 runs: instance UD6

As for the UD6 instance, instead, this behavior is limited to the setting $w_3 = 1/5$ and the trade-off among the three components is less pronounced in favor of obtaining better total cost values.

Looking at those pictures, it is possible to see that, in general, the solver that achieved the best performance on the whole benchmark set is $\mathsf{SA}(\mathcal{S}, \mathcal{N}_P) \triangleright \mathsf{SA}(\mathcal{S}', \mathcal{N}_T \oplus \mathcal{N}_Y)$. This is confirmed by the comparison among the solvers we have performed using a Wilcoxon statistical test (see, e.g., Conover 1999). The results of this test for the two weights settings are presented in Table 5, in which we report the mean rank obtained by each solver in a pairwise comparison of solvers, along with the statistical significance of the rank difference.

The solver $\mathsf{SA}(\mathcal{S}, \mathcal{N}_P) \triangleright \mathsf{SA}(\mathcal{S}', \mathcal{N}_T \oplus \mathcal{N}_Y)$ clearly outperforms the other two solvers on the weight setting $w_1 = 1$, $w_2 = 5$, $w_3 = 1/5$, whereas it shows equivalent results w.r.t. $\mathsf{SA}(\mathcal{S}, \mathcal{N}_P) \triangleright \mathsf{SA}(\mathcal{S}', \mathcal{N}_Y)$ on the first set of weights.

Finally, Tables 6(a) and 6(b) present a conclusive summary of the results of our best solver and a comparison w.r.t. the best results for GBACP. In particular, Table 6(a) shows the median values of the results and the comparison with the best solver of Chiarandini et al.

Table 5 Results of the Wilcoxon test for pairwise comparison of the Simulated Annealing solvers. The entries in the table are the average rank of each solver in the statistical test and the difference is marked as significant at the following probability values: * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Weights	Solver$_1$/Solver$_2$	$\mathsf{SA}(\mathcal{S}', \mathcal{N}_T \oplus \mathcal{N}_Y)$	$\mathsf{SA}(\mathcal{S}, \mathcal{N}_P) \triangleright \mathsf{SA}(\mathcal{S}', \mathcal{N}_Y)$	$\mathsf{SA}(\mathcal{S}, \mathcal{N}_P) \triangleright \mathsf{SA}(\mathcal{S}', \mathcal{N}_T \oplus \mathcal{N}_Y)$
$w_1 = 1, w_2 = 5, w_3 = 1$				
	$\mathsf{SA}(\mathcal{S}', \mathcal{N}_T \oplus \mathcal{N}_Y)$	–	141.3/108.6***	144.1/105.9***
	$\mathsf{SA}(\mathcal{S}, \mathcal{N}_P) \triangleright \mathsf{SA}(\mathcal{S}', \mathcal{N}_Y)$	108.6/141.3***	–	126.4/123.5
	$\mathsf{SA}(\mathcal{S}, \mathcal{N}_P) \triangleright \mathsf{SA}(\mathcal{S}', \mathcal{N}_T \oplus \mathcal{N}_Y)$	105.9/144.1***	123.5/126.4	–
$w_1 = 1, w_2 = 5, w_3 = 1/5$				
	$\mathsf{SA}(\mathcal{S}', \mathcal{N}_T \oplus \mathcal{N}_Y)$	–	136.32/113.6*	144.4/105.5***
	$\mathsf{SA}(\mathcal{S}, \mathcal{N}_P) \triangleright \mathsf{SA}(\mathcal{S}', \mathcal{N}_Y)$	113.6/136.32*	–	134.3/115.6*
	$\mathsf{SA}(\mathcal{S}, \mathcal{N}_P) \triangleright \mathsf{SA}(\mathcal{S}', \mathcal{N}_T \oplus \mathcal{N}_Y)$	105.5/144.4***	115.6/134.3*	–

Table 6 Summary of the results of the $\mathsf{SA}(\mathcal{S}, \mathcal{N}_P) \triangleright \mathsf{SA}(\mathcal{S}', \mathcal{N}_T \oplus \mathcal{N}_Y)$ solver

(a) Median values

Instances	$w_1 = 1, w_2 = 5, w_3 = 1$				$w_1 = 1, w_2 = 5, w_3 = 1/5$				Chiarandini et al. (2011)		
	f_1	f_2	f_3	F	f_1	f_2	f_3	F	f_1	f_2	F
UD1	259	35	0	296	225	35	6.6	**267.0**	248	23	271.5
UD2	153	5	0	160	139	5	2.2	**148.2**	149	0	149
UD3	164	10	0	176	160	10	2.0	175.6	162	5	**164**
UD4	324	5	5	345	324	5	0.2	**335.0**	396	0	396
UD5	64	30	54	144	60	25	16.8	**60.6**	204	15	217
UD6	51	10	8	66	41	5	1.6	**52.8**	53	5	58
UD7	210	40	0	252	182	40	6.4	229.4	195	30	**223**
UD8	45	5	0	55	34	5	2.2	**44.4**	47	0	46.5
UD9	202	40	0	245	179	35	6.6	222.4	193	25	**218**
UD10	45	5	5	57	42	5	1.8	**49.2**	49	0	49.5

(b) Best values

Instances	$w_1 = 1, w_2 = 5, w_3 = 1$				$w_1 = 1, w_2 = 5, w_3 = 1/5$				Best GBACP		
	f_1	f_2	f_3	F	f_1	f_2	f_3	F	f_1	f_2	F
UD1	213	30	19	262	215	20.0	7.0	**242.0**	248	2	258
UD2	149	0	0	149	134	0	2.2	**136.2**	146	0	146
UD3	162	5	0	167	154	5.0	3.2	162.2	160	0	**160**
UD4	324	0	1	325	324	0	0.2	**324.2**	396	0	396
UD5	42	20	39	101	34	10.0	16.6	**60.6**	201	2	211
UD6	38	5.0	9	52	38	0	2.8	**40.8**	52	0	52
UD7	177	30	14	221	161	20.0	7.6	**188.6**	184	1	189
UD8	33	0	8	41	32	0	2.0	**34.0**	42	0	42
UD9	196	20	0	216	160	15.0	7.8	**182.8**	192	2	202
UD10	39	0	5	44	38	0	1.4	**39.4**	44	0	44

(2011), while Table 6(b) reports the best values and the comparison with the best known solutions as reported in the GBACP website.

The results show that, under the weight setting $w_3 = 1/5$, our best solver was able to found a better solution for the GBAC-HC problem than those found for the GBACP problem on 9 out of 10 instances (best results are highlighted in boldface in the table). Moreover, the median results were also lower than those of Chiarandini et al. (2011) on 6 out of 10 instances.

6 Conclusions and future work

In this work, we have extended GBACP, a recently proposed timetabling problem, to a more complex version, called GBACP-HC, which better captures the real-world problems that the academic administration have to face. Due to continuous changes in national regulations and university evaluation criteria, this problem appears rather often to our university.

We have also shown that, for the available instances, the larger search space of GBACP-HC is indeed useful to obtain solutions with a better load balancing w.r.t. the restricted space of the GBAC problem.

These improvements are sharper for the case in which the weight given to the heterogeneity cost component is rather low ($w_3 = 1/5$). However, this is actually the most reasonable setting, being the heterogeneity of the class intuitively less important than a possible poor balancing of the students' load.

The experiments also show that the best solutions are obtained by a sequential combination of solvers, that first explore the restricted space and then move to the complete one. They also show that the available instances are currently out of the reach of the IP-based methods, at least using straightforward modeling and default settings. Obviously, more elaborated exact methods, such as for example branch-and-cut, could find the optimal solution in the future, as already done for other timetabling problems (e.g., Burke et al. 2012).

For the future, we plan to contact the administration of other universities with the aim to further extend and refine the model so as to capture as many as possible additional practical problems. In addition, we hope to retrieve and publish new, possibly more challenging, instances that could help in the research process.

In addition, we hope to apply the solver for the design of curricula of our university in the future revision. To this regard, we foresee two obstacles, which however are not related to the problem model or to the solver but rather to the organization of the general process itself. First, the administration at present tends to solve the assignment problem contextually along with the definition of the curricula themselves, thus making the application of the solver impracticable. Second, the current situation is more complex in terms of mandatory and optional courses than the data of UD1–UD10 (which is from some years ago). In detail, all the possible choices of the students are not made explicitly available, instead the students can select in a rather free way. This makes it very difficult to elicit the curricula so as to obtain reliable input data for the solver.

Finally, we plan also to implement and compare new local search solvers, that could use different metaheuristics (e.g., tabu search) and novel combinations of them.

Acknowledgements The authors thank Marco Chiarandini for many fruitful discussions about the GBAC and GBAC-HC problems.

The access to IBM Ilog CPLEX 12.2 has been possible through the IBM Academic Initiative.

References

Aarts, E. H. L., & Korst, J. (1989). *Simulated annealing and Boltzmann machines*. New York: Wiley.

Birattari, M., Stützle, T., Paquete, L., & Varrentrapp, K. (2002). A racing algorithm for configuring metaheuristics. In W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, & N. Jonoska (Eds.), *GECCO 2002: proceedings of the genetic and evolutionary computation conference* (pp. 11–18). New York: Kaufmann.

Burke, E. K., Mareček, J., Parkes, A. J., & Rudová, H. (2012). A branch-and-cut procedure for the Udine course timetabling problem. *Annals of Operations Research*, *194*, 71–87.

Castro, C., & Manzano, S. (2001). Variable and value ordering when solving balanced academic curriculum problems. In *6th workshop of the ERCIM working group on constraints*.

Castro, C., Crawford, B., & Monfroy, E. (2007). A quantitative approach for the design of academic curricula. In *Lecture notes in computer science: Vol. 4558. Human interface and the management of information. Interacting in information environments* (pp. 279–288). Berlin: Springer.

Černý, V. (1985). Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Applications*, *45*(1), 41–51.

Chiarandini, M., Di Gaspero, L., Gualandi, S., & Schaerf, A. (2011). The balanced academic curriculum problem revisited. *Journal of Heuristics* (30 pp.). doi:10.1007/s10732-011-9158-2.

Cioppa, T. M., & Lucas, T. W. (2007). Efficient nearly orthogonal and space-filling Latin hypercubes. *Technometrics*, *49*(1), 45–55.

Conover, W. (1999). *Practical nonparametric statistics* (3rd ed.). New York: Wiley.

Di Gaspero, L., & Schaerf, A. (2003). EasyLocal++: an object-oriented framework for flexible design of local search algorithms. *Software, Practice & Experience*, *33*(8), 733–765.

Di Gaspero, L., & Schaerf, A. (2008). Hybrid local search techniques for the generalized balanced academic curriculum problem. In M. Blesa Aguilera, C. Blum, C. Cotta, A. Fernández Leiva, J. Gallardo Ruiz, A. Roli, & M. Sampels (Eds.), *Lecture notes in computer science: Vol. 5296. 5th int. workshop on hybrid metaheuristics (HM-2008)* (pp. 146–157). Berlin: Springer.

Gent, I. P., & Walsh, T. (1999). *CSPLib: a benchmark library for constraints* (Technical report). APES-09-1999. Available from http://csplib.cs.strath.ac.uk/. A shorter version appears in *Lecture notes in computer science: Vol. 1713. Proceedings of the 5th international conference on principles and practices of constraint programming (CP-99)* (pp. 480–481). Berlin: Springer.

Hnich, B., Kızıltan, Z., & Walsh, T. (2002). Modelling a balanced academic curriculum problem. In N. Jussien & F. Laburthe (Eds.), *Proceedings of the fourth international workshop on integration of AI and OR techniques in constraint programming for combinatorial optimisation problems (CP-AI-OR'02)* (pp. 121–131).

Kirkpatrick, S., Gelatt, C. D. Jr., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*, 671–680.

Lambert, T., Castro, C., Monfroy, E., & Saubion, F. (2006). Solving the balanced academic curriculum problem with an hybridization of genetic algorithm and constraint propagation. In *Lecture notes in computer science: Vol. 4029. Artificial intelligence and soft computing—ICAISC 2006* (pp. 410–419). Berlin: Springer.

McCollum, B., Schaerf, A., Paechter, B., McMullan, P., Lewis, R., Parkes, A. J., Di Gaspero, L., Qu, R., & Burke, E. K. (2010). Setting the research agenda in automated timetabling: the second international timetabling competition. *INFORMS Journal on Computing*, *22*(1), 120–130.

Monette, J., Schaus, P., Zampelli, S., Deville, Y., & Dupont, P. (2007). A CP approach to the balanced academic curriculum problem. In B. Benhamou, B. Choueiry, & B. Hnich (Eds.), *Symcon'07, the seventh international workshop on symmetry and constraint satisfaction problems*.

Sanchez, S. M. (2005). NOLH designs spreadsheet. http://diana.cs.nps.navy.mil/SeedLab/. Visited on May 13, 2011. Last updated on April 7, 2006.

van Laarhoven, P. J. M., & Aarts, E. H. L. (1987). *Simulated annealing: theory and applications*. Norwell: Reidel/Kluwer.