Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 793671, 8 pages
http://dx.doi.org/10.1155/2013/793671

*Research Article*

# Solving the Balanced Academic Curriculum Problem Using the ACO Metaheuristic

**José-Miguel Rubio,[1,2] Wenceslao Palma,[1] Nibaldo Rodriguez,[1] Ricardo Soto,[1,3] Broderick Crawford,[1,4] Fernando Paredes,[5] and Guillermo Cabrera[1,6]**

[1] *Escuela de Ingeniería Informática, Pontificia Universidad Católica de Valparaíso, 2362807 Valparaíso, Chile*
[2] *Departamento de Computación e Informática, Universidad de Playa Ancha, 2360002 Valparaíso, Chile*
[3] *Universidad Autónoma de Chile, 7500138 Santiago, Chile*
[4] *Universidad Finis Terrae, 7501015 Santiago, Chile*
[5] *Escuela de Ingeniería Industrial, Universidad Diego Portales, 8370109 Santiago, Chile*
[6] *Department of Engineering Science, University of Auckland, Auckland 1020, New Zealand*

Correspondence should be addressed to José-Miguel Rubio; jose.rubio.l@ucv.cl

Received 4 May 2013; Revised 3 November 2013; Accepted 4 November 2013

Academic Editor: Yudong Zhang

The balanced academic curriculum problem consists in the assignation of courses to academic periods satisfying all the load limits and prerequisite constraints. In this paper, we present the design of a solution to the balanced academic curriculum problem based on the ACO metaheuristic, in particular via the Best-Worst Ant System. We provide an experimental evaluation which illustrates the effectiveness of the proposed approach on a set of classic benchmarks as well as on real instances.

## 1. Introduction

The balanced academic curriculum problem (BACP) consists in assigning courses to teaching periods satisfying prerequisites and balancing students' load in terms of credits and number of courses [1]. The BACP planning horizon is divided into academic years, and each academic year is divided into terms. Each term of a year is a teaching period in which courses can take place. The problem consists in finding an assignment of courses to periods that satisfies certain load limits and prerequisites.

The BACP is strongly NP-complete as shown in [1]. Moreover, BACP is an interesting problem because it is at the boundary of several classes of problems such as bin packing, scheduling, and balancing [2]. For instance, prerequisite constraints make BACP look like a scheduling problem. A unique resource (the student) is balanced considering unit-time (periods) activities (courses) and temporal constraints (prerequisites).

The original formulation of BACP was proposed in [3] and was included as a benchmark, called prob030, in CSPLib. The BACP has been tackled using constraint programming (CP) [2], integer linear programming (ILP) [4, 5], hybrid techniques based on genetic algorithms and constraint propagation [6, 7], integer programming (IP), and hybrid local search methods [1]. In [2], a CP approach is proposed where an ad hoc branching heuristic is used in order to guide the search towards balanced solutions improving the first-fail heuristic. In [4], ILP and CP are integrated in order to bound and guide the search and to reduce the search space, respectively. Moreover, the experimental evaluation shows that this integration decreases the run-time on many instances. Approaches based on ILP and CP techniques are proposed separately in [5] obtaining successful results on real instances. The benefits of hybridization have been shown in [6, 7] where a hybrid framework including genetic algorithms and constraint propagation is proposed and tested successfully. In [1], IP can reach good results in some cases using the basic formulation combined with a problem decomposition and relaxation. In the other cases, local search solvers are designed using the concept of generalized local Search Machines where different machines that combine Hill

Climbing, Simulated Annealing, and Dynamic Tabu Search are used in order to improve the results returned by IP.

In this paper we solve the BACP using the ant colony optimization (ACO) metaheuristic [8] model called Best-Worst Ant System (BWAS) [9, 10]. To the best of our knowledge this work is the first one tackling the BACP via the BWAS. BWAS integrates concepts from the evolutionary computing field allowing a good balance between exploration and exploitation of the search space provided by a mutation of the pheromone matrix and a pheromone updating mechanism, respectively. Additionally, a restart process avoids the algorithm getting stuck and performing unnecessary iterations.

Recent work shows that the balancing criterion is interesting not only for university timetabling but also for other scheduling applications, such as the allocation of workload to employees or, in that specific example, to nurses. We hope that the models and the techniques discussed in this work may contribute to further research in this context.

The rest of this paper is organized as follows. In Section 2, we introduce the BACP. In Section 3, we describe the design of our solution. In Section 4, we provide an experimental evaluation of our solution on benchmark and real instances. Section 5 concludes the paper and gives some direction for further work.

## 2. The BACP Formulation

In this section we present the BACP formulation given in [1]. The curriculum consists of entities which must satisfy academic and administrative constraints.

(i) *Courses*. An academic curriculum is denoted by a set of courses and a set of prerequisites related to these. Let $C$ be the set of courses to be taught during the planning horizon of a university degree. Each course $c \in C$ gives a number of credits $\text{cr} \in Z^+$.

(ii) *Periods*. The planning horizon is divided into academic years, and each academic year is divided into terms. Each term is a teaching period in which courses can take place. Let $P$ be the set of teaching periods, uniquely identified by the corresponding year and term. For example, a three-year degree organized in four terms per year has 12 periods, that is, $P = \{1 \cdots 12\}$, and the first terms of each year are $\{1, 5, 9\}$.

(iii) *Load Limits*. A minimum and a maximum number of courses, denoted by $m$ and $M$, respectively, can be assigned to each term.

(iv) *Prerequisites*. Based on their content, some courses have prerequisites, that is, a set of courses that the students must attend earlier. Prerequisites are formalized by a precedence graph, that is, a directed acyclic graph $D = (V, A)$. Each vertex $i \in V$ represents a course, and each arc $(i, j) \in A$ represents a precedence relation, stating that the course $i$ is a prerequisite of course $j$. If course $i$ is a prerequisite of course $j$, then it has to be assigned to a teaching period that strictly precedes the one assigned to course $j$.

(v) *Equal Distribution of Load*. The distribution of credits among the teaching periods must be balanced. Ideally, each term should have the same number of credits.

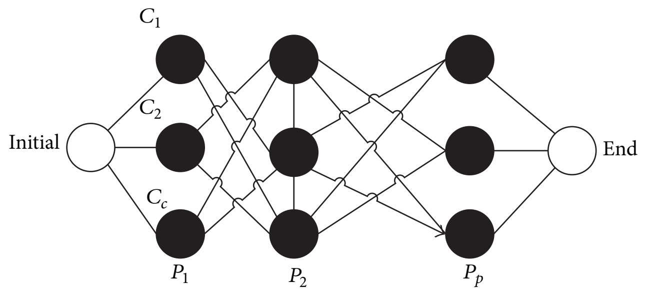

Figure 1: Construction graph.

## 3. Tackling the BACP Using the ACO Metaheuristic

In this section, we describe our approach to tackle the BACP using an ACO model called Best-Worst Ant System (BWAS) [10]. BWAS has three main components.

(i) *A Best-Worst Pheromone Trail Update Rule*. This rule reinforces the edges contained in good solutions and penalizes all the edges contained in the current worst solution.

(ii) *Pheromone Trail Mutation*. The pheromone trails are mutated, with a certain probability, in order to introduce diversity in the search process. Pheromone trails are mutated by adding or subtracting the same amount of pheromone.

(iii) *Restart of the Search Process*. When pheromone trails related to the edges belonging to the best solutions are very high and the remaining ones are close to zero (stagnation phase), the pheromone matrix is set to the initial pheromone value.

In the rest of this section, we describe the graph that represents the assignment of courses to periods and we detail the algorithm of our proposal based on the description of the aforementioned main issues.

*3.1. Construction Graph*. The graph that represents the assignment of the courses to determined periods follows a particular behaviour compared with a TSP. In this case, the graph must have $|C| \times |P|$ nodes, where $C$ is the total amount of courses and $P$ is the total amount of periods.

Figure 1 shows the graph used to represent the assignment of courses to teaching periods. For each course the ant must choose a period from the initial node to the end node. Thus, horizontally, the ant can pass in one node only. This is because one course is assigned to one period, but a period can be assigned to many courses. However, the representation of the graph in the implementation is given by a vector

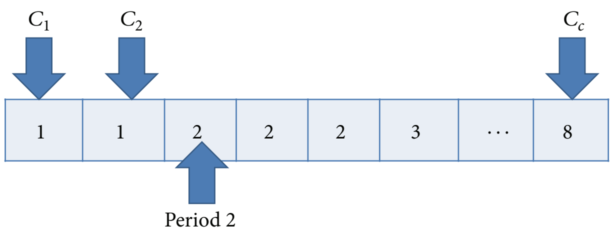

FIGURE 2: Representation of the graph using a vector.

representation as we show in Figure 2. The length of the vector is equal to the number of courses of the curriculum and each position represents the value of the period assigned to that course. Following the example of Figure 2, period 1 is assigned to course 1, period 1 is assigned to course 2, and so on until the course $C$.

To determine the period assigned to each course, a transition rule, which guides the ants in the construction of their solutions, is used. It has two elements: the pheromone trace and the heuristic information. The transition rule is the following:

$$P(c, p_i) = \frac{[\tau_{(c,p_i)}]^{\alpha}[\eta_{(c,p_i)}]^{\beta}}{\sum_{u\in P}\left([\tau_{(c,p_u)}]^{\alpha}[\eta_{(c,p_u)}]^{\beta}\right)}. \quad (1)$$

*3.2. The BWAS Algorithm.* In this section we give the details of the proposed algorithm to tackle the BACP using BWAS. Thus, we focus on how to represent courses and periods, how to update/mutate pheromone trails, and how to restart the search process when it gets stuck. These are the main issues to solve the BACP using BWAS. The details are in Algorithm 1.

The ants deposit pheromone traces when they build a solution to help guide the rest of the colony. In our approach, the pheromone matrix stores the load of pheromone in a certain node instead of showing the quantity of pheromones associated to an edge between two nodes. Let $T[1\cdots C, 1\cdots P]$ be the pheromone matrix, where $\tau_{ij}$ $(1 \le i \le C,\ 1 \le j \le P)$ is an element of $T$ representing the pheromone trace deposited by an ant when the course $i$ is assigned to period $j$. To initialize the pheromone matrix, an initial solution, which only satisfies the prerequisite constraint, is calculated. Once this initial solution is obtained, we calculate the pheromone trail using the quality $Q\ (S_k)$ of the obtained solution. The quality of a solution is calculated with respect to maximum load assigned to the academic periods using the following expression:

$$Q(\mathrm{Sol}) = \frac{1}{\mathrm{Max}\{c_1, c_2, \ldots, c_p\}}. \quad (2)$$

One of the main differences between natural and artificial ants is that the artificial ants use the pheromone trails and also information named heuristic information, denoted by $\eta(c, p_i)$, which measures the preference of moving from one node to another [1]. In static problems this value does not change but in the BACP it is required to be changed in order to give more information to the transition rule. Thus, we defined a function that allows evaluating the amount of violations generated when an assignment is made:

$$\eta(c, p_i) = \frac{1}{1 + v_d(c, p_i) - v_a(c, p_i)}, \quad (3)$$

where $v_d(c, p_i)$ determines the number of violations generated when the course $c$ is assigned to period $p_i$ and $v_a(c, p_i)$ determines the number of violations before this assignment. Thus, this equation can determine how many conflicts are generated because of the assignment. If there are new violations, the heuristic information and the probability of choosing that period become smaller. This expression has a singularity when $v_d = 1$ and $v_a = 2$. This situation shows that the number of violated constraints decreases and the node should be more preferable. Thus, we set $v_a = 1$.

There are two ways to update the pheromone trails in the pheromone matrix [10].

(i) *Deposition*. It depends on the quality of the generated solution. This update process (see (4)) is performed by all the ants. Additionally the ant which generates the best solution performs this action in order to reinforce the pheromone trail:

$$\tau_{ij} \longleftarrow \tau_{ij} + f(Q(S_k)). \quad (4)$$

(ii) *Evaporation*. After the construction of the solution, evaporation (see (5)) is performed in all the nodes

```
(1)  $\tau_o \leftarrow$ InitialSolution($I$)
(2)  initializePheromoneTrails($\tau_o$)
(3)  while (term criterion not satisfied) do
(4)    for $i = 1$ to $k$ do
(5)      for $j = 1$ to numCourses do
(6)        period $\leftarrow$ transitionRule($j$)
(7)        assignCourse($j$, period)
(8)      end for
(9)      deposition($Q(S_i)$)
(10)   end for
(11)   evaporation()
(12)   currentBest $\leftarrow$ selectBestSolutionIt($A_k$)
(13)   localSearch(currentBest)
(14)   if (best(currentBest, globalBest)) then
(15)     globalBest $\leftarrow$ currentBest
(16)   end if
(17)   depositBestAnt(globalBest)
(18)   currentWorst $\leftarrow$ selectWorstSolution($A_k$)
(19)   evaporateWorstAnt(currentWorst)
(20)   mutation()
(21)   if (stagnationcondition) then
(22)     reinitializePheromoneTrails()
(23)   end if
(24) end while
```

ALGORITHM 1: Bwas_bacp (instance $I$: a BACP instance).

of the pheromone matrix using the evaporation constant $\rho$. Additionally, in order to penalize, the nodes belonging to the worst solution are evaporated:

$$\tau_{ij} \longleftarrow \tau_{ij} \cdot (1 - \rho) . \qquad (5)$$

The uniqueness of the BWAS is that which includes three daemon actions, being the Best-Worst update rule responsible for the two first ones. (1) The best ant is selected and an extra deposit is made according to the quality of the solution. (2) The worst ant is selected and evaporation is made on the nodes that have been traversed by this ant; note that evaporation is made only if the best ant has not traversed those nodes. (3) The third action is the mutation, which brings diversity to the search space, adding or subtracting pheromone trace to certain nodes:

$$\operatorname{mut}(\mathrm{it}, \tau_{\mathrm{threshold}}) = \frac{\mathrm{it} - \mathrm{it}_r}{\mathrm{Nit} - \mathrm{it}_r} \cdot \sigma \cdot \tau_{\mathrm{threshold}}, \qquad (6)$$

where it is the actual iteration, $\mathrm{it}_r$ is the last iteration when a reset due to stagnation was performed, Nit is the total number of iterations, and $\sigma$ is the power of mutation [10]. The threshold is calculated using the pheromone deposited in the best solution divided by the number of courses. With this mutation operator, we can add or subtract pheromone trace to the pheromone matrix. Finally, in the case that the search stagnates, a reset of the pheromone trails is performed in order to go back to the last global optimum; thus the search starts from a good solution. In this algorithm we perform a reset if there is no change after 20% of the total number of iterations. In addition, a local search is performed using the best ant of the iteration. This subprocess consists in selecting all the courses of the period with a bigger academic load. Then, we select the period with the smaller academic load and we try to reassign one course of the bigger period to the smaller period. To choose this course, we first evaluate that the course does not generate new violations. Finally, we evaluate this new solution and the last one and we keep the better one as actual best.

## 4. Experimental Evaluation

The proposed algorithm was implemented using Java running on a Windows PC with a 2.40 GHz Intel Core 2 Duo T8300 processor. To test our approach, we use CSPLib benchmark instances and real instances. To obtain the parameters (see Table 1) used in the experiments, the proposed BWAS algorithm was run 30 times on the benchmark and real instances and takes the values belonging to the best solutions reported.

*4.1. Benchmark Instances.* We analysed three benchmark instances where the performance of the algorithm was measured with good results with respect to speed of convergence, quality of the solution, and constraint satisfaction. For each instance of the problem, we performed 50 executions of the algorithm achieving the following results. First, we analysed the small instance of 8 periods (bacp8). This instance has 46 courses to be distributed in 8 academic periods. In Figure 3 we can see the results for bacp8 measured as a percentage considering 50 runs where 74% were able to find the optimal value. In a smaller percentage (4%) of cases, the case was an invalid solution. In most cases, our approach was able to find the optimal value.

TABLE 1: Algorithm parameters.

| Parameter | Value |
|---|---|
| Iterations | 1000 |
| Ants | 8 |
| Evaporation coefficient | 0.2 |
| $\alpha$ | 1 |
| $\beta$ | 2 |
| Mutation probability | 0.3 |
| Mutation power | 0.5 |

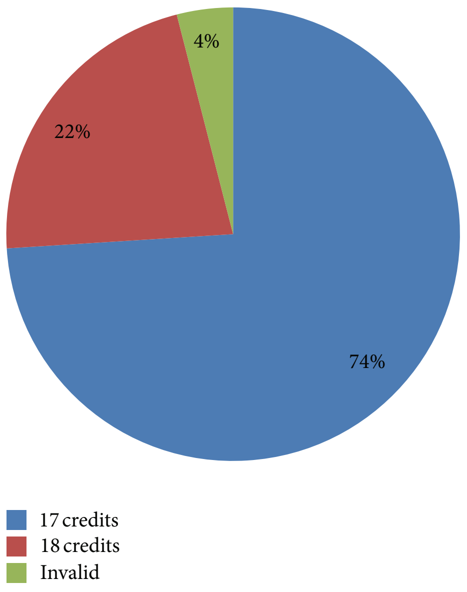

FIGURE 3: Number of credits for a small instance.

The medium instance (bacp10) has 42 courses to be assigned to 10 periods, so the maximum load is smaller than bacp8. This means that the quality is better. After 50 runs the results are given in Figure 4. Again, the optimal solution was found in the majority of the cases.

The last instance is bacp12; in this case, where the complexity is greater, it has 66 courses to be assigned in 12 academic periods. In Figure 5, we can see that the algorithm found four different values and between 18 and 19 credits.

Of the 50 runs, we took the top 10 (not taking into account the invalid ones) and we summarize the results for the three instances shown in each column: name of the instance, best result, worst result, average, standard deviation, and optimal value known for the problem [6].

Also, to compare the times that the algorithm manages to find every quality of solution for each instance, we considered the best and worst solutions. Table 2 shows the difference between bacp12 and the other two instances, where the

Table 2: Result for benchmark instances.

| Instance | BWAS Best | BWAS Worst | BWAS Average | $\sigma$ | lp-solve Best | lp-solve Worst | Optimal |
|---|---|---|---|---|---|---|---|
| Bacp8 | 17 | 18 | 17.3 | 0.48 | 17 | 54 | 17 |
| Bacp10 | 14 | 15 | 14.2 | 0.42 | 24 | 48 | 14 |
| Bacp12 | 18 | 20 | 18.4 | 0.69 | $\infty$ | $\infty$ | 18 |

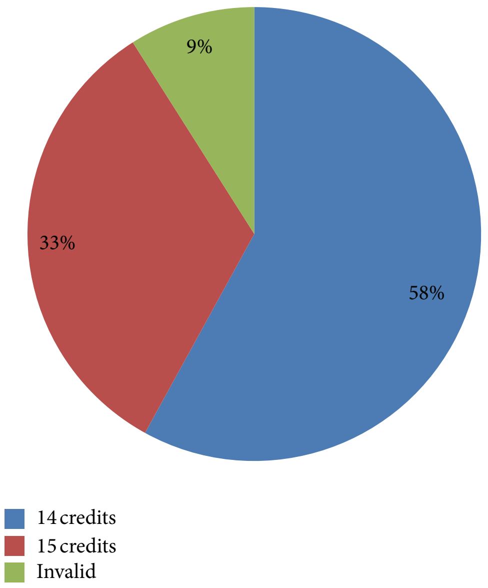

Figure 4: Number of credits for a medium instance.

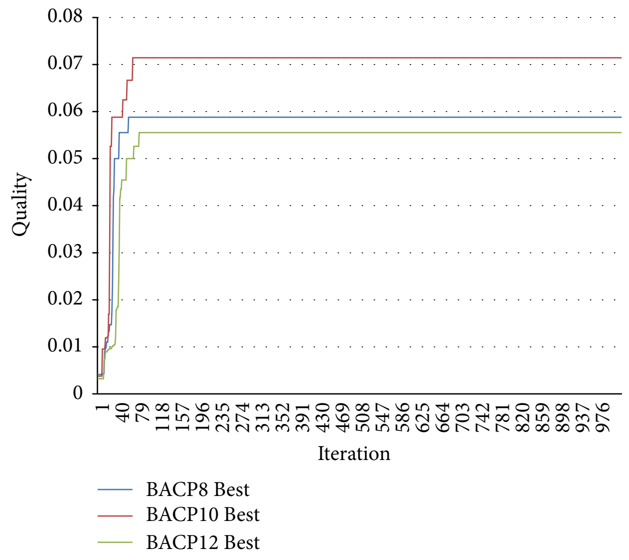

Figure 6: Efficiency of the algorithm (best solution).

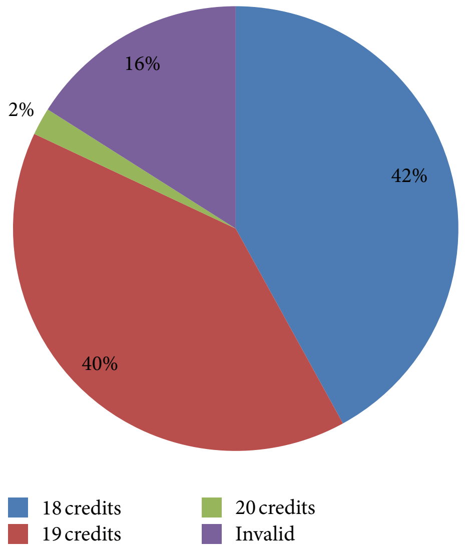

Figure 5: Number of credits for a large instance.

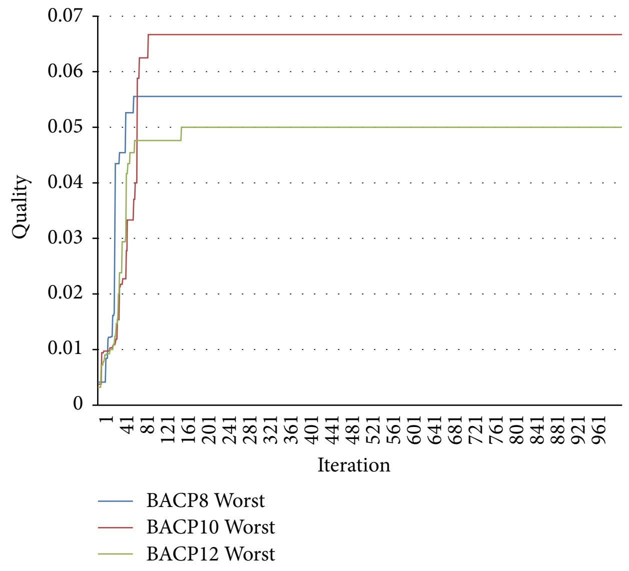

Figure 7: Efficiency of the algorithm (worst solution).

convergence time to the optimal solution is smaller in bacp8 and bacp10. But we cannot guarantee that 18 credits is the optimal solution in bacp12, but it is the best solution found by the algorithm. For the 12-period problem, lp-solve [3] does not get any log after a "turn-around-time" of 1 day.

In Figure 6, we show a graphic with the quality of the best solution found for each instance by iteration. The efficiency of the algorithm is very high, as it converges before the first 100 iterations. In this graphic the quality is measured using (3). In the case of the worst solutions, no instances achieve the optimum, but they all converge quickly. The exception was bacp12 which takes time and moves away from the best value found, even more than the other two instances (see Figure 7).

Table 3: Time (ms) for the best solution to obtain the optimum.

| Credits | Bacp8 | Bacp10 | Bacp12 | Q (sol) |
|---|---|---|---|---|
| 20 | 811 | — | 3994 | 0.05 |
| 19 | — | — | 5438 | 0.052 |
| 18 | 1045 | 1061 | 6368 | 0.055 |
| 17 | 1248 | 1342 | | 0.058 |
| 16 | | 1404 | | 0.062 |
| 15 | | 1451 | | 0.066 |
| 14 | | 1919 | | 0.071 |

Table 4: Time (ms) for the worst solution to obtain the optimum.

| Credits | Bacp8 | Bacp10 | Bacp12 | Q (sol) |
|---|---|---|---|---|
| 20 | 1201 | 1502 | 23448 | 0.05 |
| 19 | 1279 | 1532 | | 0.052 |
| 18 | 1840 | — | | 0.055 |
| 17 | | — | | 0.058 |
| 16 | | 1602 | | 0.062 |
| 15 | | 2222 | | 0.066 |

In Tables 3 and 4 we can see in the first column the maximum amount of credits of the curricula designed by the BWAS algorithm. The next three columns are the instances and the last column is the quality of the solution calculated using (3). The hyphen means that the quality of the solution was skipped. For example, in Table 4, the bacp10 instance went from a quality of 19 to a quality of 16. The empty spaces mean that the solution did not reach that value. The performance of lp-solve is very limited in contrast to the BWAS algorithm. In the case of the 8-period problem, the optimum plan is obtained in 1460 seconds. For the 10-period problem, no result is obtained after 5 hours, and the last result was logged before 1700 seconds (see additional details in [3]).

In Figure 7, we show the worst solution found by the BWAS algorithm for solving the BACP measured by quality.

*4.2. Real Instances.* To test our proposal with real instances, three benchmark files were created using the same format used in the test instances. The curriculums evaluated are Informatics Engineering, short program (8 periods), Informatics Engineering, long program (12 periods), from Catholic University of Valparaiso, and Informatics Engineering, medium program (10 periods), from Playa Ancha University. The academic plan of Informatics Engineering (short program, INF) has 34 courses; the complexity is lower than that of bacp8 because of the number of courses. Table 5 summarizes the results obtained (the optimal value is the maximum of the real curriculum for the career); for example, from 50 executions, the algorithm found a better solution than the real curricula (20 credits) with a maximum amount of 18 credits (see Figure 8).

Also, we evaluate the performance of the BWAS in the curriculum of Informatics Engineering (long program); in this instance the complexity is bigger than that of INF but lower than that of bacp12. It has 53 courses to be assigned to 12 academic periods. The results are shown in Figure 9.

Table 5: Result for real instances.

| Instance | Best | Worst | Average | $\sigma$ | Optimal |
|---|---|---|---|---|---|
| INF | 18 | 18 | 18 | 0 | 20 |
| ICI | 18 | 18 | 18 | 0 | 18 |
| UPLA | 13 | 15 | 14 | 0.66 | 14 |

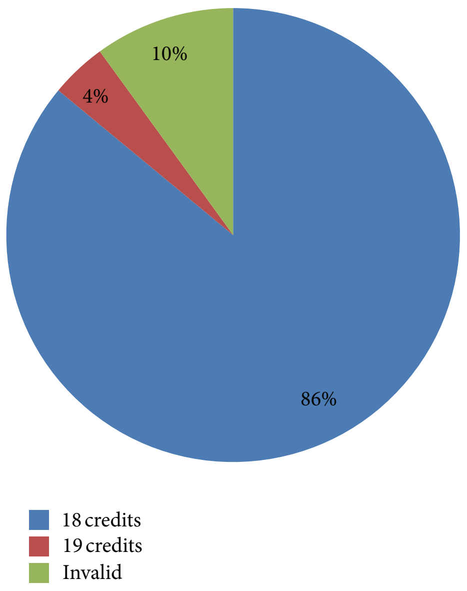

Figure 8: Number of credits for short real instance.

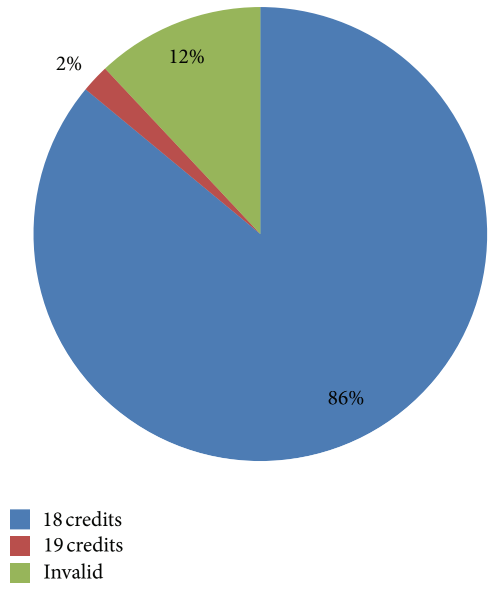

Figure 9: Number of credits for long real instance.

In this instance, the effectiveness was maintained with 86% of the runs, but the number of invalid solutions increases to 12%. The last evaluation corresponds to the Playa Ancha University instance (UPLA) where the optimal solution was found in only 20% of the iterations, surpassing the 14 credits of the official curriculum. However, the objective value was 14

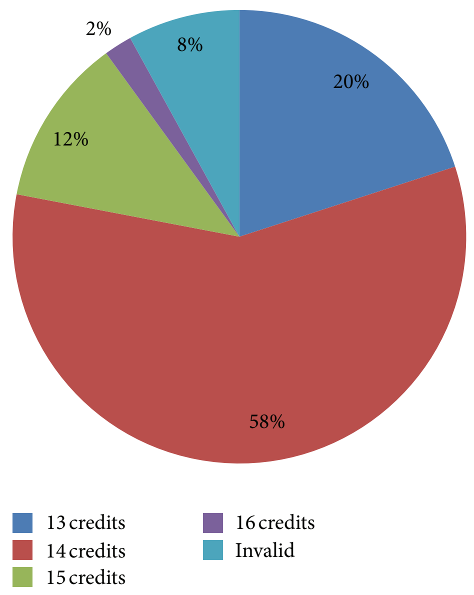

Figure 10: Number of credits for medium real instance.

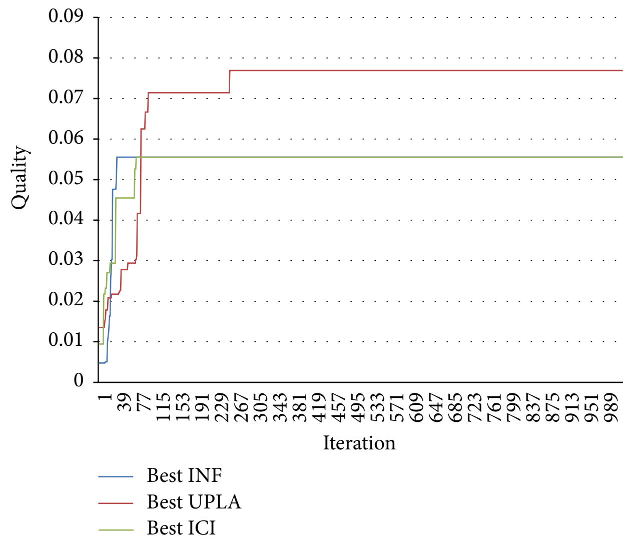

Figure 11: Efficiency of the algorithm in real instances (best solution).

Table 6: Time (ms) for the best solution to obtain the optimum in real instances.

| Credits | INF | ICI | UPLA | Q (sol) |
|---|---|---|---|---|
| 20 | 594 | 3484 | — | 0.05 |
| 19 | — | 3692 | — | 0.052 |
| 18 | 976 | 4080 | — | 0.055 |
| 17 | | | — | 0.058 |
| 16 | | | 2993 | 0.062 |
| 15 | | | 3078 | 0.066 |
| 14 | | | 3392 | 0.071 |

Table 7: Time (ms) for the worst solution to obtain the optimum in real instances.

| Credits | INF | ICI | UPLA | Q (sol) |
|---|---|---|---|---|
| 20 | 3143 | 3969 | — | 0.05 |
| 19 | 3527 | 16671 | — | 0.052 |
| 18 | 8928 | 17274 | — | 0.055 |
| 17 | | | 3574 | 0.058 |
| 16 | | | 3682 | 0.062 |
| 15 | | | 3864 | 0.066 |
| 20 | 3143 | 3969 | — | 0.05 |

credits with 58% of the runs. This curriculum has 49 courses assigned in 10 semesters (see Figure 10). Again, we took the top 10 (regardless of the invalid solutions) from the 50 runs and summarized the results for the three instances.

Now we are going to compare the time and iteration in which the algorithm finds the best and worst solutions. In Table 6 we can see the time in milliseconds that the algorithm takes to reach the different solutions. In the case of INF and ICI, the time is very low. In the case of UPLA, it takes a little more than 10 seconds to reach the optimal value.

Figure 11 shows the quality of solution by iteration where we can see the speed of convergence.

In the case of the worst solutions (see Table 7), INF and ICI reach the optimum, but this time takes more seconds. In the case of UPLA, this instance did not reach the optimum but it takes less seconds.

In Figure 12 they see the worst solutions take a longer time to converge; that this is because the optimum value for the ICI and INF instances has the majority of the percentage. For UPLA the optimum is 13 credits, but the most repeated solution corresponds to 14 credits, which explains why the worst solution takes a longer time to converge. However, the worst solution found by the algorithm has 15 credits for this instance.

## 5. Conclusions

In this work, we proposed an algorithm to solve the balanced academic curriculum problem using the ACO metaheuristic. The experimental evaluation shows the effectiveness of artificial ants solving constraint satisfaction problems. All the tests show that the margin of error is much smaller than the valid solutions delivered. For benchmark instances the quality of the solution was satisfactory and the optimal values of each instance were reached in the majority of the cases. In the real instances the results were also positive. In 2 instances (INF and UPLA) the results were better than in the official curricula and in the ICI results they were equal than in the official curricula. Moreover, the number of iterations to reach the optimum was low. Therefore, the BWAS algorithm is able to satisfy all constraints of the problem in most cases. Thus, we state that our proposal is a fairly reliable alternative to solve the BACP. For future work, we plan to tackle more complicated versions of BACP as was proposed in [1].

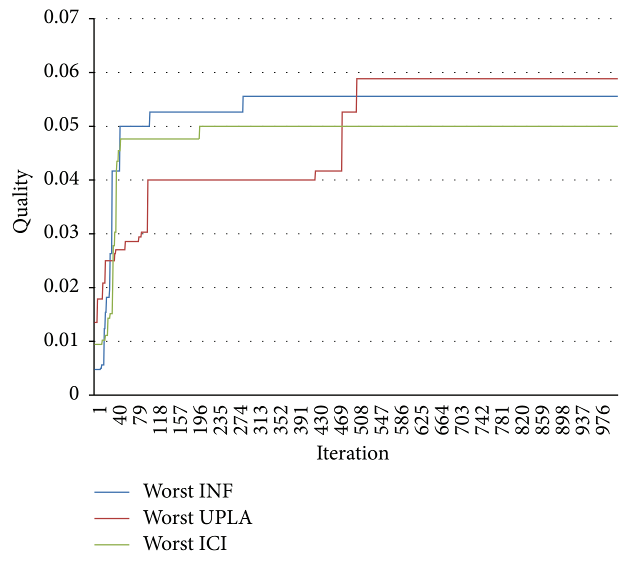

Figure 12: Efficiency of the algorithm in real instances (worst solution).

## Acknowledgments

Nibaldo Rodriguez is supported by Grant CONICYT/FONDECYT/Regular 1131105 and Ricardo Soto is supported by Grant CONICYT/FONDECYT/Iniciación 11130459.

## References

[1] M. Chiarandini, L. D. Gaspero, S. Gualandi, and A. Schaerf, "The balanced academic curriculum problem revisited," *Journal of Heuristics*, vol. 18, no. 1, pp. 119–148, 2012.

[2] J. N. Monette, P. Schaus, S. Zampelli, Y. Deville, and P. Dupont, "A cp approach to the balanced academic curriculum problem," in *Proceedings of the 7th International Workshop on Symmetry and Constraint Satisfaction Problems*, 2007.

[3] C. Castro and S. Manzano, "Variable and value ordering when solving balanced academic curriculum problems," in *Proceedings of the 6th Workshop of the ERCIM WG on Constraints*, June 2001.

[4] B. Hnich, Z. Kiziltan, and T. Walsh, "Modelling a balanced academic curriculum problem," in *Proceedings of the CP-AI-OR*, 2002.

[5] C. Castro, B. Crawford, and E. Monfroy, "A quantitative approach for the design of academic curricula," in *Human Interface and the Management of Information. Interacting in Information Environments*, vol. 4558 of *Lecture Notes in Computer Science*, pp. 279–288, 2007.

[6] T. Lambert, C. Castro, E. Monfroy, M. C. Riff, and F. Saubion, "Hybridization of genetic algorithms and constraint propagation for the BACP," in *Proceedings of the 21st International Conference on Logic Programming (ICLP '05)*, pp. 421–423, October 2005.

[7] T. Lambert, C. Castro, E. Monfroy, and F. Saubion, "Solving the balanced academic curriculum problem with an hybridization of genetic algorithm and constraint propagation," in *Artificial Intelligence and Soft Computing—ICAISC 2006*, vol. 4029 of *Lecture Notes in Computer Science*, pp. 410–419, 2006.

[8] M. Dorigo and C. Blum, "Ant colony optimization theory: a survey," *Theoretical Computer Science*, vol. 344, no. 2-3, pp. 243–278, 2005.

[9] O. Cordón, I. F. de Viana, and F. Herrera, "Analysis of the best-worst ant system and its variants on the qap," in *Ant Algorithms*, pp. 228–234, 2002.

[10] O. Cordón, I. F. de Viana, F. Herrera, and L. Moreno, "A new aco model integrating evolutionary computation concepts: the best-worst ant system," in *Ant Algorithms*, 2000.