Article

Metaheuristic with Cooperative Processes for the University Course Timetabling Problem

Martín H. Cruz-Rosales [1], Marco Antonio Cruz-Chávez [2,*], Federico Alonso-Pecina [1], Jesus del C. Peralta-Abarca [3], Erika Yesenia Ávila-Melgar [2], Beatriz Martínez-Bahena [2] and Juana Enríquez-Urbano [2]

[1] Faculty of Accounting, Administration & Informatics, Autonomous University of Morelos State, Morelos 62209, Mexico; mcr@uaem.mx (M.H.C.-R.); federico.alonso@uaem.mx (F.A.-P.)
[2] Research Center in Engineering and Applied Sciences, Autonomous University of Morelos State, Morelos 62209, Mexico; erikay@uaem.mx (E.Y.Á.-M.); bmartinezb@uaem.mx (B.M.-B.); juana.enriquez@uaem.mx (J.E.-U.)
[3] Research Faculty of Chemical Sciences and Engineering, Autonomous University of Morelos State, Morelos 62209, Mexico; carmen.peralta@uaem.mx
* Correspondence: mcruz@uaem.mx

Abstract: This work presents a metaheuristic with distributed processing that finds solutions for an optimization model of the university course timetabling problem, where collective communication and point-to-point communication are applied, which are used to generate cooperation between processes. The metaheuristic performs the optimization process with simulated annealing within each solution that each process works. The highlight of this work is presented in the algorithmic design for optimizing the problem by applying cooperative processes. In each iteration of the proposed heuristics, collective communication allows the master process to identify the process with the best solution and point-to-point communication allows the best solution to be sent to the master process so that it can be distributed to all the processes in progress in order to direct the search toward a space of solutions which is close to the best solution found at the time. This search is performed by applying simulated annealing. On the other hand, the mathematical representation of an optimization model present in the literature of the university course timing problem is performed. The results obtained in this work show that the proposed metaheuristics improves the results of other metaheuristics for all test instances. Statistical analysis shows that the proposed metaheuristic presents a different behavior from the other metaheuristics with which it is compared.

Keywords: restart; landscape; hamming distance; collective communication; point-to-point communication

check for updates

Citation: Cruz-Rosales, M.H.; Cruz-Chávez, M.A.; Alonso-Pecina, F.; Peralta-Abarca, J.d.C.; Ávila-Melgar, E.Y.; Martínez-Bahena, B.; Enríquez-Urbano, J. Metaheuristic with Cooperative Processes for the University Course Timetabling Problem. *Appl. Sci.* **2022**, *12*, 542. https://doi.org/10.3390/app12020542

Academic Editors: Absalom Ezugwu, Haruna Chiroma, Laith Abualigah and Roberto A. Vazquez

Received: 13 December 2021
Accepted: 31 December 2021
Published: 6 January 2022

1. Introduction

Resource programming is an activity that needs to be performed in different areas, such as distribution of resources for the manufacture of objects, in work centers for the distribution of tasks to staff or customer service, in the scheduling of machines when establishing the sequence of tasks to be carried out [1], in the distribution of products to place them at sale points, in the programming of transport systems [2], in the sequence of processes in industry, commerce and services, on a computer or any other machine [3], among others. Most of these problems are classified as NP [4] problems. To deal with the problem of resource optimization, since the last century, heuristic and metaheuristic methods have been developed and applied to NP-type problems, which can obtain solutions close to the global optimum in a polynomial time without renouncing the search for the optimal solution. The development of specialized heuristics is very important because, to date, there are no exact deterministic optimization methods that can solve NP-type problems in polynomial times [5]. Examples of some heuristic strategies applied to NP-type problems in different areas of knowledge are presented below:

- In [6], the authors consider that, by optimizing multi-objective problems with different characteristics, the strategy must be changing. To achieve this, they apply Learning Automation in an evolutionary algorithm so that it can be adapted to the characteristics of the problem. Authors adjust the reference vector to improve the ability to solve problems with a Pareto front.
- In [7], an adaptive polyploid memetic algorithm for scheduling trucks at a cross-docking terminal is developed. The author proposes a new adaptive algorithm to program the entry and exit of trucks in a terminal. The adaptive algorithm stores copies of the parental chromosomes before applying the crossover operator. Moreover, various hybridization techniques to facilitate the search process are used.
- In [8], they develop a multi-objective evolutionary algorithm. They propose an angle-based selection strategy and a displacement-based density estimation strategy. These two strategies are used in environmental selection to purify the population. They apply diversity and convergence to compare pairs of individuals and eliminate the worst. They indicate that their method can be easily extended to solve other multi-objective optimization problems.
- To improve the efficiency of the distribution of perishable products, in [9], a study with a mathematical formulation of mixed integers is presented and they solve it with an evolutionary algorithm and compare its results with other metaheuristics. The objective is to minimize the total cost incurred during the service of the truck.
- In [10], they address the problem of vehicle route generation by developing a linear mixed integer programming model to minimize the total cost of the supply chain. They use CPLEX to solve the optimization model for small problems and with the metaheuristics of evolutionary algorithm, variable neighborhood search, taboo search and simulated annealing solve the model for large-scale problem cases.
- In [11], the authors present a proposal for distinguishing between bacterial and viral meningitis using genetic programming and decision trees. The authors consider two different cases. In one case, they consider blood parameters as cerebrospinal; in the other, they are based exclusively on blood data. For this, the authors make use of formulations already used before for the same purpose and methodologies based on machine learning. In experimentation, they make use of a genetic algorithm integrated in MATHLAB.

Resource programming is also used in the academic area to obtain the timetabling for students for every school period. Depending on the school level, designing school timetabling is not always an easy task. It becomes a complicated problem, mainly because the resources are always limited and constraints related to the availability of material resources (rooms, laboratories, support material, technologies and others) and human resources (teacher capacity, academic needs, student needs, time availability and others) have to be considered. Furthermore, every school timetabling is specific for a school year. It means that this schedule cannot be generally applied for the next school year since the circumstances and needs frequently change. The university course timetabling problem (UCTP) aims to distribute school resources in space and time to generate course timetabling per week. Course timetabling can be a table where the classes that are given in each room are scheduled, or it can also be timetabling that includes the courses for each student. Likewise, the representation of class timetabling can be one that contains the class schedules or courses depending on the different control restrictions imposed by a university. UCTP is a NP-complete problem [4,12,13]. In order to tackle NP-complete problems, approximate methods are frequently used. Because of the complexity of the UCTP, when it is applied to real-life practical problems, a parallel simulated annealing metaheuristic is implemented to obtain acceptable solutions, close to the optimal solution, in reasonable computation times.

The first attempts to automate course timetabling were made in 1960, where techniques based on simulating design were used to obtain schedules by a manual simulation process [14]. The basic idea of this technique was to first schedule the classes with the highest degree of restrictions.

Subsequently, at the beginning of the 1970s, more generic techniques began to be applied to solve the problem. Some of the best-known research includes algorithms based on integer programming [15] and network flow [16–18], among others.

The UCTP problem has also been addressed as a transformation to the coloring graphs problem [19,20], where the vertices represent events and edges represent conflicts.

As well, heuristic approaches based on search techniques are presented. The solution methods vary from constraint satisfaction [21], simulated annealing [22,23], genetic algorithms [24] and tabu search [25]. In [26], a complete study of a theoretical model for UCTP is presented, but they do not consider the teachers' constraint. They present benchmarks that can be solved by using various heuristics such as ant colony, iterated local search, genetic algorithms, simulated annealing and tabu search. The benchmark is used in this work for the tests by using the parallel simulated annealing algorithm. In [13], a very complete review of various types of timetabling programming and their formulations with genetic algorithms can be found [27,28], including tabu search and constraint satisfaction.

In [29], the construction of a timetabling programmer for academic institutions with a constraint programming model is presented. In [30], a model that applies tabu search with reinforced learning is presented. In [31], they present a model for UCTP with a three-phase approach. At the first phase, a feasible solution is created; in the second phase, they use simulated annealing to order the solution space created; and in the third phase, they also use simulated annealing with a neighborhood structure to improve the solution by the exchange of events. The algorithms of these optimization heuristics are generic and independent of the problem to be optimized and are known under the name of metaheuristics [32].

In [33], a practical case of UCTP is presented. A constructive heuristic is proposed to find solutions for a real-life instance, for the Faculty of Chemical Sciences and Engineering of the Autonomous University of the Morelos state, in Mexico City. In [34], the authors propose a solution of timetabling courses, for specific academic areas, of the Juarez Autonomous University of Tabasco, Mexico. They model the constraints with specification and validation tools, typical of the Unified Modeling Language (UML). They establish their solution strategy through two stages with a version of the tabu search, and a software to test their solution proposal is implemented. In [35], the authors solve a particular case of UCTP, focused on the assignment of teacher-course timetabling for an academic department considering the traditional restrictions of full-time and part-time teachers, academic profiles, availability of the courses and time. They present a linear integer programming model to solve the case of study and obtain the optimal solution with low computational effort through a classic branch and bound algorithm.

In [36], a parallel genetic algorithm to solve UCTP is presented. The authors introduce some parallelization techniques using MPI libraries. They assume the master–slave management structure and, based on the number of processing nodes and the size of the population, the scalability of the system and the quality of the solution is estimated. In [37], the authors present a novel real application to maximize the use of resources. They developed a genetic algorithm which uses a simple weighted sum formula to manage conflicts and contemplate teacher preferences. Moreover, a composite aptitude function that considers the use of space is applied. For the tests, they used a large set of real data from the Faculty of Commerce of the University of Alexandria, in Egypt. In [38], a mixed integer linear programming model is present. A hybrid genetic algorithm (HGA) that includes a tabu search strategy is implemented to solve the model. For the analysis, the authors indicate how to map UCTP to the 3D container packaging problem. The tabu search algorithm is used to compare results with the hybrid genetic algorithm. They show that HGA obtains a better solution than the tabu search algorithm in a reasonable time. In [39], a hybrid algorithm based on parallel genetic algorithm and local search to solve UCTP is presented. They combine direct representation of timetabling with crossover operators to ensure that hard constraints are not violated to always obtain feasible solutions. The authors apply instances of Ben Paechter's competences and reproduce some of the known results in the state of the art. In [40], they propose to solve UCTP with a hybrid method based on genetic algorithms and tabu search. They propose a mixed linear programming model, which serves as a reference when defining the problem and the constraints to be considered. For the validation, they use real data for the scheduling of the academic periods, semester 1 of 2018 and semester 2 of 2018, for the academic program of Industrial Engineering, at the Industrial Santander University. The results are compared with the data of the timetabling courses manually obtained by the coordinator. The authors say it can take from hours to days to obtain the data manually, depending on the designer's ability, while their algorithm can obtain the results in a limited time.

As a consequence of the boom that UCTP has had as a model for the assignment and obtaining of timetabling of university courses and the interest of obtaining feasible solutions, in 1995, the International Conference on the Practice and Theory of Automated Timetabling (PATAT) was born. It has been regularly held every 2 years, and its 13th version will be carried out in the year 2022. PATAT also works as a forum for an international community of researchers and practitioners. It has also been supporting a range of competitions and challenges. PATAT is considered as the main conference for the EURO Working Group on Automated Timetabling (EWG-PATAT). Until the fourth conference, held in 2002, UCTP was the main theme and, from 2004, they extended to other topics on timetabling.

In this work, a metaheuristic that addresses the university course timetabling problem (UCTP), which applies processes cooperation with collective communication and point-to-point communication, is presented. It is applied to the UCTP model of Rossi-Doria [26]. A mathematical representation of this model is also proposed.

After the introduction, the paper presents the following sections: Section 2 presents the symbolic representation and the mathematical model of UCTP. Section 3 presents the proposed metaheuristic of simulated annealing with cooperative processes (SACP) that works on a distributed computing system. Section 4 presents the SACP results in efficiency and efficacy; it also includes a statistical analysis of the proposed metaheuristics. Finally, the conclusions of this work are presented.

2. University Course Timetabling Problem (UCTP)

The symbolic representation for the UCTP problem, used to design the data structure, is shown. It stores the representation for the instance to be solved by the proposed metaheuristic. Then, the mathematical formulation of the UCTP optimization model, to represent the benchmark of Rossi-Doria [26], is presented. It defines a target function using soft constraints and also features several sets of hard constraints.

2.1. Symbolic Representation

Table 1 shows the representation of a timetable with 45 timeslots. The relationship among days (Monday to Friday) and periods (five time periods) forms a set of 45 timeslots which, when scheduling events and assigning them to the timeslots, a timetable for each room is obtained. So, the solution that is obtained in UCTP has as many tables as rooms exist. Therefore, a solution will consist of a set of timetables.

The schedule table is the abstraction of a two-dimensional structure for each room, with the scheduling of the week, which is formed with the days on the X axis and the periods on the Y axis. In this two-dimensional structure, the events that are taught and the students who attend, during the week each of these events are scheduled. In addition, the room must offer the necessary facilities for the event to be carried out. A third dimension is created to form the set of the schedule tables, by scheduling events and students in the different rooms.

Table 1. Two-dimensional table of schedules assigned to a room with 45 timeslots.

Slot	Day				
	Monday	**Tuesday**	**Wednesday**	**Thursday**	**Friday**
1	1	2	3	4	5
2	6	7	8	9	10
3	11	12	13	14	15
4	16	17	18	19	20
5	21	t (5, 2) = 22	23	24	25
6	26	27	28	29	30
7	31	32	33	34	35
8	36	37	38	39	40
9	41	42	43	44	45

Figure 1 presents a three-dimensional structure with three dimensions: day, period, room. This abstraction of three-dimensional structure can store the UCTP solution by having events assigned to its students. Each coordinate of the matrix, x, y, z, represents a timeslot. This is the structure used by the metaheuristic proposed SACP.

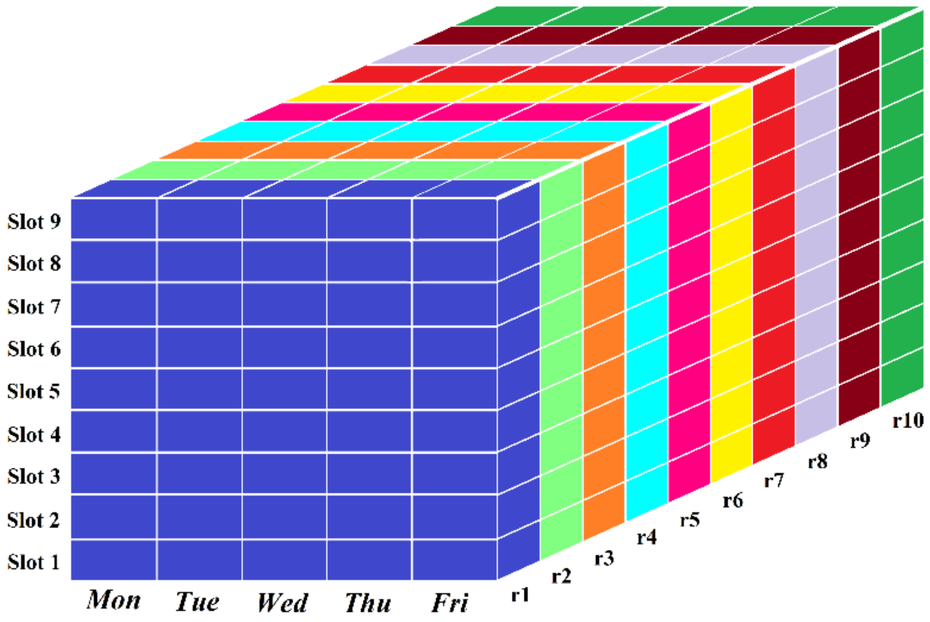


Figure 1. Three-dimensional table of schedules assigned to various rooms (r_1 to r_{10}).

2.2. Optimization Model

According to what Paechter raised in [13,26,41], a feasible solution is the one in which all events are scheduled, assigning them in the possible 45 existing timeslots for each room. This schedule of events considers the capacity of the room, the facilities offered by the room, the needs required by the event and the students who will attend the event. The optimization model is made up of sets of soft constraints and sets of hard constraints. Hard constraints are constraints that must be met to obtain a feasible solution. These constraints are related to the physical resources. Soft constraints are constraints that are preferable to comply with, but they do not need to be met for the solution to have a feasible result. A penalty is assigned, in the objective function, which is to minimize, when the constraints are not met.

The optimization model presents the following sets. The set of events $E = \{1, 2, 3, \ldots, n_E\}$, the set of timeslots $T = \{1, 2, 3, \ldots, 45\}$, the set of days $D = \{1, 2, 3, 4, 5\}$, the set of periods $P = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, the set of rooms $R = \{1, 2, 3, \ldots, n_R\}$, the set of students $S = \{1, 2, 3, \ldots, n_S\}$ and the set of facilities $F = \{1, 2, 3, \ldots, n_F\}$. Equation (1) minimizes the objective function which exclusively accounts for violations committed to the soft constraints, *Soft1*, *Soft2* and *Soft3*, that are related to each student and that are implicit in functions *F1*, *F2* and *F3*. The soft constraint *Soft1* indicates that a student $s \in S$ should not have class in the last period of the day. The soft constraint *Soft2* indicates that a student should not have more than two adjoining classes. The soft constraint *Soft3* indicates that a student should not have a single class per day, from the set $T = \{1, 2, 3, \ldots, 45\}$ of timeslots, which represents 45 h available in a 5-day school week with 9 periods per day. Events can be scheduled with students and the schedule table can be found for each room, where each room can offer different facilities and each event can have different needs. The UCTP optimization model is presented with the formulation from (1) to (7).

$$min\ OF = \sum_{s=1}^{n_A} F1(s, Soft1) + \sum_{s=1}^{n_A} F2(s, Soft2) + \sum_{s=1}^{n_A} F3(s, Soft3) \tag{1}$$

s.t.

$$X(e, \Phi(t,r)) = \begin{cases} 1, & if \Phi(t,r) = 1\ and\ R(r,e) = 1 \\ 0, & In\ another\ way \end{cases} \qquad \forall e \in E \tag{2}$$

$$\sum_{t=1}^{45} \sum_{r=1}^{n_R} X(e, \Phi(t,r)) = 1 \qquad \forall e \in E \tag{3}$$

$$\sum_{e=1}^{n_E} X(e, \Phi(t,r)) \leq 1 \qquad \begin{matrix} \forall t \in T \\ \forall r \in R \end{matrix} \tag{4}$$

$$\sum_{s=1}^{n_s} Z(s, X(e, \Phi(t,r))) \leq Capacity(r) \qquad \begin{matrix} \forall e \in E \\ \forall t \in T \\ \forall r \in R \end{matrix} \tag{5}$$

$$\sum_{e=1}^{n_E} \sum_{r=1}^{n_R} Z(s, X(e, \Phi(t,r))) \leq 1 \qquad \begin{matrix} \forall s \in A \\ \forall t \in T \end{matrix} \tag{6}$$

$$t, r, e, s\ \in (N - \{0\}) \tag{7}$$

The five sets of hard constraints presented from (2) to (6) have to be satisfied to obtain a feasible solution to the optimization model. A pair (t, r) is defined, where $t \in T$ represents a timeslot and $r \in R$ represents a room, such that:

$$\Phi(t,r) = \begin{cases} 1, & If\ any\ e\ is\ assigned\ in\ t\ of\ r \\ 0, & In\ another\ way \end{cases} \tag{8}$$

For example, if $\Phi(22, 3) = 1$, this indicates that there is an assigned event in timeslot 22 in room 3. A pair (r, e) is defined for each $r \in R$ room and for each $e \in E$ event, such that:

$$R(r,e) = \begin{cases} 1, & If\ r\ meets\ the\ needs\ of\ e \\ 0, & In\ another\ way \end{cases} \tag{9}$$

The set of constraints in (2) indicates that a room r must meet the needs of the event e that is scheduled in timeslot t. If Equations 8 and 9 are true, that is, they have a value of 1, the relation $X(e, \Phi(t,r))$ is satisfied.

The set of constraints in (3) indicates that all events have to be scheduled. For each $e \in E$, the expression (3) of the set of constraints checks that all events e are scheduled in some timeslot t of some room r $(\Phi(t,r) = 1)$ and that meets the needs of the event.

Moreover, each event can be scheduled only once in the week and, because of this, the equality must always be 1.

The set of constraints in (4) indicates that, in each room alone, there will be one event at a time. When in some timeslot $t \in T$ and some room $r \in R$, $(e, \Phi(t,r)) = 1$, then the event e is scheduled in the timeslot t of room r, $(\Phi(t,r) = 1)$. So $X(e', \Phi(t,r)) = 0$, $\forall e' \in E$, with $e' \neq e$.

The set of constraints in (5) indicates that room r must have sufficient capacity to attend to the student of the event. To define this constraint, from the expression (10) that indicates that for each student $s \in S$, this will be true, if the student s is attending the event e that occurs in the timeslot t of room r, $(\Phi(t,r) = 1)$. So, the number of students in class should not exceed the capacity of the room.

$$Z(s, X(e, \Phi(t,r))) = \begin{cases} 1, & if\ X(e, \Phi(t,r)) = 1,\ s\ attends\ to\ e \\ 0, & In\ another\ way \end{cases} \tag{10}$$

The set of constraints in (6) indicates that a student will not attend more than one event at the same time. For a student $s \in S$, the expression (10) means that if the student s is attending the event e in the room r, then this student will not be able to attend another event e' in another room in the same timeslot, because $Z(s, X(e', \Phi(t,r))) = 0$, $\forall e \in E$, with $e \neq e$. Hence, when evaluating the expression (6) (with value 0 or 1) of the set of constraints, it indicates that, if the expression (10) is true (equal to 1), and the sum in (6) is 0, the student s does not attend any event in any room in the timeslot t. Otherwise, if the sum in (6) is 1, then the student s attends a single event in a single room in the timeslot t.

In (7), the values of the variables are within the set of natural numbers.

For each of the soft constraints, *Soft1*, *Soft2* and *Soft3*, that make up the function to be optimized, in Equation (1), each occurrence of violation, of any of these constraints, assigns to the penalty variable the value of one. For the soft constraint *Soft1*, a timeslot t, an event $e \in E$ and a room $r \in R$ in which is the student $s \in S$, a soft constraint is evaluated for each student (11):

$$F1(s, Soft1) = \begin{cases} 1, & If\ Z(s, X(e, \Phi(t,r))) = 1, \text{to } t \in 41 \leq t \leq 45 \\ 0, & In\ another\ way \end{cases} \tag{11}$$

The penalty for the occurrence of $F1(s, Soft1) = 1$ indicates that a student should not have a class in the last period of the day, which is in $41 \leq t \leq 45$.

For the soft constraint *Soft2*, there will be a day $d \in D$, a period $p \in P$, for each student $s \in S$ it is defined $PD = \{5p + d | 0 \leq p \leq 8\}$ and for each $\{pd1, pd2, pd3\}$ subset in PD of three elements, such that $pd1 = 5\text{p} + d, pd2 = pd1 + 5$ and $pd3 = pd1 + 10$. So, for a $r \in R$ room, a $e \in E$ event, a $d \in D$ day and a $p \in P$ period with $0 \leq p \leq 6$, it is defined in (12):

$$F2(s, Soft2) = \begin{cases} 1, & If\ Z(s, X(e, \Phi(t,r))) = 1, \text{for each } s \in \{pd1, pd2, pd3\} \\ 0, & In\ another\ way \end{cases} \tag{12}$$

The penalty of an occurrence $F2(s, Soft2) = 1$, it indicates that a student should not have more than two classes in a row in continuous periods of the same day.

For the soft constraint *Soft3*, there will be a day $d \in D$, for each student $s \in S$, for each day $d \in D$, it is defined in (13):

$$F3(s, Soft3) = \begin{cases} 1, & if\ \sum_{r=1}^{n_R} \sum_{e=1}^{n_E} \sum_{p=0}^{8} Z(s, X(e, \Phi(5p + d, r))) = 1 \\ 0, & In\ another\ way \end{cases} \tag{13}$$

When a penalty occurs for a student in $F3(s, Soft3) = 1$, it indicates that a student has only one class per day.

For a penalty not to occur in $F3(s, Soft3)$ then it has to be $\sum_{r=1}^{n_R} \sum_{e=1}^{n_E} \sum_{p=0}^{8} Z(s, X(e, \Phi(5p+d, r))) > 1$, which indicates that a student has more than one class in one day.

3. Simulated Annealing with Cooperative Processes (SACP)

In this work, the simulated annealing metaheuristic (SA) is used due to the great capacity it presents to obtain good local optimums through the procedure known as the acceptance criterion that allows to improve the solution quickly, escaping from the local optimum. Unlike population-based metaheuristics that require working with a population of solutions to optimize through exploration of the solutions space, SA only requires an initial solution to perform optimization through exploitation of the solutions space. Because this metaheuristic presents an asymptotic convergence over time, in this work, a cooperation of processes is applied to improve the obtaining of results in shorter times, performing a search with exploitation and exploration procedures in the space of solutions. Exploitation is carried out by applying local search to each SA and exploration is carried out by generating a cooperation of distributed processes that executes each of the SA with quenching-type restarts and cooperating with each other to direct the search toward a space of good solutions. This section presents the simulated annealing (SA) metaheuristics with processes cooperation.

Figure 2 presents a flow diagram of the algorithm. The algorithm generates a SA with restart for each process *i*, so that in each term of an SA the cooperation between a pair of solutions is carried out. This is the best solution TTSA$_i$ of SA obtained by the process *i* vs. the best solution obtained by all processes, BTT$_i$. The full explanation of the algorithm is shown:

1. The algorithm begins its execution in process zero by reading the input file that contains the instance information of the problem to be solved.
2. The information is stored in a data matrix represented in Figure 1.
3. The data types are defined below using the MPI protocol to be able to work with n processes.
4. The information received in process $i = 0$ is sent to each one of the n processes generated using collective communication through the MPI-Bcast function.
5. At the beginning, each process i generates a feasible random initial solution TT*i*. To generate this feasible solution, the Constructive Approach Algorithm is used. This is presented in [33]; this solution is passed to SA.
6. At the end of each process *i* in SA a solution optimized TTSA$_i$ is obtained and collected in process $i = 0$ through each process *i*. The best solution $bs_i = f$(TTSA$_i$) obtained by the cost function (Equation (1)) in SA, through an arrangement of best solutions, BS = Set ($bs_0, \ldots, bs_{n\text{-}1}$) and also the number of process npi = ranki that corresponds to the bsi solution by means of a number of processes array, NP = Set($np_0, \ldots, np_{n-1}$). Collective communication with MPI_Gather is applied to send the best cost value bsi (best solution) of each process *i*, to the process $i = 0$ and thus obtain the BS array located in process $i = 0$.
7. Applying collective communication with MPI_Bcast, the BS array containing the cost of each solution and its Process Number NP, is sent to each process *i*.
8. On each process *i* the cost of the best global solution bgs_i = min (BS) is located in BS, which is the same in each process *i*. However, it is necessary to know in all processes the best number of process $bnp_i = f$ (BS, NP), which contains bgs_i, in order to evaluate the Hamming distance in each process.
9. In the bnp_i process, the best solution of all processes is found in the TTSA$_i$ array. This solution is called Best Timetabling BTT$_i = f$ (bnpi, TTSAi).
10. Applying point-to-point communication with MPI_Send, the BTT$_i$ array is sent to the process $i = 0$ and it is received with MPI_Recv.
11. In process $i = 0$, applying collective communication with MPI_Bcast, the best BTT$_i$ solution is distributed to all the processes.

12. In each process, the Hamming distance is calculated with d_h = d-Hamming (BTT$_i$, TTSA$_i$), evaluating the difference between the two solutions BTT$_i$ vs. TTSA$_i$. With this, the processes cooperation is carried out because it is carried out by reducing d-Hamming between the solution obtained in TTSA$_i$ vs. BTT.
    - 12.1 The Hamming distance is obtained by comparing each timeslot (x, y, z) of both solutions, see Figure 1.
    - 12.2 If the same event is stored in the timeslot (x, y, z) of both solutions, then the similarity of both solutions is the same in that timeslot.
    - 12.3 If the same event is not stored in the timeslot (x, y, z) of both solutions, then there is no similarity in that timeslot.
    - 12.4 The comparison is made between each of the timeslots (x, y, z) of both solutions, BTT$_i$, TTSA$_i$, and the degree of difference between both solutions is counted.
    - 12.5 A large d-Hamming value indicates that the similarity between BTTi and TTSA$_i$ is very little.
    - 12.6 A small d-Hamming value indicates that the similarity between BTT$_i$ and TTSA$_i$ is greater.
13. Figure 3 presents processes cooperation in the flow diagram. TT$_i$ = cooperation (BTTi, TTSAi), brings the best BTTi solution of all processes closer to the best TTSA$_i$ solution of each process by reducing its Hamming distance. Figure 4 presents a simple example of how the processes cooperation is performed.
    - 13.1 A segment segSA$_i$ <— random (di,pi,TTSAi) of TTSA$_i$ is taken randomly with coordinates (x, y) = (d_i, p_i) = (d_2, p_3). This segment includes the full depth of the cube, that is $(z_1, \ldots, z_n) = (r_1, \ldots, r_n)$, see Figure 4c.
    - 13.2 The segSA$_i$ segment replaces the segBi segment that is located at the same coordinates in the BTT$_i$ solution, so that all constraints for events in that segment are maintained.
    - 13.3 The replaced segment segB$_i$ of BTT$_i$ is presented in Figure 4d.
    - 13.4 Although the segSA$_i$ replacement segment does not violate constraints in BTTi, there are repeated events in the new BTT$_i$ <— replacement (segSA$_i$, segBi, BTTi) solution.
    - 13.5 To return the feasibility in BTT$_i$ <— feasibility (BTT$_i$), repeated events Re$_i$ <— find_repeat_ev (segB$_i$, segSA$_i$), presented in the segments of replacement segSA$_i$ and segB$_i$ are searched. In the example in Figure 4c,d, the event Re$_i$ = {2} appears as repeated, and this indicates that event 2 will not appear repeated in the new BTT$_i$ solution (Figure 4e).
    - 13.6 The remaining events 7 and 10 of segSA$_i$ (Figure 4c) will appear repeated at other coordinates in BTT$_i$ (Figure 4e).
    - 13.7 Then, the procedure seeks to insert in BTT$_i$ (Figure 4e) in the position where repeated events 7 and 10 (Figure 4e) are located, to the deleted events Le$_i$ = <— find_lost_ev (segB$_i$, segSA$_i$). Figure 4d shows the lost events {3, 5} when its segB$_i$ segment of BTT$_i$ was removed.
    - 13.8 The deleted events Le$_i$ = {3, 5}, are tried to be replaced by the events 7 and 10 which are not inserted in Figure 4c. These events 7 and 10 are searched in BTT$_i$ outside the segment inserted segSAi, which is defined as a tabu segment. Le$_i$ events already inserted are proven to meet all constraints to obtain a feasible solution in BTT$_i$ = feasibility (BTT$_i$) when any of them do not meet any constraints. The events are inserted anyway and an exchange that is feasible with other events is sought randomly (minus the events of the segment inserted segSA$_i$ which is tabu), until achieving the feasibility of BTT$_i$. Tabu events are not exchanged so as not to increase d-Hamming.
    - 13.9 The previous cooperation procedure makes the new BTT$_i$ solution (Figure 4e) more similar to TTSA$_i$ (Figure 4a). This procedure is performed with each of the TTSA$_i$ solutions of each process i and the best BTT$_i$ solution obtained

from all processes. Once the BTT_i solution is feasible, it is assigned as the new $TT_i = BTT_i$, solution to start with the next SA in process *i*.

14. Figure 2 shows cooperation between processes is repeated every time that ends a restart with SA.
15. The algorithm counts the RSA_i number of SA restarts (NR). If the total number NR_i, has already been reached, the algorithm ends its execution, otherwise the execution of a new SA begins in process *i* with the new TT_i = cooperation (BTT_i, $TTSA_i$) solution, obtained by cooperation between processes. NR_i is the stopping criterion of SACP and is evaluated in each process *i*.

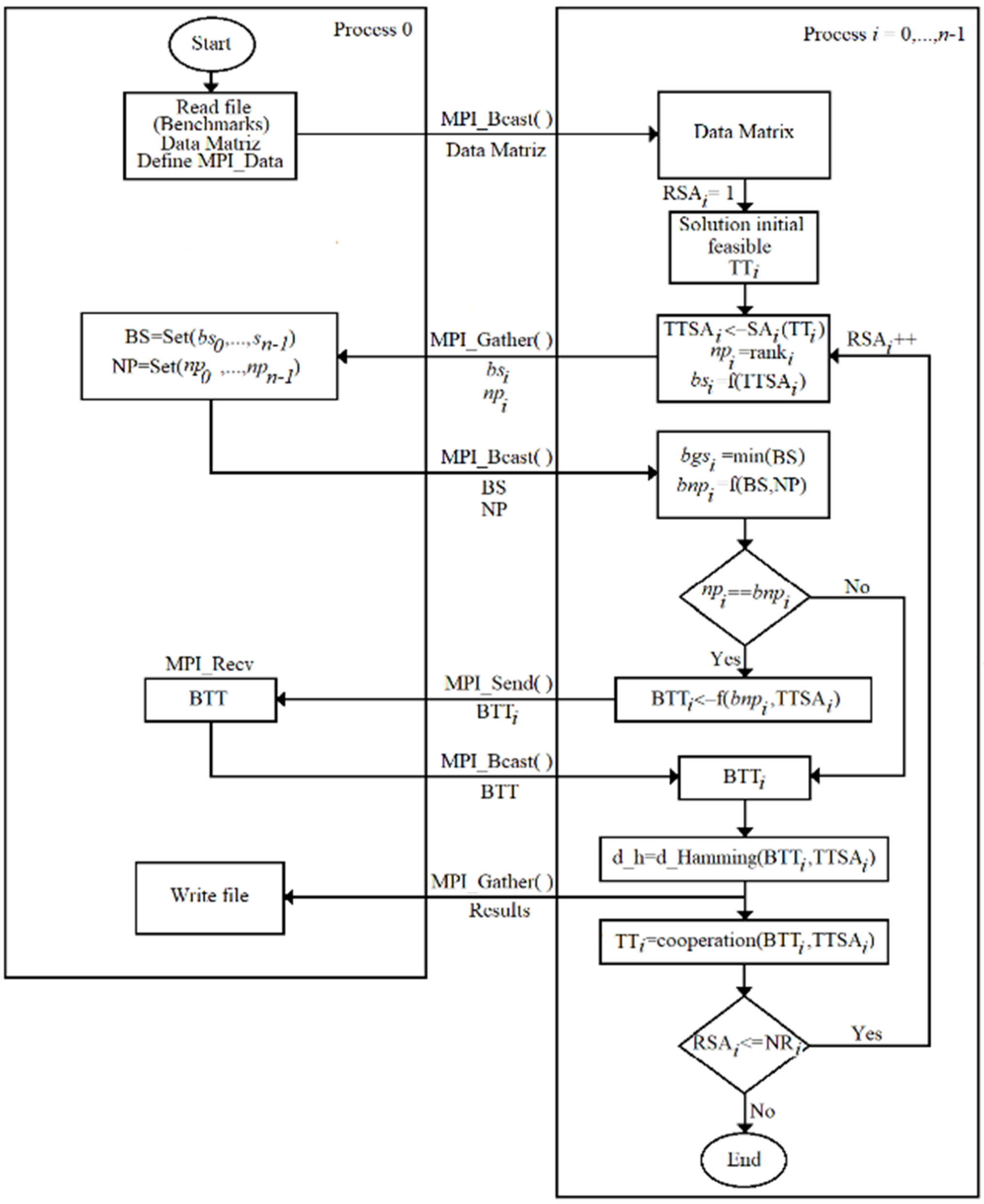
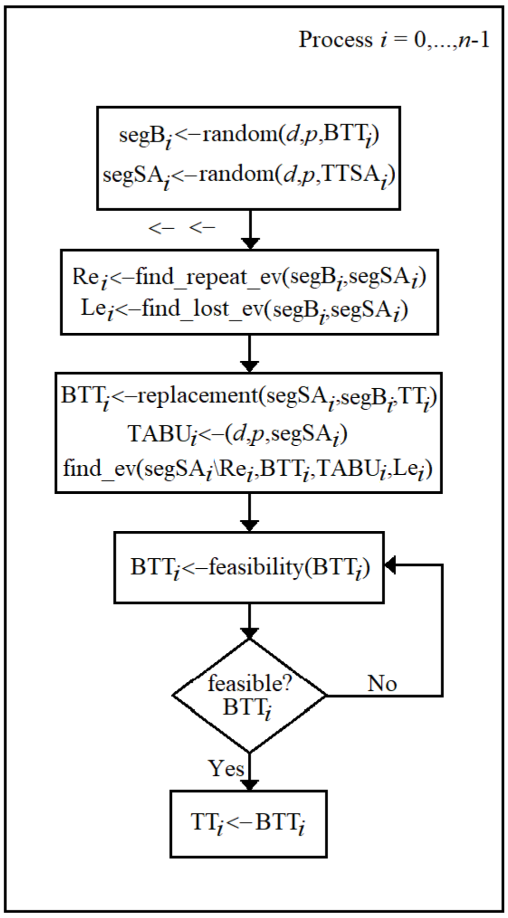
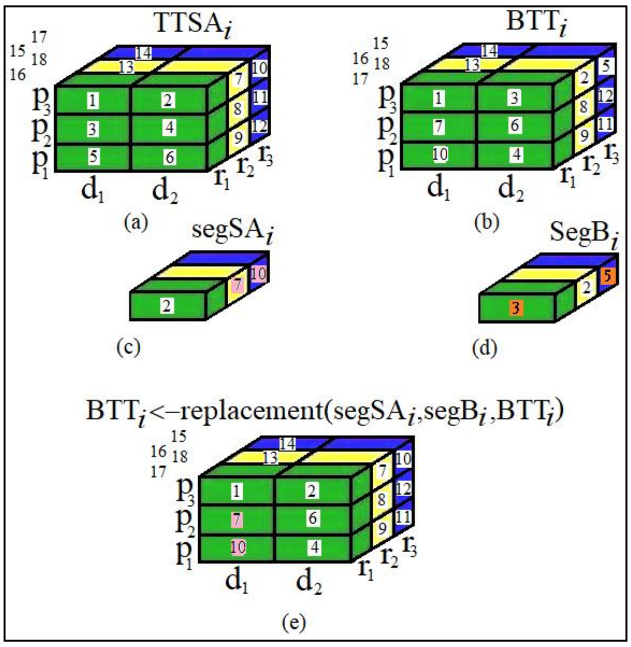

Process 0
Start
Read file (Benchmarks) Data Matriz Define MPI_Data
Process i = 0,...,n-1
MPI_Bcast( )
Data Matriz
Data Matrix
RSA_i = 1
Solution initial feasible TT_i
BS=Set(bs_0,...,s_n-1) NP=Set(np_0,...,np_n-1)
MPI_Gather( )
bs_i
np_i
TTSA_i<-SA_i(TT_i) np_i=rank_i bs_i=f(TTSA_i)
RSA_i++
MPI_Bcast( )
BS
NP
bgs_i=min(BS) bnp_i=f(BS,NP)
np_i==bnp_i
No
Yes
MPI_Recv
BTT
MPI_Send( )
BTT_i
BTT_i<-f(bnp_i,TTSA_i)
MPI_Bcast( )
BTT
BTT_i
d_h=d_Hamming(BTT_i,TTSA_i)
Write file
MPI_Gather( )
Results
TT_i=cooperation(BTT_i,TTSA_i)
RSA_i<=NR_i
Yes
No
End

Figure 2. Simulated annealing with cooperative processes (SACP).



Process i = 0,...,n-1
segB$_i$<–random(d,p,BTT$_i$)
segSA$_i$<–random(d,p,TTSA$_i$)
<– <–
Re$_i$<–find_repeat_ev(segB$_i$,segSA$_i$)
Le$_i$<–find_lost_ev(segB$_i$,segSA$_i$)
BTT$_i$<–replacement(segSA$_i$,segB$_i$,TT$_i$)
TABU$_i$<–(d,p,segSA$_i$)
find_ev(segSA$_i$\Re$_i$,BTT$_i$,TABU$_i$,Le$_i$)
BTT$_i$<–feasibility(BTT$_i$)
feasible? BTT$_i$
No
Yes
TT$_i$<–BTT$_i$

Figure 3. Processes cooperation procedure, TTi = cooperation (BTTi, TTSAi).



TTSA$_i$
BTT$_i$
segSA$_i$
SegB$_i$
BTT$_i$<–replacement(segSA$_i$,segB$_i$,BTT$_i$)
p$_1$ p$_2$ p$_3$ d$_1$ d$_2$ r$_1$ r$_2$ r$_3$
(a)
(b)
(c)
(d)
(e)

Figure 4. Processes cooperation between a pair of solutions, the best obtained by a process vs. the best of all processes (TTSA$_i$ vs. BTT$_i$). (**a**) Best solution obtained by process i, TTSA$_i$. (**b**) Best solution of all processes, BTT$_i$. (**c**) TTSA$_i$ segment to insert into BTT$_i$. (**d**) BTT$_i$ segment to be replaced from BTT$_i$. (**e**) New BTT$_i$ solution with a lower d-Hamming value with TTSA$_i$ that requires a correction to make it feasible (procedure 13.5).

Algorithm 1 presents the proposed simulated annealing algorithm that is applied in SACP within each process *i*. Function SA$_i$ (TT$_i$). The steps are explained below:

1. The SA algorithm always requires starting with a solution feasible of the TT$_i$ problem. This feasible solution is evaluated with the objective function in (1) to obtain the bsc_i cost. The best known bsi solution is initialized with the value of bsc_i.
2. SA input parameters, such as the control parameter T_i, the length of the Markov chain MCL, the control coefficient α, that controls the decrease in the control parameter and the final value control parameter, with the value T_f, which works as the SA stopping criterion, are initialized.
3. Lines 3 and 4, the external cycle of SA is performed as long as T does not reach T_f. In addition, T is decremented with α, see line 28.
4. Lines 5 and 6, the internal cycle represents the cycle of Metropolis, and this is repeated until MCL is met, see line 26.
5. Line 7, a disturbance is performed by applying the neighborhood structure that exchanges a couple of randomly chosen events to assign each of them in a different timeslot. The TT$_i$ feasibility is evaluated with the sets of constraints presented in (2) to (6) of the optimization model, in case it is not feasible, then the pair of events is returned to its initial position and tested with another pair of events until TT$_i$ is feasible. The new feasible solution is stored in TT$_p$.
6. Line 8, the bsp_c cost of the disturbed TT$_p$ solution is obtained with Equation (1).
7. Line 9, the costs of TT$_i$ and TT$_p$ solutions are compared. If a better cost of the disturbed solution is obtained then online 10 and 11, TT$_p$ is accepted as the new TT$_i$ solution.
8. Lines 12 to 17, if the cost of bsp_c is less than or equal to that best solution obtained, then TT$_p$ is stored in TTSA$_i$ as the best solution obtained.
9. Lines 18 to 23 evaluate the criterion of acceptance of SA. This applies when the bsp_c cost obtained by the disturbance turns out to be greater than the cost of bsc_i.
10. Line 18, a random number ρ is generated uniformly distributed between zero and one.
11. Line 19, the Boltzmann function is applied to obtain the paccept probability to accept a poor quality bspc solution.
12. Line 20, if paccept is greater than ρ, then the bspc solution is accepted as a new solution to continue with SA.
13. The SA procedure is repeated as long as T does not reach T_f, see line 4.
14. When SA completes its execution, the best found TTSA$_i$ solution and its bs_i cost are returned as a result.

Algorithm 1. Simulated annealing with restart

```
1   $bs_{ci} = f(\mathrm{TT}_i)$, $bs_i = bs_{ci}$
2   $T_i = 2$, $MCL = 400$, $\alpha = 0.98$, $T_f = 0.001$
3   $T = T_i$
4   while $T >= T_f$ do
5       $MC = 0$
6       while $MC < = MCL$ do
7           $\mathrm{TT}_p <-$ perturbation ($\mathrm{TT}_i$)
8           $bs_{pc} = f(\mathrm{TT}_p)$
9           if $bs_{pc} < = bs_{ci}$ then
10              $bs_{ci} = bs_{pc}$
11              $\mathrm{TT}_i <- \mathrm{TT}_p$
12              if  $bs_{ci} < bs_i$ then
13                  $bs_i = bs_{ci}$
14                  $\mathrm{TTSA}_i <- \mathrm{TT}_i$
15              end if
16          end if
17          else
18              ρ = random number between (0,1)
```

```
19                        $p_{accept} = e^{-(\frac{bs_{pc} - bs_{ci}}{T})}$
20                 if ρ < $p_{accept}$ then
21                     $bs_{ci} = bs_{pc}$
22                     $TT_i$ <— $TT_p$
23                 end if
24             end else
25             MC=MC+1
26         end while
27         $T = T$*α
28     end while
29     return $bs_i$; $TTSA_i$
```

4. Computational Results

To perform the efficiency and efficacy tests of the metaheuristic with cooperative processes SACP, a computational cluster, consisting of four nodes, each node with a motherboard with two processors, Intel Xeon six core 3.06 GHz, with a total of 48 cores, 96 GB RAM, Ethernet connection 100BASE-TX, 100 Mbs, O.S. Centos 5.5. MPI libraries GNU. The instances of the university course timetabling problem are those presented by the Metaheuristic Network group [26]. These are the so-called small01, small02, small03, small04, small05, medium01, medium02, medium03, medium04, medium05, large01 and large02. Table 2 presents the characteristics of these instances, which can be downloaded from [42]. Computational experiments are limited to a few instances because they are the only ones found in the literature for the optimization model that is presented in this work.

Table 2. Characteristics of the instances.

Instance	Small	Medium	Large
Events	100	400	400
Rooms	5	10	10
Features	5	5	10
Approx_features_per_room	3	3	5
Percent_feature_use	70	80	90
Students	80	200	400
Max_events_per_student	20	20	20
Max_students_per_event	20	50	100
Periods	9	9	9
Days per week	5	5	5

To carry out the tests with SACP, a sensitivity analysis of the main SACP parameters was performed. Table 3 presents the tuned values of the parameters by means of a sensitivity analysis, where *NP* indicates the number of processes running in SACP. The sensitivity analysis was applied according to the procedure presented in [43], which performs a sweep of values for each parameter (T_i, T_f, *MCL*, α) in interval from a lower bound up to an upper bound. These limits defined taking as the center of these intervals the value of the parameter that is frequently applied to SA in optimization problems.

Table 3. Tuned simulated annealing values.

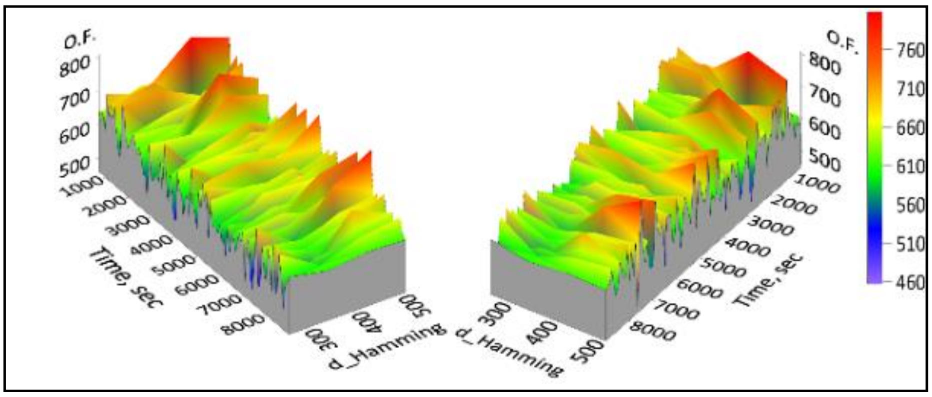
Parameter	Tuned Value
T_i	2
T_f	0.001
MCL	400
α	0.98
NR	12
NP	48

For the *NR* parameter, it was tuned according to the execution time limit defined for SACP, and for the *NP* parameter, it was tuned according to the best results obtained by SACP.

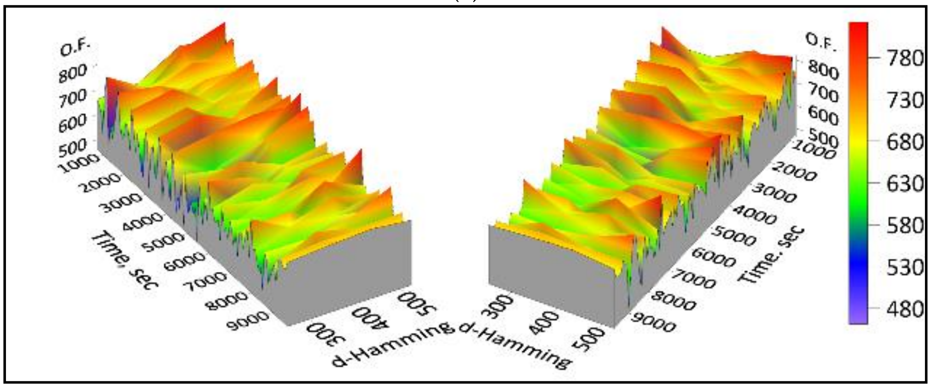
4.1. Landscape and Efficacy of SACP

An analysis of the UCTP landscape was performed for the two large-type instances. The landscape is presented based on three variables, the OF objective function, the Hamming distance and the elapsed time in the SACP execution, which is a function of the *NR* that is executed in SACP. The execution time was about 8000 s.

Figure 5a presents the landscape for the instance large01 without applying cooperation between processes. It is observed that, with greater distance Hamming, the behavior of the solutions obtained as a function of time generates lower quality in the evaluation of the objective function with Equation (1). In addition, similarly, a shorter Hamming distance impacts on the behavior of the solutions obtained as a function of time generating a higher quality in the evaluation of the objective function. However, there are solutions that, despite having a large Hamming distance, the quality of their result is very good; the same happens when the Hamming distance is small. It is observed that, between a Hamming distance of 300 to 400, there is a path in which no peaks appear in the landscape. The opposite occurs when the Hamming distance is less than 300 and greater than 500. Figure 5b presents the landscape for the instance large02 without applying the cooperation among processes. Here, there is no clear behavior depending on the distance. As in large01, there are solutions that, despite having a large Hamming distance, the quality of their result is very good. The same happens when the Hamming distance is small. It is also observed that near to a Hamming distance of 400 there is a path in which no peaks appear in the landscape. The opposite occurs when the Hamming distance is less than 300 and greater than 500.



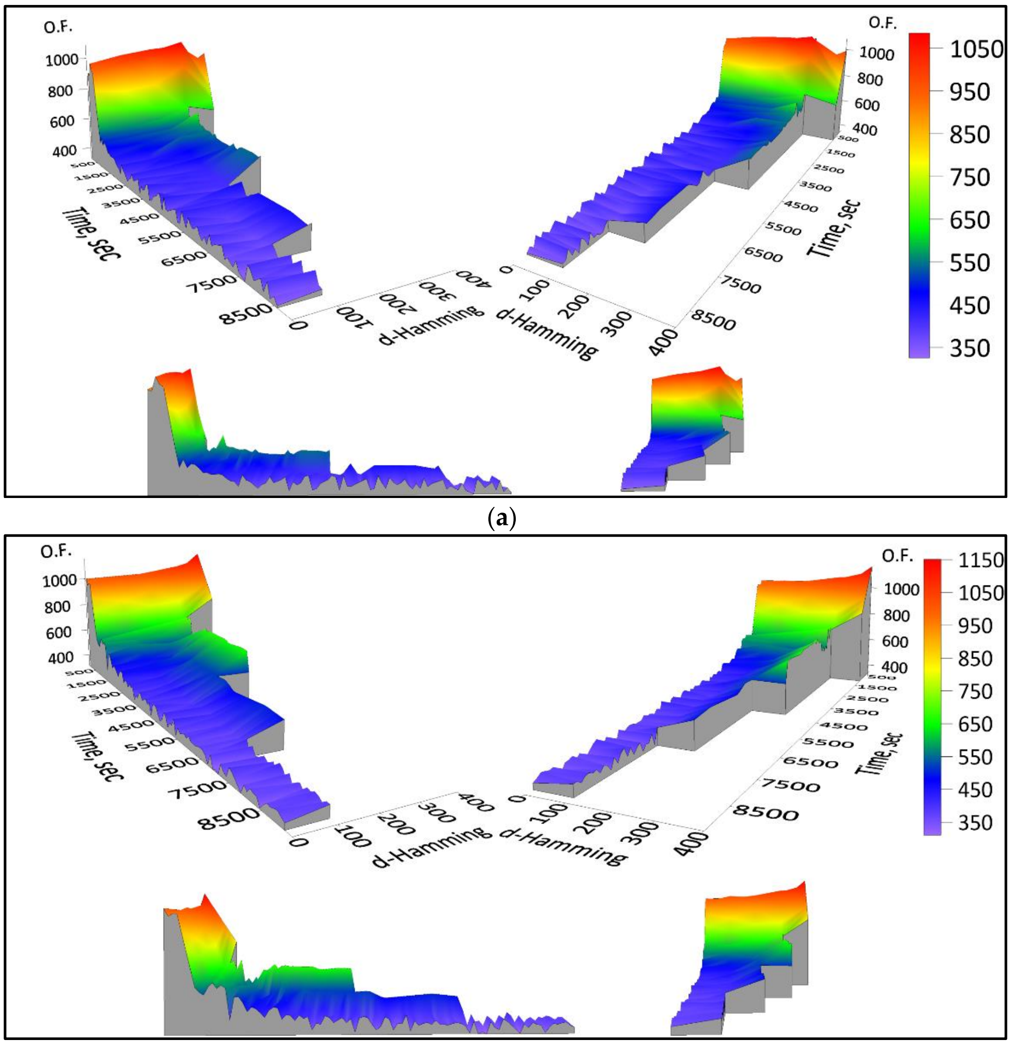
(**a**)



(**b**)

Figure 5. SACP with 48 processes generating the landscape without applying cooperation between the processes. Results of 30 tests. (**a**) Instance large01. (**b**) Instance large02.

Figure 6 presents the landscape for the instances large01 and large02 applying cooperation between the processes.



(**a**)

(**b**)

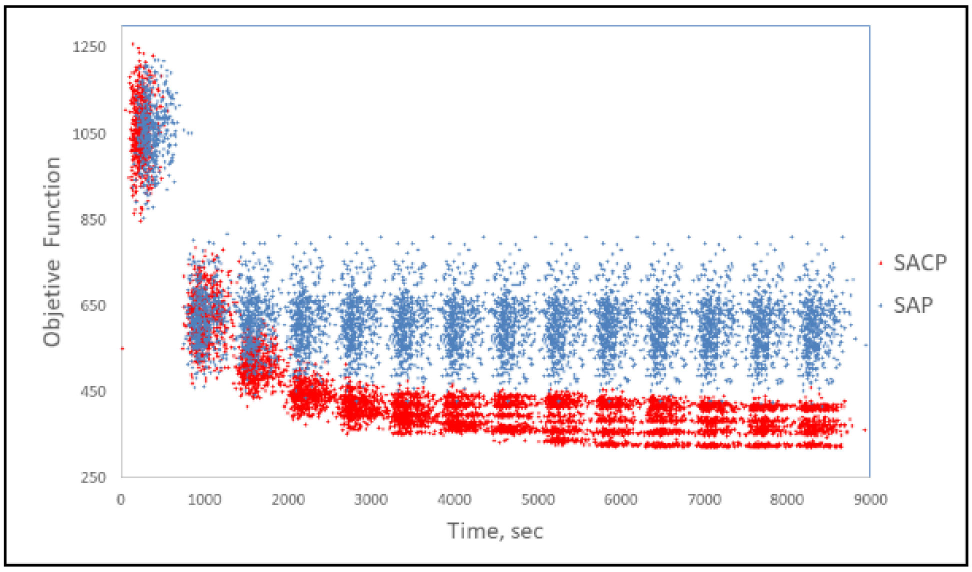
Figure 6. SACP with 48 processes generating the landscape applying cooperation between the processes. Results of 30 tests. (**a**) Instance large01. (**b**) Instance large02.

Figure 6a presents the landscape for instance large01 applying processes cooperation. It is observed that, with greater distance Hamming, the behavior of the solutions obtained as a function of time generates more clearly lower quality in the evaluation of the objective function with the Equation (1), unlike when there is no processes cooperation. In addition, at a shorter Hamming distance throughout the time interval, the behavior of the solutions obtained as a function of time generates the highest quality in the evaluation of the objective function. Moreover, if we compare Figure 6a,b, it can be observed that the value of the objective function has better quality throughout the interval of time from the application of the cooperation of processes that starts on the surface marked in blue (Figure 6b). It is observed that the best solutions obtained with cooperation are presented when the value of the Hamming distance is close to zero and in the SACP execution time limit.

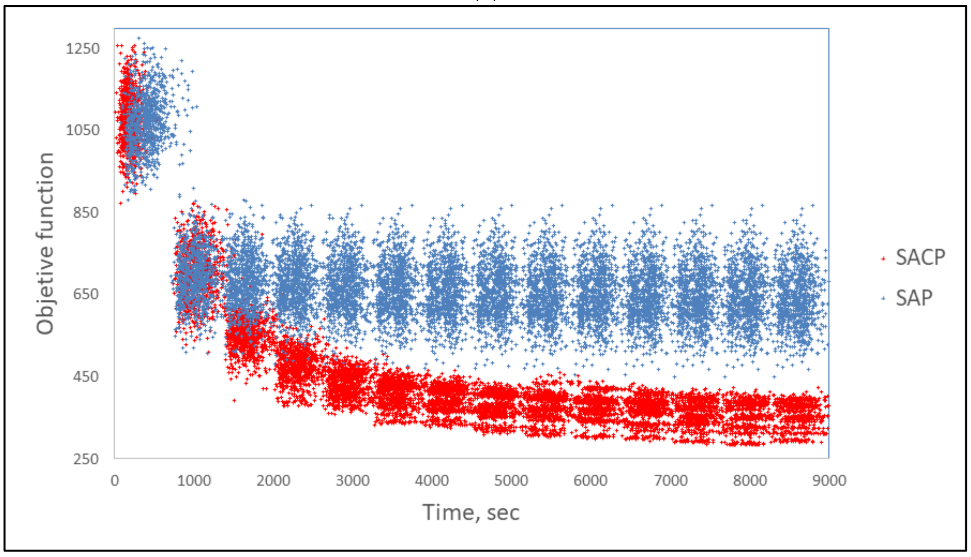
Figure 6b presents the landscape for the instance large02 applying cooperation among processes. It is observed that, with greater distance Hamming, the behavior of the solutions obtained as a function of time generates more clearly lower quality in the evaluation of the objective function with Equation (1), unlike when there is no processes cooperation. In addition, at a shorter Hamming distance throughout the time interval, the behavior of the solutions obtained as a function of time generates the highest quality in the evaluation of the objective function. It is observed that the best solutions obtained with cooperation are presented when the Hamming distance value is close to zero and in the SACP execution time limit. Moreover, when comparing Figures 5 and 6, it is observed that the value of the objective function is of better quality throughout the time interval from the application of the cooperation of processes that begins on a surface marked in blue (Figure 6).

Based on the analysis carried out of the landscape for large problems, it can be argued that addressing the solutions provided by each SACP process toward a closer appearance with the best solution obtained (the best of all obtained by all processes) on each SACP execution time, allows to considerably improve the final result obtained when cooperation amoung the processes is applied.

Figure 7 presents the effect of processes cooperation on time function for instances large01 and large02.



(**a**)



(**b**)

Figure 7. Processes with cooperation SACP and without cooperation SAP, in function of time with 48 processes. Results of 30 tests. (**a**) Instance large01. (**b**) Instance large02.

Figure 7a presents the effect of processes cooperation for the large01 problem, when SAP (simulated annealing with processes) without processes cooperation and when SACP with processes cooperation are applied. The graph presents infinity of points with blue and red colors. Each point represents a result obtained from some of the 48 processes in execution in SAP and SACP in the time interval of zero to 9000 s. It is observed that, at the beginning when the initial solution is generated where there is no cooperation in SACP, the results are similar as in SAP (0–700 s). When optimization starts with SA, both algorithms improve their solutions, but it is observed that as the execution time progresses, SACP tends to be better than SAP. Additionally, the SACP algorithm reaches an asymptotic convergence.

Figure 7b presents the effect of processes cooperation for the large02 problem, when SAP processes cooperation is not applied (simulated annealing with processes) and when SACP processes cooperation is applied. It is observed that, at the beginning when the initial solution is generated where there is no cooperation in SACP, the results are similar to SAP (0–700 s). When SA optimization is started, both algorithms improve their solutions, but it is observed that as the execution time progresses, SACP tends to be better than SAP and also presents an asymptotic convergence. In this graph, a greater separation is observed between the SAP vs. SACP solutions in Figure 7b. The latter indicates that the processes cooperation applied in SACP for the large02 problem works more efficiently by obtaining a greater difference in SACP vs. SAP results.

Figure 8 presents the behavior of how the value of the objective function is minimized vs. the Hamming distance for the instances large01 and large02.

In both instances in Figure 8a,b, the same behavior occurs, indicating that with a shorter Hamming distance, the result is better; this is observed in SACP. In the case of SAP, the result also improves without reducing the Hamming distance, but the result is of lower quality than in SACP. It is also observed that, for both instances large01 and large02, the best interval for the Hamming distance, in which the best solutions are found, is between 50 and 100.

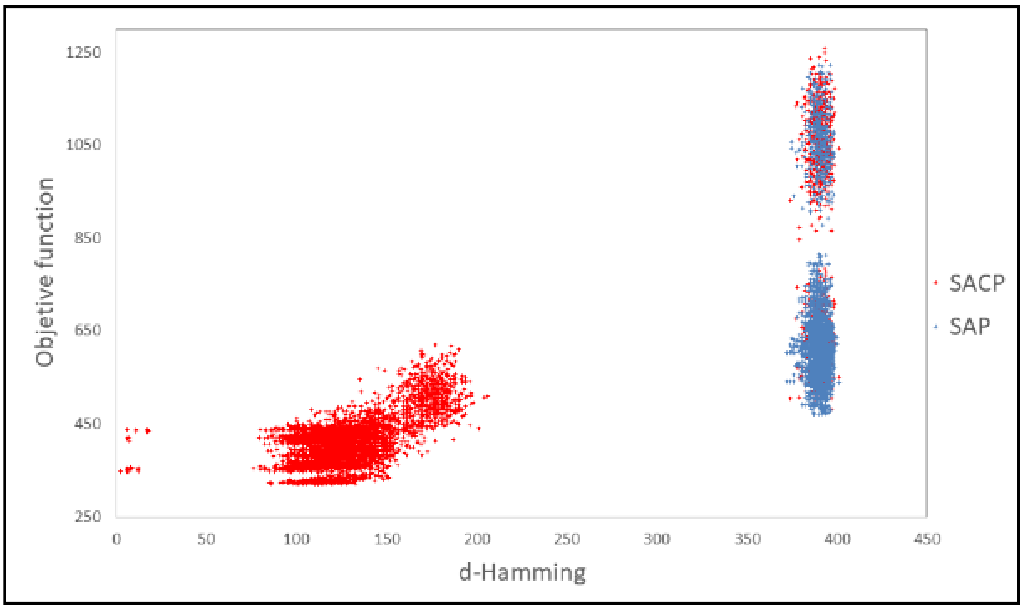
Figure 9 presents the value of the soft constraints and their Hamming distance for the best solution obtained by SACP for each of the 48 processes in execution, for the instances large01 and large02.

Figure 9a shows that, for the large01 instance, the lowest values are always obtained with the *Soft3* constraint in the 48 processes. Then, the values obtained with the *Soft1* constraint are placed as lowest values and, finally, the ones obtained with the *Soft2* constraint. This behavior is observed in all the processes. This indicates that SACP has more problems optimizing the *Soft2* constraint, and it is easier to optimize the *Soft3* constraint. The longest Hamming distance in a process does not exceed the value of 140.

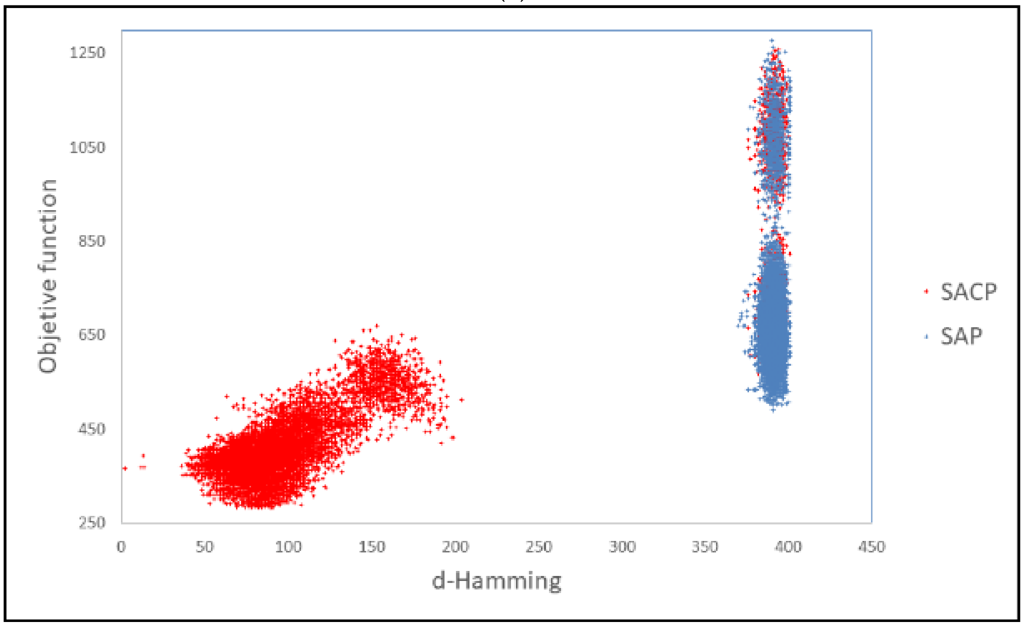
Figure 9b shows that, generally, the lowest value is for the Soft1 constraint in 43 of 48 processes, and the *Soft3* constraint follows and the *Soft2* constraint is at the end. This indicates that SACP has more problems optimizing the *Soft2* constraint, and it is easier to optimize the *Soft1* constraint. The longest distance Hamming does not exceed the value of 100. Optimizing soft constraints depends on the instance, because in large01 it is best optimized in the order of *Soft3* −> *Soft1* −> *Soft2*, while in large02 it is best optimized in the order of *Soft1* −> *Soft3* −> *Soft2*.

Table 4 presents the results obtained with SACP and SAP. Execution times and number of executions in SACP and SAP are the same. For each problem size, the times are different to be able to compare SACP (equal time in SAP) with other algorithms existing in the literature and using the same instances (ACO, GA, ILS, TS, SA).

For small instances, the number of executions for each instance is 100, and for medium and large instances, the number of executions for each instance is 30. For large problems, the execution time limit is 9000 s; for medium problems, the execution time limit is 4000 s; and for small problems, the recorded time is the time to obtain the global optimum. In all small instances, the global optimum is obtained with SACP. For small03, SACP obtained the optimum in 82 of 100 executions in the first iteration where processes cooperation is not yet applied, and for the remaining 18 executions, only a second iteration of SACP was required to obtain the global optimum. For small04, SACP obtained the optimum in 91 of 100 executions in its first iteration where processes cooperation is not yet applied, and for the remaining 9 executions, only a third iteration of SACP was required to obtain the global optimum. For the other small instances, SACP only required the execution of a single iteration to obtain the optimal solution; naturally the first execution of SACP does not apply processes cooperation. In the case of SAP, the global optimum is also obtained in all small instances, according to the mean, mode, median and standard deviation. The time is similar and does not reach more than 10 s in both cases (SACP vs. SAP).

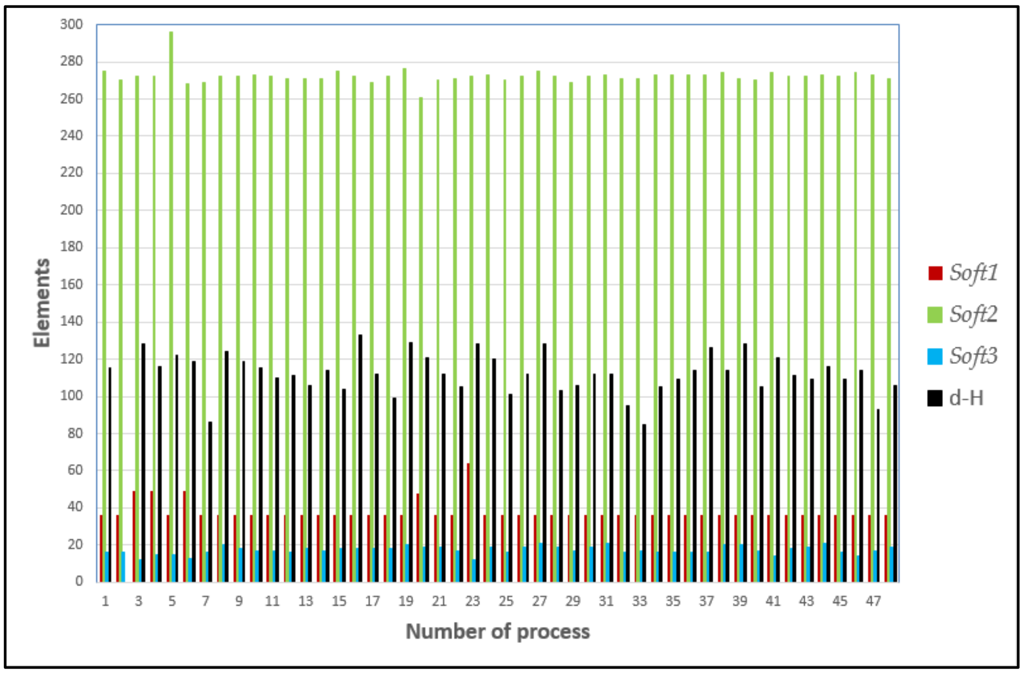


(**a**)

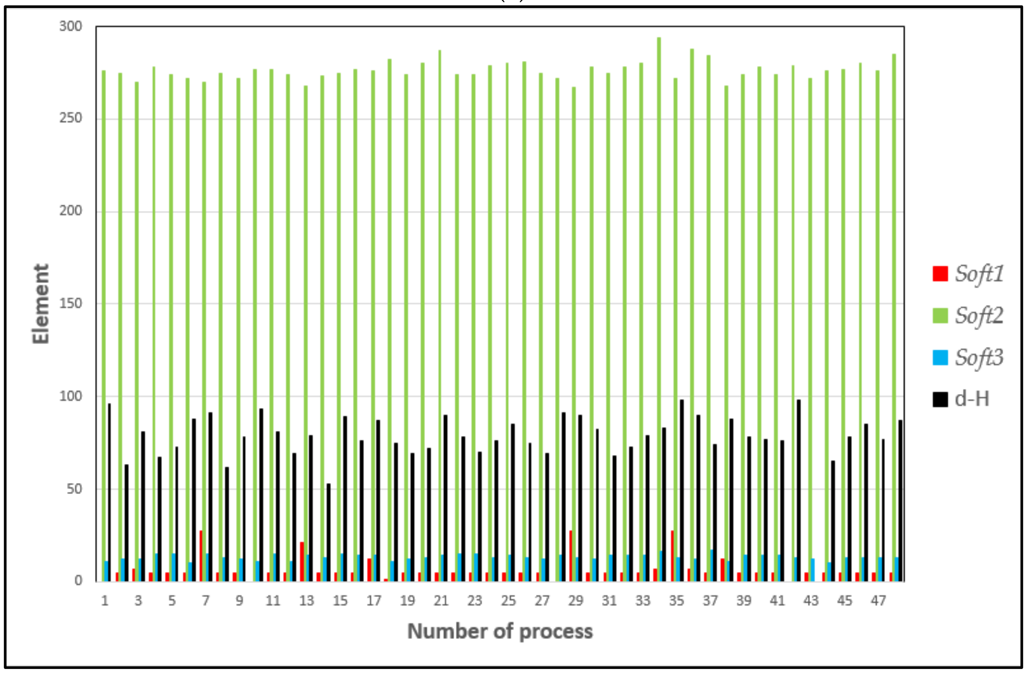


(**b**)

Figure 8. Processes with cooperation SACP and without cooperation SAP, in function of d-Hamming with 48 processes. Results of 30 tests. (**a**) Instance large01. (**b**) Instance large02.



(**a**)



(**b**)

Figure 9. Best solution obtained by SACP with 48 processes with the soft constraints. (**a**) Instance large01. (**b**) Instance large02.

In the case of medium and large instances, SACP is always better than SAP in obtaining the best solution. The best mean and median value is for SACP; in the case of mode, SAP is best only in medium02. In the standard deviation, SAP is best in all except medium02. The execution time is a little longer for SACP because cooperation between processes is applied in this algorithm, which is a little more time consuming, and in SAP there is no cooperation between the processes.

Table 4. Results of simulated annealing metaheuristic with cooperation of processes (SACP) and simulated annealing metaheuristic with processes (SAP) but without cooperation.

Problem	SACP						
	Min	Max	Mean	*t* s	σ	Mode	Median
Easy01	0	0	0	6	0	0	0
Easy02	0	0	0	8	0	0	0
Easy03	0	0	0	5	0	0	0
Easy04	0	0	0	6	0	0	0
Easy05	0	0	0	5	0	0	0
Medium01	44	60	50.69	4049.36	4.80	50	50
Medium02	48	56	51.5	4004.90	3.06	54	51
Medium03	57	77	57	3967.19	8.54	57	55
Medium04	47	62	51.7	3979.48	5.54	52	52
Medium05	7	20	11.8	3651.59	3.77	12	11.5
Large01	321	411	377.7	8354.97	32.51	407	379
Large 02	285	390	346.5	8656.48	32.93	390	358
Problem	**SAP**						
	Min	**Max**	**Mean**	***t* s**	**σ**	**Mode**	**Median**
Easy01	0	0	0	7	0	0	0
Easy02	0	0	0	8	0	0	0
Easy03	0	0	0	7	0	0	0
Easy04	0	0	0	5	0	0	0
Easy05	0	0	0	5	0	0	0
Medium01	53	60	55.8	3899.92	2.44	57	55.5
Medium02	48	62	52	3954.85	7.29	46	52.5
Medium03	58	65	60.9	3885.50	4.07	65	61
Medium04	48	59	53.6	3901.92	3.66	54	54
Medium05	16	25	21.8	3560.87	3.43	22	22
Large 01	410	504	478.22	8339.55	22.98	488	484
Large 02	450	528	500.33	8617.84	22.79	507	507

Table 5 presents a comparison of the best results obtained with SACP and other algorithms present in the literature. In the case of small problems, the best and worst result is presented in each algorithm. As can be seen for small01, in 100 executions, SACP always finds the global optimum (0–0) and, for example, in 100 executions, GA (0–19) finds the global optimum but not always, so the worst value found is 19. The value presented in the table in parentheses is the average rank and, as it can be seen, SACP always achieves first place in all the revised instances. In some cases, some algorithms presented in the literature do not find a solution for large instances, this is represented with * (ACO, GA and SA), which does not happen in SACP because it always finds solutions for any instance.

Table 5. Comparison of results obtained by the SACP metaheuristic with other results obtained by other metaheuristics at a time limit of 9000 s for large problems, 900 s for medium problems and 90 s for small problems.

Problem	**Metaheuristics**					
	SACP	**ACO**	**GA**	**ILS**	**TS**	**SA**
Easy01	0–0(1)	0–7(4)	0–19(5)	0–5(3)	0–34(6)	0–3(2)
Easy02	0–0(1)	0–15(4)	1–30(6)	0–10(3)	0–27(5)	0–6(2)
Easy03	0–0(1)	0–12(4)	0–22(5)	0–7(3)	0–29(6)	0–4(2)
Easy04	0–0(1)	0–7(4)	0–21(5)	0–6(3)	0–24(6)	0–5(2)
Easy05	0–0(1)	0–5(3)	0–13(5)	0–6(4)	0–21(6)	0–1(2)
Medium01	48(1)	182(6)	181(5)	120(3)	172(4)	71(2)
Medium02	49(1)	194(6)	170(4)	121(3)	183(5)	70(2)
Medium03	73(1)	252(6)	246(5)	171(3)	226(4)	111(2)
Medium04	49(1)	168(6)	148(4)	107(3)	163(5)	65(2)
Medium05	36(1)	176(4)	208(5)	142(3)	238(6)	55(2)
Large01	321(1)	* (6)	* (6)	900(2)	1170(3)	* (6)
Large02	285(1)	705(3)	1010(5)	690(2)	1000(4)	* (6)
Average ranking	1	4.7	5	2.9	5	1.8

4.2. Statistical Analysis

The test data are the values obtained by each algorithm (SACP, ACO, GA, ILS, TS, SA) with respect to its objective function (Table 5). For small instances, all algorithms find the global optimum solution, and for the statistical analysis purposes, the worst value obtained by each algorithm is used in order to know if there are differences in the effects of each type of algorithm in the worst of the results obtained. For the other medium and large instances, the statistical analysis uses the best value obtained in each algorithm.

The null hypothesis, H_0 in Equation (14), indicates that the means of the results are equal. The alternative hypothesis, Equation H_1 in (15), points out that the means are not equals or at least one is different. Statistical analysis tests whether the null hypothesis is true. If the null hypothesis is true, then there will be no difference in the effects of the analyzed algorithms.

$$H_0 : \overline{X_1} = \overline{X_2} = \ldots = \overline{X_r} \tag{14}$$

$$H_1 : \textit{Not all are the same} \tag{15}$$

For conducting the statistical analysis, both data normality and homoscedasticity are first verified. Figure 10 presents the normality of the data of each algorithm. The results show that there is no normality in the data of any algorithm since the points are not located on the diagonal of the graphs.

The homoscedasticity analysis for SACP, ILS and TS for the same number of data is performed graphically with the boxplot. Figure 11 shows that the boxes are not the same, since there is a greater dispersion of data in the TS box, and ILS follows and, finally, the least dispersion has SACP. TS presents the greatest variance and SACP is the one that has the smallest variance of the three algorithms. Therefore, a common variance cannot be admitted. It indicates that there is no homoscedasticity between the three algorithms. Applying the Bartlett test with the Bartlett-test function of R [44], the p-value = 0.001169 is found (0.001169 < 0.05), this shows that there are different variances; therefore, it is confirmed that there is no homoscedasticity when having a small p-value.

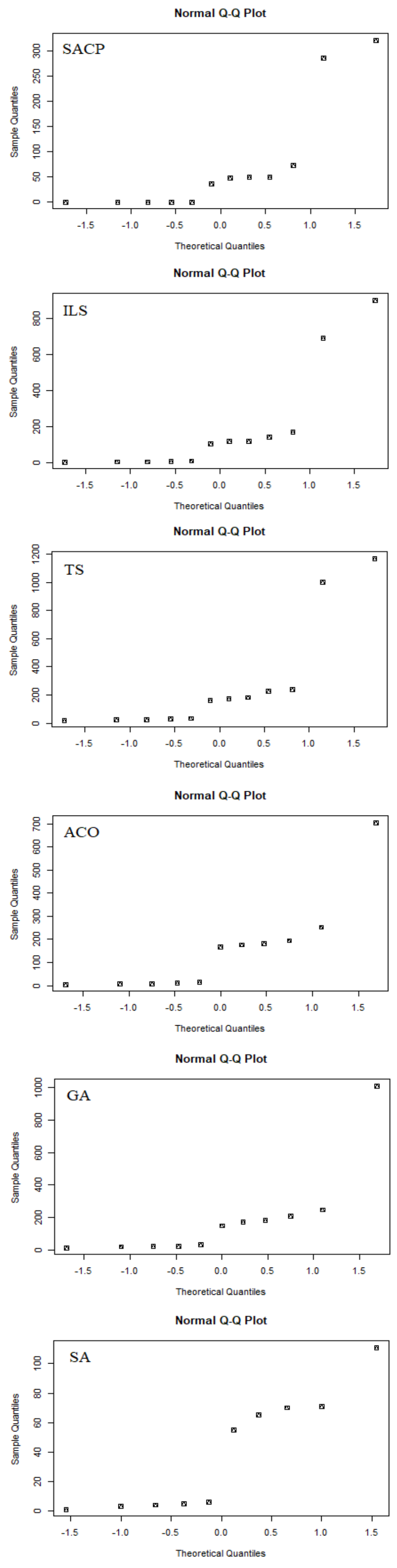


Figure 10. Normality graphs for SACP, ACO, GA, ILS, TS and SA algorithms.

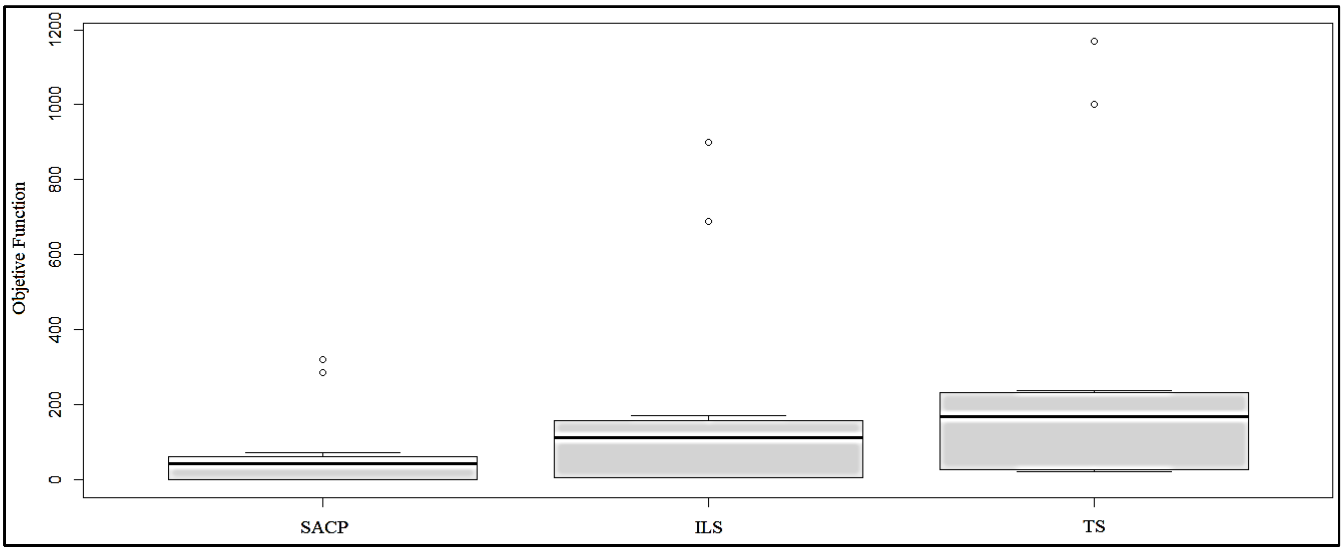


Figure 11. Boxplot of the SACP, ILS and TS algorithms.

The homoscedasticity analysis for SACP, ACO and GA for the same number of data is performed graphically with the boxplot. Figure 12 shows that the boxes are not the same, since a greater dispersion of data is observed in the ACO and GA boxes and the lower dispersion is presented in SACP. SACP is the algorithm that has the least variance. Therefore, a common variance cannot be admitted, which indicates that there is no homoscedasticity. Applying the Bartlett test with the Bartlett-test function of R [44], the *p*-value = 0.002409 is found (0.002409 < 0.05), this shows that there are different variances; therefore, there is no homoscedasticity when having a small *p*-value.

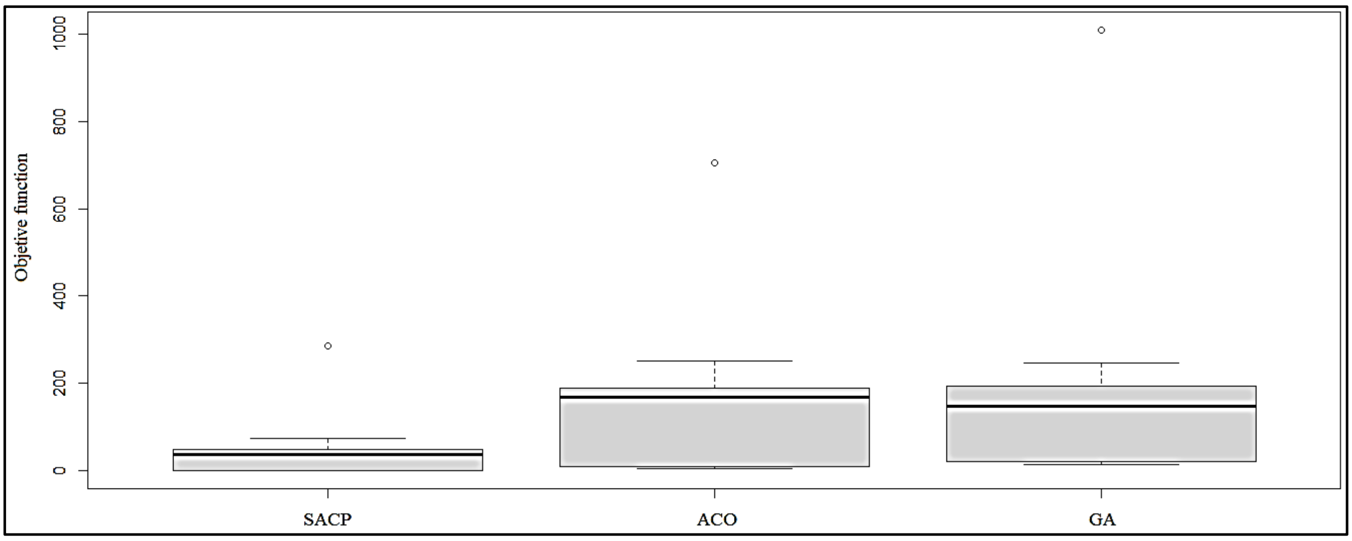


Figure 12. Boxplot of the SACP, ACO and GA algorithms.

The homoscedasticity analysis for SACP and SA for the same number of data is performed graphically with the boxplot. Figure 13 shows that the boxes are not the same since a greater dispersion of data is observed in the SA box and the lower dispersion is presented by SACP. The greatest variance is observed in SA, while SACP has the smallest variance. So, a common variance cannot be admitted so there is no homoscedasticity. Homoscedasticity is analyzed with the Bartlett test, applying the Bartlett-test function of R [44]. The *p*-value 0.3233 > 0.05 indicates that there are very similar variances; therefore, the Bartlett test indicates that there is homoscedasticity when having a *p*-value that exceeds the significance level of 0.05.

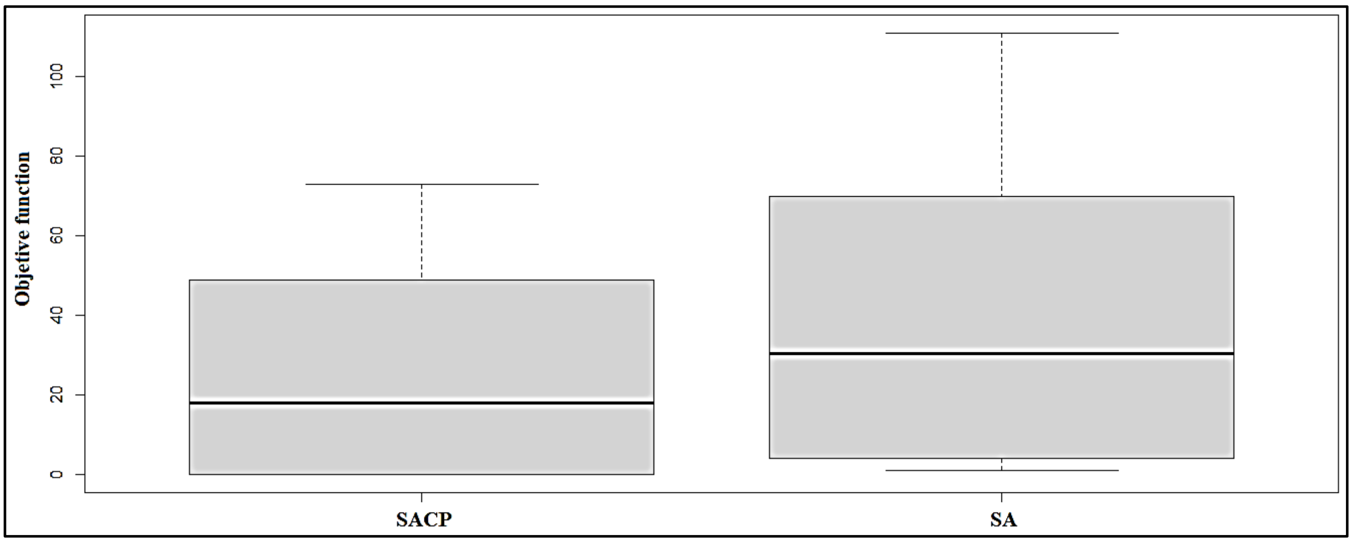


Figure 13. Boxplot of the SACP and SA algorithms.

In order to know if there are significant differences in the behavior of the objetive function values obtained by the algorithms, it is required to analyze the results obtained by the six algorithms. As there are not both assumptions of normality and homoscedasticity of the data provided by the six algorithms, and also because the number of data analyzed by each algorithm is small, Kruskal–Wallis non-parametric method, proposed by William Henry Kruskal and Whallen Wallis [45], will be applied. The Kruskal–Wallis test requires that the observations of each algorithm, which must be different, be presented in ranges. The Equation (16) presents the Kruskal–Wallis test. If there are observations with equal values, then Equation (16) is divided by the relation (17), where H is the value of the contrast statistic, C is the number of the algorithms to analyze (in this case there are six algorithms), n_i is the number of observations in the *ith* algorithm (number of results of each algorithm), $N = \sum n_i$ is the total number of results of all algorithms and R_i is the sum of the ranges in the *ith* algorithm.

$$H = \frac{12}{N(N+1)} \sum_{i=1}^{C} \frac{R_i^2}{n_i} - 3(N+1) \tag{16}$$

In relation 17, the sum is over all groups of repeated data and $T = t^3 - t$ is for each group of repeated data. Where t is the number of repeated results in the group.

$$1 - \frac{\sum T}{N^3 - N} \tag{17}$$

From Table 5, the first ten data, which are the results for small and medium problems, are taken from each algorithm in order to make a comparison of all the algorithms. The Kruskal–Wallis test is applied with the Kruskal-test function of R [44] to obtain the p-value. The contrast statistic H is 26.304 and the *p*-value is $7.79 \times 10^{-5} < 0.05$; therefore, the null hypothesis of equality of the effects of algorithms is rejected. This indicates that there are differences in the effects of algorithm types by having a *p*-value lower than their significance level.

Post Hoc Comparisons

Since the Kruskal–Wallis test is significant, it implies that at least two groups of algorithms from among those compared are significantly different, but does not indicate which ones. To find out, it is necessary to make a comparison between them. Comparison is made with the results obtained from SACP metaheuristic vs. results obtained by each metaheuristic presented in Table 5 (metaheuristic 1 vs. metaheuristic 2) as follows SACP vs. ACO, SACP vs. GA, SACP vs. ILS, SACP vs. TS and SACP vs. SA, with the Mann Whitney Wilcoxon method [46] represented by Equations (18)–(20), where n_1 is the number of results obtained by metaheuristic 1, n_2 is the number of results obtained of metaheuristic 2, R_1 is the sum of the ranges in metaheuristic 2.

$$H = \min(H_1, H_2) \tag{18}$$

$$H_1 = n_1 n_2 + \frac{n_1(n_1+1)}{2} - R_1 \tag{19}$$

$$H_2 = n_1 n_2 + \frac{n_2(n_2+1)}{2} - R_2 \tag{20}$$

The Mann Whitney Wilcoxon test is applied between each pair of algorithms with the Wilcox-test function of R [44] with the first 5 data from each algorithm to obtain the p-value in small problems. Table 6 presents the results of the Mann Whitney Wilcoxon test. It is observed in the SACP comparison with each of the other algorithms that there are differences in the effects between each pair of algorithms compared because the p-value does not exceed the significance level which is 0.05. Moreover, in Table 6, it is observed that SACP has the best average rank and TS the worst.

Table 6. Post hoc test and average ranking for multiple p-values comparisons between SACP vs. other algorithms taking only data from small instances.

Algorithm	Average Rank	SACP p-Values
		Post Hoc Wilcoxon–Mann–Whitney
SACP	1	-
SA	2	0.005346
ILS	3.2	0.005189
ACO	3.8	0.005189
GA	5.2	0.005346
TS	5.8	0.005346

Again, the Mann Whitney Wilcoxon test is applied between each pair of algorithms with the Wilcox-test function of R [44] with the following 5 data from each algorithm to obtain the p-value in medium problems. Table 7 presents the results of the Mann Whitney Wilcoxon test. It is observed in the comparison of SACP with each of the other algorithms that there are differences in the effects between each pair of algorithms compared because the p-value does not exceed the level of significance that is 0.05. Moreover, in Table 7, it is observed that SACP has the best average rank and ACO the worst. In the case of p-value with SA, it exceeds the significance value of 0.05; however, SACP has a better average rank and also SA does not obtain results for any of the large problems. In the case of SACP, in all the executions carried out of the algorithm, feasible results are obtained for the large problems.

Table 7. Post hoc test and average ranking for multiple *p*-values comparisons between SACP vs. other algorithms taken only data from medium instances.

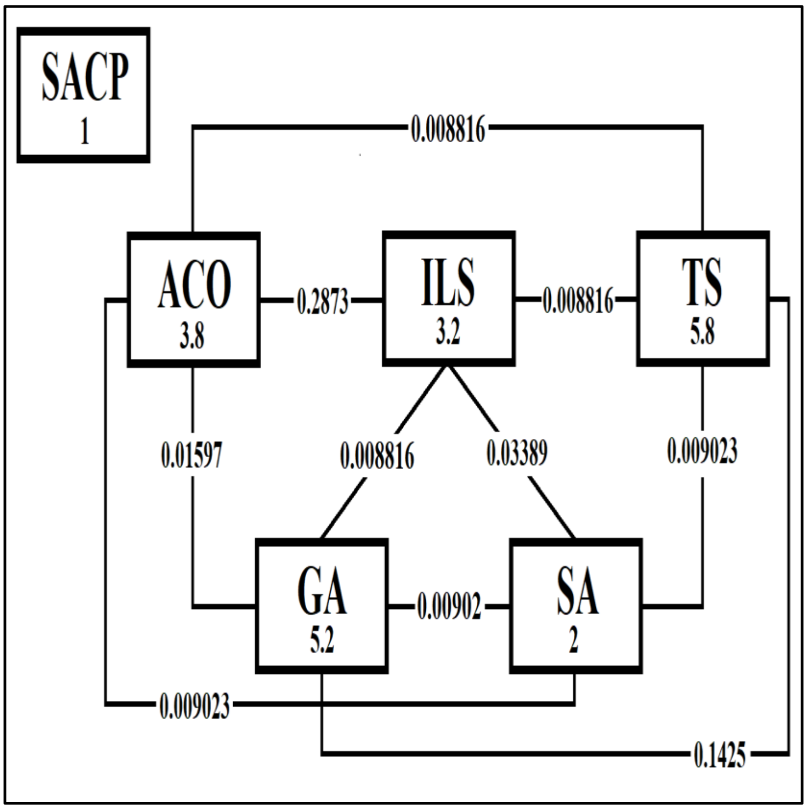
Algorithm	**Average Rank**	**SACP *p*-Values**
		Post Hoc Wilcoxon–Mann–Whitney
SACP	1	-
SA	2	0.074910
ILS	3	0.008816
GA	4.6	0.008816
TS	4.8	0.008816
ACO	5.6	0.008816

Figure 14 presents the *p*-values in pairs of algorithms that were compared with SACP for small and medium instances. Figure 14a presents the *p*-values between a pair of algorithms that were compared with SACP for small instances. It is observed that among eight pairs of them, there is also a difference in the effects of the types of algorithms by having a lower *p*-value at its significance level, such as ACO vs. TS, ACO vs. SA, ACO vs. GA, ILS vs. TS, ILS vs. SA, ILS. However, also between two pairs of them, there are no differences in the effects of the types of algorithms, as these have a *p*-value greater than their level of significance, such as ACO vs. ILS with a *p*-value of 0.2873 and GA vs. TS with a *p*-value of 0.1425.

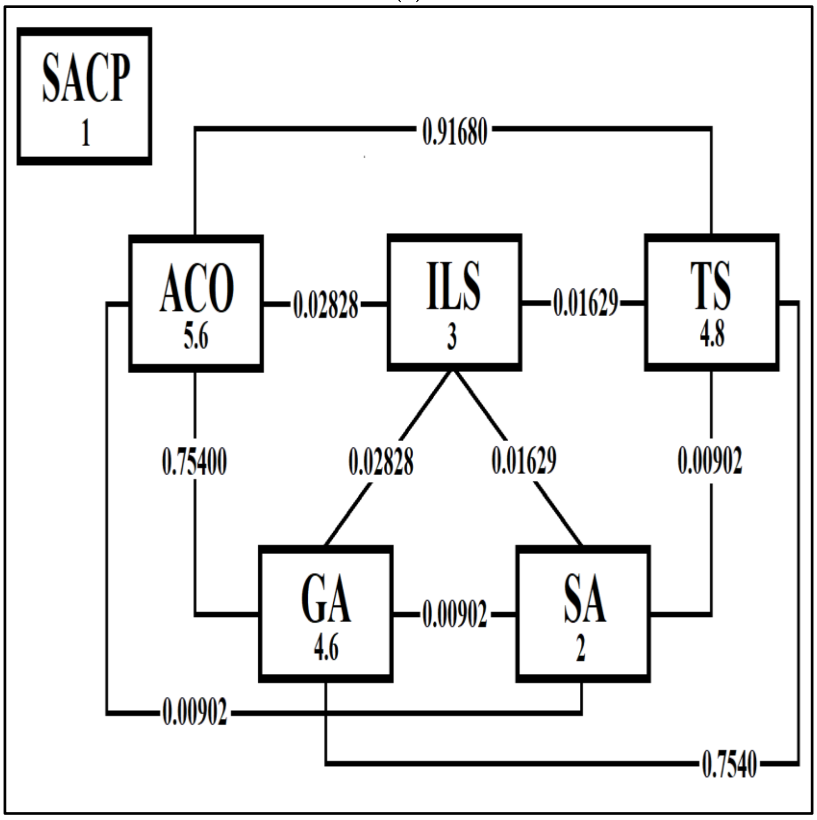
The average rank is presented in the box for each algorithm, which classifies the best algorithm in terms of its results, with SACP being the best average rank. Figure 14b presents the *p*-values between a pair of algorithms that were compared with SACP for medium instances. It is observed that, among seven pairs of them, there are also differences in the effects of algorithm types by having a *p*-value lower at its significance level, such as ACO vs. SA, ACO vs. ILS, ILS vs. TS, ILS vs. GA, ILS vs. SA. However, also between three pairs of them, there are no differences in the effects of the types of algorithms as they have a *p*-value greater than their level of significance, such as ACO vs. TS with a *p*-value of 0.754, ACO vs. GA with a *p*-value of 0.754 and GA vs. TS with a *p*-value of 0.754. The average rank is presented in the box for each algorithm, which classifies the best algorithm in terms of its results, with SACP being the best average rank.

4.3. SACP Efficiency

Figure 15 presents the efficiency of SACP evaluated for large instances. It can be seen that, up to 21 processes, efficiency presents a superlinear speedup above the ideal. This is because the first 12 processes in SACP are executed on a single node of the computational cluster and does not require communication with the other three nodes of the cluster. From 21 processes onwards, efficiency begins to decrease considerably. This is due to two things. First of all, that communication between nodes is performed with an Ethernet 100BASE-TX, 100 Mbs connection, which slows down the sending of messages from SACP between nodes. Second, that SACP carries out cooperation between processes, which implies that in each iteration there is always communication between all the processes that are executed at all nodes of the cluster, applying collective communication and point-to-point communication.



(**a**)



(**b**)

Figure 14. Graph of the *p*-values and average ranking between the compared algorithms. (**a**) Small instances. (**b**) Medium instances.

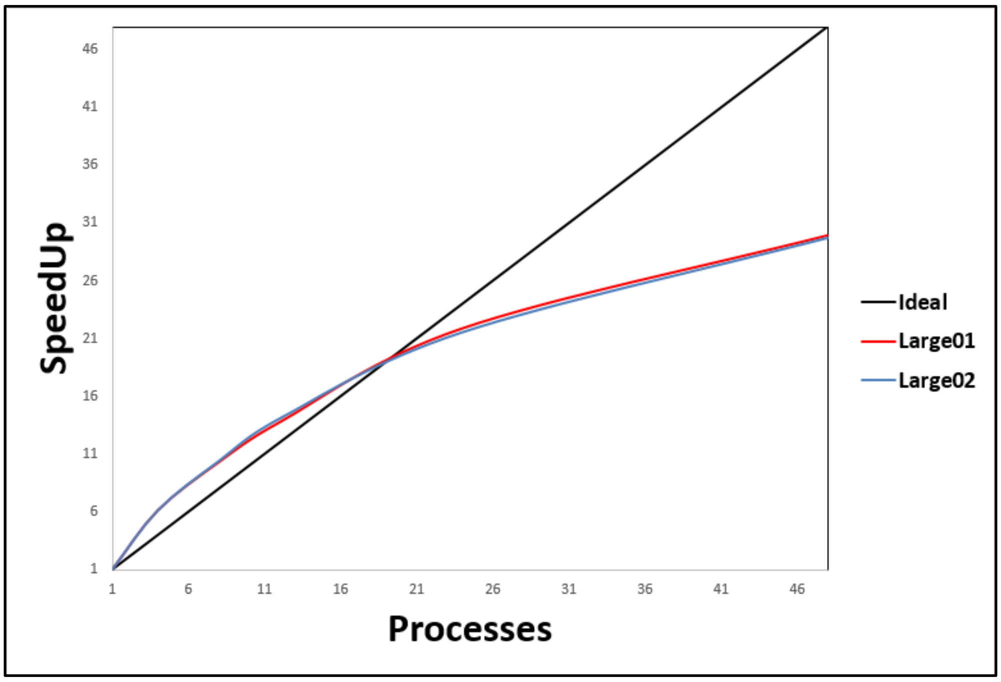


Figure 15. SACP speedup for large UCTP problems.

5. Conclusions

It is concluded that applying processes cooperation in SACP, through addressing of the solutions of each process using the Hamming distance in each iteration of the algorithm, allows improving the final result obtained.

Optimizing values in the soft constraints with SACP for large problems has a different complexity since the degree of difficulty in optimization increases in the order of Soft3 −> Soft1 −> Soft2 for large01 and in the order of Soft1 −> Soft3 −> Soft2 for large02.

The efficacy of SACP is the highest compared to five heuristics proposals found in the literature, ACO, GA ILS, TS and SA. SACP always achieves first place in all revised instances (average ranking). Furthermore, in some cases, some algorithms in the literature do not find a solution for large instances (ACO, GA and SA), which does not happen in SACP because it always finds solutions for any instance.

Statistical analysis shows that, in the SACP comparison with each of the other algorithms, there are differences in the effects of SACP with the other algorithms compared because the p-value does not exceed the significance level of 0.05. In the case of the p-value obtained with respect to SA, it exceeds the significance value of 0.05 for very little. Added to this, SACP has a better average rank than SA and SA also does not obtain results for any of the large problems. In the case of SACP, in all the executions carried out by the algorithm, feasible results are obtained for the large problems. SACP efficiency is very good when using a single node in the cluster, but it is quickly reduced by having to use more than one node in the cluster, further added to the communication between processes that each iteration of the algorithm requires to apply processes cooperation.

The limitations found for this work were that no more test instances were found for the presented optimization model with which more results could be presented, nor were other works found that will present other optimization methods to compare the efficacy of SACP.

In future works, the aim will be to improve the speedup by applying better communication between cluster nodes with the infinband network. A better and more efficient compiler such as INTEL will also be used instead of the free GNU compiler.

Some future needs are the elaboration of new test instances of the optimization model and the development of heuristics of populations that present results for the optimization model presented in this work with which more robust comparative tests can be carried out.

Author Contributions: Conceptualization, M.H.C.-R. and M.A.C.-C.; methodology, F.A.-P. and J.d.C.P.-A.; software, M.A.C.-C.; validation, M.H.C.-R., E.Y.Á.-M. and B.M.-B.; formal analysis, J.E.-U.; investigation, B.M.-B. and J.E.-U.; writing—original draft preparation, M.A.C.-C. and E.Y.Á.-M.; writing—E.Y.Á.-M. and J.d.C.P.-A.; supervision, F.A.-P.; project administration, M.A.C.-C.; funding acquisition, M.A.C.-C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by CONACYT grant number 317029.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ben-Ayed, O.; Hamzaoui, S. Multiobjective multiproduct parcel distribution timetabling: A real-world application. *Int. Trans. Oper. Res.* **2012**, *19*, 613–629. [CrossRef]
2. Shen, Y.; Xu, J.; Zeng, Z. Public Transit planning and scheduling based on AVL data in China. *Int. Trans. Oper. Res.* **2015**, *23*, 1089–1111. [CrossRef]
3. Kiris, S. AHP and multichoice goal programming integration for course planning. *Int. Trans. Oper. Res.* **2014**, *21*, 819–833. [CrossRef]
4. Garey, M.R.; Johnson, D.S. *Computer and Intractability: A Guide to the Theory of NP-Completeness*; W. H. Freeman & Company: New York, NY, USA, 1990; ISBN-10: 0716710455, ISBN-13: 978-0716710455.
5. Papadimitriou, C.H.; Steiglitz, K. *Combinatorial Optimization Algorithms and Complexity*; Dover Publication, Inc.: New York, NY, USA, 2013; p. 530, ISBN-10: 0486402584.
6. Zhao, H.; Zhang, C. An Online-Learning-Based Evolutionary Many-Objective Algorithm. *Inf. Sci.* **2020**, *509*, 1–21. [CrossRef]
7. Dulebenets, M.A. An Adaptive Polyploid Memetic Algorithm for Scheduling Trucks at a Cross-Docking Terminal. *Inf. Sci.* **2021**, *565*, 390–421. [CrossRef]
8. Liu, Z.-Z.; Wang, Y.; Huang, P.-Q. AnD: A Many-Objective Evolutionary Algorithm with Angle-based Selection and Shift-based Density Estimation. *Inf. Sci.* **2020**, *509*, 400–419. [CrossRef]
9. Theophilus, O.; Dulebenets, M.A.; Pasha, J.; Lau, Y.-Y.; Fathollahi-Fard, A.M.; Mazaheri, A. Truck scheduling optimization at a cold-chain cross-docking terminal with product perishability considerations. *Comput. Ind. Eng.* **2021**, *156*, 107240. [CrossRef]
10. Pasha, J.; Dulebenets, M.A.; Kavoosi, M.; Abioye, O.F.; Wang, H.; Guo, W. An Optimization Modelo and Solution Algorithms for the Vehicle Routing Problem with a "Factory-in-a-Box". *IEEE Access* **2020**, *8*, 134743–134763. [CrossRef]
11. D'Angelo, G.; Pilla, R.; Tascini, C.; Rampone, S. A Proposal for Distinguishing between Bacterial and Viral Meningitis Using Genetic Programming and Decision Trees. *Methodol. Appl.* **2019**, *23*, 11775–11791. [CrossRef]
12. Burke, E.; Carter, M. (Eds.) *The Practice and Theory of Automated Timetabling: Selected Papers from the Second International Conference, Lectures Notes in Computer Science 1408*; Springer: Berlin/Heidelberg, Germany, 1997.
13. Schaerf, A. A survey of automated timetabling. *Artif. Intell. Rev.* **1999**, *13*, 87–127. [CrossRef]
14. Appleby, J.S.; Blake, D.V.; Newman, E.A. Techniques for reducing school timetables on a computer and their application to other scheduling problems. *Comput. J.* **1960**, *3*, 237–245. [CrossRef]
15. Smith, G. On maintenance of the opportunity list for class-teacher timetable problems. *Commun. ACM* **1975**, *18*, 203–208. [CrossRef]
16. De Werra, D. An introduction to timetabling. *Eur. J. Oper. Res.* **1985**, *19*, 151–162. [CrossRef]
17. De Werra, D.; Pasche, C.; Petter, A. Time-tabling problems: Should they be canonical? *INFOR Inf. Syst. Oper. Res.* **1986**, *14*, 304–308. [CrossRef]
18. Cheng, E.; Kruk, S.; Lipman, M. On the multicommodity flow formulations for the student scheduling problem. *Congr. Numer.* **2003**, *160*, 177–181.
19. Brelaz, D. New methods to color the vertices of a graph. *Commun. ACM* **1979**, *22*, 251–256. [CrossRef]
20. Burke, E.; Elliman, D.; Weare, R. A university timetabling system based on graph colouring and constraint manipulation. *J. Res. Comput. Educ.* **1994**, *27*, 1–18. [CrossRef]
21. Faber, W.; Leone, N.; Pfeifer, G. Representing school timetabling in a disjunctive logic programming language. In Proceedings of the 13th Workshop on LOgic Programming (WLP'98), Karlsplatz, Vienna, 6–8 October 1998.
22. Elmohamed, S.; Fox, G. A comparison of annealing techniques for academic course scheduling. In *Second International Conference on Practice and Theory of Automated Timetabling II*; Lecture Notes in Computer Science; Burke, E., Carter, M., Eds.; Springer: Berlin/Heidelberg, Germany, 1997; pp. 92–114.
23. Abramson, D. Constructing school timetables using simulated annealing: Sequential and parallel algorithms. *Manag. Sci.* **2001**, *37*, 98–113. [CrossRef]
24. Abramson, D.; Abela, J. *A Parallel Genetic Algorithm for Solving the School Timetabling Problem*; Technical Report; Division of Information Technology, C.S.I.R.O. University of Edinburg: Carlton, Australia, 1992.
25. HHertz, A. Tabu search for large scale timetabling problems. *Eur. J. Oper. Res.* **1991**, *54*, 39–47. [CrossRef]

26. Rossi-Doria, O.; Sampels, M.; Chiarandini, M.; Knowles, J.; Manfrin, M.; Mastrolilli, M.; Paquete, L.; Paechter, B. *A Comparison of the Performance of Different Metaheuristics on the Timetabling Problem*; Practice and Theory of Automated Timetabling IV (PATAT'02, Selected Papers), Lecture Notes in Computer Science, 2740; Burke, E., De Causmaecker, P., Eds.; Springer: Berlin/Heidelberg, Germany, 2003; pp. 329–351.
27. Thepphakorn, T.; Pongcharoen, P.; Hicks, C. Modifying regeneration mutation and hybridising clonal selection for evolutionary algorithms based timetabling tool. *Math. Probl. Eng.* **2015**, *2015*, 841748. [CrossRef]
28. Yu, E.; Sung, K.-S. A genetic algorithm for a university weekly courses timetabling problem. *Int. Trans. Oper. Res.* **2002**, *9*, 703–717. [CrossRef]
29. Zervoudakis, K.; Stamatopoulos, P. *A Generic Object-Oriented Constraint-Based Model for University Course Timetabling*; PATAT 2000, LNCS 2079; Burke, E., Erben, W., Eds.; Springer: Berlin/Heidelberg, Germany, 2001; pp. 28–47. [CrossRef]
30. Burke, E.; Kendall, G.; Soubeiga, E. A Tabu-Search Hyperheuristic for Timetabling and Rostering. *J. Heuristics* **2003**, *9*, 451–470. [CrossRef]
31. Kostuch, P. *The University Course Timetabling Problem with a Three-Phase Approach*; PATAT 2004, LNCS 3616; Burke, E., Trick, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; pp. 109–125.
32. Blum, C.; Roli, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.* **2003**, *35*, 268–308. [CrossRef]
33. Cruz-Chávez, M.A.; Flores-Pichardo, M.; Martínez Oropeza, A.; Moreno-Bernal, P.; Cruz-Rosales, M.H. *Solving a Real Constraint Satisfaction Model for the University Course Timetabling Problem: A Case Study, Mathematical Problems in Engineering*; Hindawi Publishing Corp.: London, UK, 2016; pp. 1–14, ISSN 1024-123X. [CrossRef]
34. Chávez-Bosquez, O.; Hernández-Torruco, J.; Hernández-Ocaña, B.; Canul-Reich, J. Modeling and Solving a Latin American Univesity Course Timetabling Problem Instance. *Mathematics* **2020**, *8*, 1833. [CrossRef]
35. Arratia-Martinez, N.M.; Maya-Padron, C.; Avila-Torres, P. University Course Timetabling Problem with Professor Assignment. *Math. Probl. Eng.* **2021**, 6617177. [CrossRef]
36. Banczyk, K.; Boinski, T.; Krawczyk, H. Parallelisation of Genetic Algorithms for Solving University Timetabling Problems. In Proceedings of the International Symposium on Parallel Computing in Electrical Engineering (PARELEC'06), Bialystok, Poland, 13–17 September 2006; IEEE: New York, NY, USA; pp. 325–330. [CrossRef]
37. Abdelhali, E.A.; El Khayat, G.A. A Utilization-based Genetic Algorithm for Solving the University Timetabling Problem (UGA). *Alex. Eng. J.* **2016**, *55*, 1395–1409. [CrossRef]
38. Xuehao, F.; Yuna, L.; Ilkyeong, M. An integer program and a hybrid genetic algorithm for the university timetabling problem. *Optim. Methods Softw.* **2017**, *32*, 625–649. [CrossRef]
39. Rezaeipanah, A.; Abshirini, Z.; Boshkani Zade, M. Solving Univesity Course Timetabling Problem Using Parallel Genetic Algorithm. *Int. J. Sci. Res. Comput. Sci. Eng.* **2019**, *7*, 5–13. Available online: https://www.isroset.org (accessed on 12 November 2020).
40. Arias-Osorio, J.; Mora-Esquivel, A. A solution to the university course timetabling problem using a hybrid method based on genetic algorithms. *DYNA* **2020**, *87*, 47–56. [CrossRef]
41. Chiarandini, M.; Stützle, T. *Experimental Evaluation of Course Timetabling Algorithms*; Technical Report; FG Intellektik, TU Darmstadt: Darmstadt, Germany, 2002.
42. Rossi-Doria, O.; Sampels, M.; Birattari, M.; Chiarandini, M.; Dorigo, M.; Gambardella, L.M.; Knowles, J.; Manfrin, M.; Mastrolill, M.; Paechter, B. Supporting Material for the Paper: A Comparison of the Performance of Different Metaheuristics on the Timetabling Problem. Available online: http://iridia.ulb.ac.be/supp/IridiaSupp2002-001/ (accessed on 27 September 2021).
43. Romeo, F.; Sangiovanni-Vincentelli, A. A theoretical framework for simulated annealing. *Algorithmica* **1991**, *6*, 302. [CrossRef]
44. R Version 4.0.5. The Foundation for Statistical Computing. 31 March 2021. Available online: https://www.r-project.org/ (accessed on 7 May 2021).
45. Kruskal, W.H.; Wallis, W.A. Use of Ranks in One-Criterion Variance Analysis. *J. Am. Stat. Assoc.* **1952**, *47*, 583–621. [CrossRef]
46. Mann, H.B.; Whitney, D.R. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other, The Annals of Mathematical Statistics. *Inst. Math. Stat.* **1947**, *18*, 50–60. [CrossRef]