A Genetic Local Search Algorithm for the Multiple Optimisation of the Balanced Academic Curriculum Problem

Chapter *in* Communications in Computer and Information Science · January 2009

DOI: 10.1007/978-3-642-02298-2_119

3 authors:

Carlos Castro
Universidad Técnica Federico Santa María

Broderick Crawford
Pontificia Universidad Católica de Valparaíso

Eric Monfroy
University of Nantes

A Genetic Local Search Algorithm for the Multiple Optimisation of the Balanced Academic Curriculum Problem

Carlos Castro[1,⋆], Broderick Crawford[2], and Eric Monfroy[1]

[1] Departamento de Informática, Universidad Técnica Federico Santa María
Valparaíso, Chile
FirstName.LastName@inf.utfsm.cl
[2] Escuela de Informática, Pontificia Universidad Católica de Valparaíso
Valparaíso, Chile
FirstName.LastName@ucv.cl

Abstract. We deal with the Balanced Academic Curriculum Problem, a real world problem that is currently part of CSPLIB. We introduce a Genetic Local Search algorithm to solve this problem using two objectives which is a more realistic model than the one we used in our previous research. The tests carried out show that our algorithm obtains better solutions than systematic search techniques in the same amount of time.

1 Introduction

Integer Programming (IP) and Constraint Programming (CP) techniques have been successfully applied for solving real-life combinatorial problems. However, when problems become too hard, these NP problems cannot be solved using complete methods. Assuming that an incomplete approach can obtain good solutions very quickly, we have been interested in solving a real-life combinatorial problem using a Genetic Local Search algorithm.

The Balanced Academic Curriculum Problem (BACP) consists of assigning courses to periods in such a way that the academic load of each period will be balanced, i.e., as similar as possible. We consider, as academic load, the notion of credit that represents the effort in hours per week that a student needs to successfully follow a course. We concentrate on the three Informatics careers offered by the Federico Santa María Technical University at Valparaíso. In previous works, we have been able to solve each problem independently using both IP and CP techniques [3]. The careers mentioned share some common courses and so in trying to solve a realistic problem, they should ideally be assigned to the same period. We are now currently interested in solving the Multiple BACP. We have designed a Genetic Local Search algorithm. Preliminary results show that the proposed algorithm gives quicker and better solutions than the ones obtained by complete methods.

⋆ The first author has been partially supported by the Chilean National Science Fund through the project FONDECYT 1070268.

Y. Shi et al. (Eds.): MCDM 2009, CCIS 35, pp. 824–832, 2009.

This paper is organized as follows: in Section 2, we briefly describe the BACP. In Section 3, we present the new model of BACP. In Section 4, we introduce the Genetic Local Search Algorithm and its components. Section 5 presents the tests solved using CP techniques, as examples of a complete approach, and the Genetic Local Search Algorithm, as incomplete technique, and we evaluate and compare the results. Finally, in Section 6, we conclude the paper and give further research lines.

2 Balanced Academic Curriculum Problem

The BACP consists of assigning courses to periods in such a way that the academic load of each period will be balanced, i.e., as similar as possible. We consider, as academic load, the notion of credit that represents the effort in hours per week needed to successfully follow the course. We concentrate on three Informatics careers offered by the Federico Santa María Technical University at Valparaíso. The first attempt of balancing was done on the curriculum of a 8 academic periods career considering 48 courses [8]. In that work, IP techniques only solved 6 academic periods. Later on, we also applied CP as an alternative and the BACP was successfully solved using both IP and CP techniques [3]. For each of the three careers used as tests we were able to find the optimal solution. Then, BACP was included in the CSPLIB and it has also been used by other researchers to evaluate the performance of different models [5,4]. As a general framework, we consider administrative as well as academic regulations:

Academic Curriculum. An academic curriculum is defined by a set of courses and a set of precedence relationships among them.

Number of periods. Courses must be assigned within a maximum number of academic periods.

Academic load. Each course has associated a number of credits or units that represent the academic effort required to successfully follow it.

Prerequisites. Some courses can have other courses as prerequisites.

Minimum academic load. A minimum amount of academic credits per period is required to consider a student as full time.

Maximum academic load. A maximum amount of academic credits per period is allowed in order to avoid overload.

Minimum number of courses. A minimum number of courses per period is required to consider a student as full time.

Maximum number of courses. A maximum number of courses per period is allowed in order to avoid overload.

3 Multiple BACP Model

Based on the encouraging results of solving each problem separately, in this work we are interested in solving the three problems simultaneously. The main motivation for modelling and solving this new problem comes from the fact that the three informatics careers at the Federico Santa María Technical University share a set of courses covering fundamentals of informatics. The difference among these threee careers mainly concerns the last years of each curriculum. In fact, the first years of the three careers are seen as a common cycle where students follow almost the same courses. In this section, we present an IP model for the Multiple BACP (MBACP). In this model the objective is to find an homogeneous courses allocation for each semester and for each career. We also impose the constraint that common courses must be shared.

- Parameters
  Let
  θ : Number of careers
  n_z : Number of academic periods in career z; $\forall z = 1, \ldots, \theta$
  n : Number of maximum academic periods ($n = Max\,(n_1, \ldots, n_\theta)$)
  m : Number of courses
  α_i : Number of credits of course i; $\forall i = 1, \ldots, m$
  $\phi_{iz} = \begin{cases} 1\ if\ course\ i\ is\ included\ in\ career\ z; \forall i = 1, \ldots, m; \forall z = 1, \ldots, \theta \\ 0\ en\ otro\ caso \end{cases}$
  β : Minimum academic load allowed per period
  γ : Maximum academic load allowed per period
  δ : Minimum amount of courses per period
  ϵ : Maximum amount of courses per period
- Decision variables
  Let

$$x_{ij} = \begin{cases} 1 \text{ if course } i \text{ is assigned to period } j; \forall i = 1, \ldots, m; \forall j = 1, \ldots, n \\ 0 \text{ otherwise} \end{cases}$$

  c_{jz}: academic load of period j in career z; $\forall j = 1, \ldots, n_z$; $\forall z = 1, \ldots, \theta$
  Max_z: maximum academic load for all periods in career z; $\forall z = 1, \ldots, \theta$
  Min_z: minimum academic load for all periods in career z; $\forall z = 1, \ldots, \theta$
  Δc_z: Difference between Max_z and Min_z in career z; $\forall z = 1, \ldots, \theta$
  Δc: Maximum among differences between Max_z and Min_z for all career z

$$\Delta c = Max\,(\Delta c_1, \ldots, \Delta c_\theta)$$

- Objective function

$$Min\ \ \Delta c$$

- Constraints
  - Course b has course a as prerequisite:

$$x_{bj} \leq \sum_{r=1}^{j-1} x_{ar} = 1; \forall j = 2, \ldots, n$$

  - The academic load of period j in career z is defined by:

$$c_{jz} = \sum_{i=1}^{m} \phi_{iz} \times \alpha_i \times x_{ij}; \forall j = 1, \ldots, n_z; \forall z = 1, \ldots, \theta$$

  - The academic load of period j in career z must be greater than or equal to the minimim required:

$$c_{jz} \geq \beta; \forall j = 1, \ldots, n; \forall z = 1, \ldots, \theta$$

  - The academic load of period j in career z must be less than or equal to the maximum allowed:

$$c_{jz} \leq \gamma; \forall j = 1, \ldots, n; \forall z = 1, \ldots, \theta$$

  - The number of courses of period j in career z must be greater than or equal to the minimum allowed:

$$\sum_{i=1}^{m} \phi_{iz} \times x_{ij} \geq \delta; \forall j = 1, \ldots, n_z; \forall z = 1, \ldots, \theta$$

  - The number of courses of period j in career z must be less than or equal to the maximum allowed:

$$\sum_{i=1}^{m} \phi_{iz} \times x_{ij} \leq \epsilon; \forall j = 1, \ldots, n_z; \forall z = 1, \ldots, \theta$$

  - All course i in career z must be assigned to some period j less than or equal to n_z

$$\sum_{j=1}^{n_z} x_{ij} \geq \phi_{iz}; \forall i = 1, \ldots, m; \forall z = 1, \ldots, \theta$$

  - The maximum difference among load academics for all the periods of career z is defined by:

$$\Delta c_z = Max_z - Min_z; \forall z = 1, \ldots, \theta$$

    It can be represented by using the following linear constraints:

$$\begin{aligned} c_{jz} &\leq Max_z; \forall j = 1, \ldots, n_z; \forall z = 1, \ldots, \theta \\ c_{jz} &\geq Min_z; \forall j = 1, \ldots, n_z; \forall z = 1, \ldots, \theta \\ \Delta c_z &= Max_z - Min_z; \forall z = 1, \ldots, \theta \end{aligned}$$

Remark 1. We have used Δc in this model as a mesure of balancing. However, it is important to remark that we can define other objective functions which also could represent the final goal of to have the three careers with a curricula load balanced.

4 The Genetic Local Search Algorithm

We have designed a simple genetic local search algorithm based on Aydin and Fogarty's work [1]. Roughly speaking, it is a population search algorithm. The algorithm uses a mutation-like operator (MSA) which does a simulated annealing procedure. Thus, the algorithm does both exploration and exploitation according to the temperature values. The operator takes into account the constraints by making moves which generate only feasible solutions. We have decided to only use a mutation-like operator without crossover, given the constraints complexity.

Representation. Representation of BACP strongly depends on the technique used to solve the problem. In [4], the authors use a binary model for IP and an integer domain model for CP allowing each technique to take advantage of the model. In our case, we use an integer-coded representation that can easily manage most of the constraints and with a reduced search space: an integer variable x_i is defined for each course i and the problem consists in finding a value in $[1, \ldots, n]$ for each variable where n represents the maximum number of academic periods a course can be assigned to. It is important to remark that the evaluation of each individual of the population at each generation becomes harder than using a binary representation. We explain how we have taken into account this inconvenience in the following sections.

Evaluation Function. MBACP is a multiple optimisation problem: the best individual must minimise the load differences among the periods for the three careers satisfying all of the problem constraints. The evaluation function is computed using the objective function described in the model problem section. Thus, the individual values require to be translated into the binary format for evaluation.

Algorithm Structure. The idea of combining Genetic Algorithms (GA) and Simulated Annealing (SA) is to take advantage of a diverse population provided by the GA and Hill-Climbing provided by SA. Some research reported good results for many applications [6,7,1]. Our algorithm is based on the generic structure proposed by Aydin and Fogarty as follows:

```
Begin /* Algorithm ESA */
Generate Initial Population
For j=1 to NumberofGenerations
   Evaluate Population
   Copy the best individual in the next population
   i=0
   Repeat
        Select an individual
        Operate by MSA and get a new individual
        i=i+1
   until i = popsize
EndFor
End /* Algorithm */
```

The SA operator, named MSA, begins at the highest temperature each time, thus it is able to do more exploration than applying traditional SA mechanisms. We define a mutation-like operator (MSA) which does a simulated annealing procedure, as follows:

Begin `/* Procedure SA operator */`
`Set the highest temperature` ($Tmax$)
`solution=fitness(Individual)`
$T = Tmax$
Repeat
`newsolution=Allowed-Move(Individual)`
If `(newsolution-solution)`< 0 **then** `acceptnewsolution`
Else
    $R = random(0, 1)$
    `If` $exp(\frac{-(newsolution-solution)}{T}) > R$ **then**
    `acceptnewsolution`
Endif
$T = f(T)$
until T `reached a pre-defined certain level`
return (newsolution, Individual)
End `/* Procedure */`

It determines an allowed move, i.e, it changes the period of a course which was selected randomly, only if this change respects the constraints. For efficiency reasons, only constraints involved in the move are checked. We also define a partial evaluation to decide either to reject or to accept the move. Due that a move does not touch all courses, it is not required to do a complete evaluation again. Roughly speaking, given an evaluation function value of an individual, we reduce the fitness-contribution of the course selected to be moved and we add the fitness-contribution obtained after the move application, allowing to strongly reduce the computing time of MSA, as follows:

Begin `/* Procedure Allowed Move and Partial Evaluation */`
$Course = random(1, nbcourses)$
$j = Period$
$j_1 = random(course - allowed - periods)$
In case of `Course only belongs to z-career`
then
    $c_{jz} = c_{jz} - \alpha_{Course}$
    $c_{j_1z} = c_{j_1z} + \alpha_{Course}$
In case of `Course belongs to common-career-courses`
then `for all k careers which include Course compute`
    $c_{jk} = c_{jk} - \alpha_{Course}$
    $c_{j_1k} = c_{j_1k} + \alpha_{Course}$
`return(`$fitness$`)`
End `/* Procedure */`

The mutation based on simulated annealing checks constraints related to courses precedence order. We can change a course only to a feasible period, considering the constraints 'before' and 'after'. This task is accomplished using the Oz language for arc-consistency techniques where filtering domains reduce the search space. The algorithm works with elitism and for selection uses the Roulette-Wheel procedure. Simulated annealing works with cooling rate equal to 0.955 applied at each iteration, as it is proposed by Aydin et al. [2].

5 Experimental Results

We compare among the results obtained using CP and our evolutionary approach for solving three careers involving 8, 10, and 12 academic periods, respectively. The first tests use Δc as the evaluation function, it corresponds to the difference between the minimum and the maximum load period for the three careers. To carry out the tests we have used an Athlon XP 3200+ with 512 MB RAM. The following table presents the results obtained using version 1.3.0 of Oz (`www.mozart-oz.org`):

Time (seconds)	Δc (credits	Δc_1 (credits	Δc_2 (credits	Δc_3 (credits
1	4	4	4	4

In this case, the best solution is obtained in 1 second, and for each career the maximum difference among academic loads is four credits. The following table presents the results obtained using our GLSA:

Iteration	Time (seconds)	Δc (credits)
1	0	29
4	1	7
6	1	6
12	4	5
58	21	4

We have determined by tuning a maximum of 4176 iterations (about 30 minutes) and a population size of 10 individuals. For the Simulated Annealing operator we have fixed a maximum of 400 iterations and an initial temperature of 140. In this case, the best solution is obtained in 21 seconds, and for each career the maximum difference among academic loads is four credits. We can see that the best solution given by our GLSA is the same solution given by Oz but our algorithm takes 58 seconds and Oz takes only 1 second.

It seems that the evaluation function has problems to guide the search because it considers all academic loads as if they belong to only one career. For instance, Δc is equal to 3 at least in the next two cases: when each Δc_j is 3 $\forall j$, and also when $\Delta c_1 = 1$, $\Delta c_2 = 2$ and $\Delta c_3 = 3$. Thus, the algorithm searchs to minimise the complete difference and not the partial difference for each career. We should obviously like that the algorithm considers as best solution, between them, the second one instead of the first one. Intending to improve the results we tried solving the problem using the following evaluation function: $\Delta c_1 + \Delta c_2 + \Delta c_3$. The following table presents the results obtained using Oz when solving the problem:

Time (seconds)	$\sum_3^{i=1} \Delta c_i$(credits)	Δc_1(credits)	Δc_2(credits)	Δc_3(credits)
1	12	4	4	4
80	11	4	3	4
84	10	4	3	3

In this case, the best solution is obtained in 84 seconds, and for each career the maximum difference among academic loads is four, three, and three credits, respectively. We can see that the best solution given by Oz is obtained in 84 seconds and using the new evaluation function it is able to find a better solution than the previously reported. Finally, the following table presents the results obtained using our GLSA when solving the problem and considering the same parameters for our algorithm:

Iteration	Time (seconds)	$\sum_3^{i=1} \Delta c_i$ (credits)	Iteration	Time (seconds)	$\sum_3^{i=1} \Delta c_i$ (credits)
1	0	87	8	2	11
3	1	85	13	4	9
4	1	20	40	14	8
5	1	15	326	120	6
7	2	13			

The best solution is obtained in 120 seconds, and for each career the maximum difference among academic loads is two credits. Changing the evaluation function has allowed to better guide the search carried out by both the CP technique and mainly by our evolutionary algorithm. It seems that the operator we use in our evolutionary algorithm better simulate the heuristics carried out by humans when solving this problem: once a feasible solution is obtained this is improved by changing a course from a period with too much credits to another one with few credits. This kind of improvement is not easy to perform by CP techniques because they do not have a global view of assignments.

6 Conclusions

We have presented the Multiple BACP and we have designed and implemented a basic Genetic Local Search algorithm that outperforms a complete approach represented by CP techniques for solving this problem. We continue studying different evaluation functions in order to improve these results and we are also considering different mathematical models. As further work we are interested in tuning the GLSA to better understand the advantages and drawbacks of our implementation.

References

1. Aydin, M., Fogarty, T.: A Distributed Evolutionary Simulated Annealing for Combinatorial Optimisation Problems. J. of Heuristics 10(3), 269–292 (2004)
2. Aydin, M., Fogarty, T.: A simulated annealing algorithm for multi-agent systems: a job-shop scheduling application. J. of Intell. Manufacturing 15(6), 805–814 (2004)
3. Castro, C., Manzano, S.: Variable and Value Ordering When Solving Balanced Academic Curriculum Problems. In: 6th Workshop ERCIM 2001, Prague, Czech Republic (June 2001)
4. Hnich, B., Kiziltan, Z., Miguel, I., Walsh, T.: Hybrid Modelling for Robust Solving. Annals of Operations Research 130(1-4), 19–39 (2004)
5. Hnich, B., Kiziltan, Z., Walsh, T.: Modelling a Balanced Academic Curriculum Problem. In: 4th Intl. Workshop on Integration of AI and OR Tech. in CP for Combinatorial Opt. Problems, Le Croisic, France, March 2002, pp. 121–132 (2002)
6. Jeong, I., Lee, J.: Adaptive Simulated Annealing Genetic Algorithm for System Identification. Eng. Applications of Artificial Intelligence 9(5), 523–532 (1996)
7. Syw, W.: Hybrid simulated annealing/genetic algorithm approach to short term hydro-termal scheduling with multiple thermal plants. Elect. Power and Energy Systems 23, 565–575 (2001)
8. Vergara, C.: Desarrollo de un Modelo de Programación Lineal para la Generación de Mallas Curriculares Balanceadas. Título IEI, UTFSM, Valpso, Chile (December 1994)