Received: 31 May 2023 | Revised: 3 November 2023 | Accepted: 13 January 2024

DOI: 10.1111/exsy.13554

ORIGINAL ARTICLE

Expert Systems WILEY

Improved cooperative Ant Colony Optimization for the solution of binary combinatorial optimization applications

Roberto Prado-Rodríguez[1] | Patricia González[1] | Julio R. Banga[2] | Ramón Doallo[1]

[1]CITIC, Computer Arquitecture Group, University of A Coruña, A Coruña, Spain

[2]Computational Biology Lab, MBG-CSIC, Spanish National Research Council, Pontevedra, Spain

Correspondence
Roberto Prado-Rodríguez, CITIC, Computer Arquitecture Group, University of A Coruña, A Coruña, Spain.
Email: roberto.prado@udc.es

Funding information
Ministerio de Ciencia e Innovación; European Regional Development Fund; Xunta de Galicia

Abstract

Binary combinatorial optimization plays a crucial role in various scientific and engineering fields. While deterministic approaches have traditionally been used to solve these problems, stochastic methods, particularly metaheuristics, have gained popularity in recent years for efficiently handling large problem instances. Ant Colony Optimization (ACO) is among the most successful metaheuristics and is frequently employed in non-binary combinatorial problems due to its adaptability. Although for binary combinatorial problems ACO can suffer from issues such as rapid convergence to local minima, its eminently parallel structure means that it can be exploited to solve large and complex problems also in this field. In order to provide a versatile ACO implementation that achieves competitive results across a wide range of binary combinatorial optimization problems, we introduce a parallel multicolony strategy with an improved cooperation scheme to maintain search diversity. We evaluate our proposal (Binary Parallel Cooperative ACO, BiPCACO) using a comprehensive benchmark framework, showcasing its performance and, most importantly, its flexibility as a successful all-purpose solver for binary combinatorial problems.

KEYWORDS
Ant Colony Optimization, binary combinatorial optimization, metaheuristic, parallel strategies

1 | INTRODUCTION

To optimize means to find the best solution among several conflicting demands subject to predefined requirements. The key elements of these problems are the decision variables, the objective function, and the constraints that must be met. From the whole space solution, those that satisfy the constraints form the set of feasible solutions. If the set of solutions is finite, problems belongs to combinatorial optimization. Specifically, if the decision variables are boolean, we will refer to them as binary combinatorial problems.

Binary combinatorial optimization problems appear in numerous application areas, such as feature selection (Li et al., 2017), dimensionality reduction (Carreira-Perpinán, 1997; Pal & Maiti, 2010), unit commitment (Baldick, 1995; Yuan et al., 2009), manufacturing (Papaioannou & Wilson, 2010), computational biology (Bakhteh et al., 2018; Banga, 2008; Biggs & Papin, 2017; Jimenez-Guardeño et al., 2022; Lewis et al., 2021; Morris et al., 2010), and medicine (Potyagaylo et al., 2014; Weber et al., 2006) among many others. The quadratic binary optimization problem (QUBO) is a versatile subclass with diverse applications in areas from operations research and finance to physics, quantum computing and engineering design (Punnen, 2022).

Although in general these problems are NP-hard, a number of deterministic approaches have been developed for their resolution (Punnen & Sotirov, 2022). Among the advantages of these techniques are their theoretical properties and exact nature for small and medium size problems. However, their disadvantages quickly arise when the dimension of the problems increases, generally requiring excessive execution times and, in many cases, prohibitive memory requirements. Therefore, other stochastic methods, and especially metaheuristics, have gained popularity in recent years (Agrawal et al., 2021; Lü et al., 2010; Sörensen & Glover, 2013). Examples of recent papers exploting metaheuristics to solve problems from the classes listed above include (Mafarja et al., 2018), (Al-Tashi et al., 2019), (Taghian & Nadimi-Shahraki, 2019) and (Arora & Anand, 2019) for feature selection, (Hussien et al., 2020) for dimensionality reduction, (Reddy et al., 2019) for unit commitment, (Pan et al., 2021) for manufacturing cell formation, or (González, Prado-Rodriguez, et al., 2022) for cell signaling networks.

There are many different metaheuristics used for general combinatorial optimization problems (Crawford et al., 2017; Dahi et al., 2015; Hussien et al., 2020; Lozano & García-Martínez, 2010; Reddy et al., 2019; Taghian et al., 2018). One of the most popular is Ant Colony Optimization (ACO) (Dorigo & Stützle, 2019). It is inspired by the social behavior of ant colonies, specifically in the deposition of pheromones along the explored paths during the search for food sources. ACO has been found to be robust and easily tailored to a wide range of optimization problems, and it has been applied to a number of binary combinatorial instances (Al-Ani, 2005; Jang et al., 2011; Kashef & Nezamabadi-pour, 2015; Kong & Tian, 2005).

The basic ACO is a general-purpose algorithm, easy to understand and implement. ACO achieves good results in unimodal problems, that is, those defined by the fact that all solutions are guided towards the same optimal result without local minima. However, when tackling problems in which local minima abound, its convergence quickly suffers, easily stagnating in one of the local solutions. This fact has driven most recent proposals in the literature to highly adapted solutions to the problem at hand, and therefore, loosing its all-purpose feature.

In this work, we present an extension of a parallel ACO implementation to handle challenging binary combinatorial problems. The main goals of the work presented in this paper are:

- An extensive analysis of the performance of ACO algorithm for large and difficult instances using a previous cooperative parallel implementation (González, Osorio, et al., 2022).
- The improvement of the performance of the cooperative parallel ACO by proposing a self-adaptive runtime implementation guided by the results of the previous analysis.
- The preservation of the general-purpose solver feature of the ACO algorithm, that can be applied as a black-box solver to a large range of problems.

The structure of the paper is as follows. Section 2 presents the related work. Section 3 describes the ACO algorithm adapted to handle binary combinatorial problems. In Section 4 the cooperative parallel scheme proposed is explained. In Section 5, we present the experiments carried out and discuss the results. Finally, in Section 6 we summarize the conclusions of this work.

2 | RELATED WORK

In this section, a bibliographic review of related work is made from two different points of view: state-of-the-art on metaheuristics for combinatorial problems and development of parallel implementations.

2.1 | Metaheuristics for combinatorial problems

Metaheuristic methods are among the most advanced global optimization algorithms. Their application to combinatorial optimization problems is a booming field of research, due to the importance of those problems for both the scientific and industrial worlds.

A significant subset of metaheuristics consists of so-called swarm intelligence algorithms, which are often inspired by nature. Probably the two best known algorithms within this category are Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO). ACO (Dorigo & Stützle, 2019) simulates the behavior of ants in their colonies when foraging for food. PSO (Wang et al., 2018) attempts to emulate the group behavior of some animals, such as flocks of birds, by using information exchange between individuals to move the whole group. Both algorithms are general-purpose ones, and they have been used successfully for more than 20 years. However, other alternatives develop ad-hoc for specific problems outperform ACO and PSO. Therefore, during these last two decades, many variants have appeared, as well as many other innovative proposals. A nice recent review can be found in (Karimi-Mamaghan et al., 2022).

Most recent research works in this field have developed novel proposals focusing on problems with very specific characteristics. One of the goals of our work is to extend the ACO, as a widely recognized metaheuristic, through a parallel cooperative implementation that includes a self-adaptive solution, so that the user does not to have to tune the parameters of the new variant and it preserves its general-purpose feature.

2.2 | Parallel metaheuristics

Current trends in computing, such as the proliferation of multicore system and accelerators, or the convenience of accessing HPC resources in the Cloud, increase the interest in parallelizing time-consuming methods. Metaheuristics can help to solve large and difficult problems, by means of parallelization, when the search space cannot be fully explored by a single process. Many different parallel solutions have been proposed in the literature. A good review can be found in (Alba et al., 2013). In this section, we will focus on reviewing related work on ACO metaheuristics.

The parallelization of the ACO has been studied before in a number of previous works illustrating the use different paradigms, programming languages and parallel infrastructures (Cecilia et al., 2013; Delisle et al., 2005; Randall & Lewis, 2002; Stützle, 1998; Zhou et al., 2018). In this paper, we adopt a multicolony model, where several colonies explore the search space in isolation but including cooperation steps where information is exchanged among them. Other authors have previously explored this model for the parallelization of the ACO algorithm (Twomey et al., 2010; Chen et al., 2012; Starzec et al., 2020; González, Osorio, et al., 2022). All of these works confirm that a compromise between exploration within each colony and cooperation through information exchange is required to achieve accurate results and good performance.

We have extensive experience in the parallelization of different metaheuristics using different strategies (González et al., 2018, 2019; Pardo et al., 2020; Penas et al., 2015, 2015b; Penas et al., 2017; Teijeiro et al., 2016,b). Based on this previous experience, we have proposed in González et al. (2022), a parallel ACO algorithm adapted to a particular binary combinatorial problem, the signaling of cellular networks. The most notable features of the proposed algorithm were decentralization, since a coordination process is not needed to organize and control the algorithm, and the use of an asynchronous communication protocol between processes. However, while that proposal initially proved efficient for the specific problem at hand, it encountered convergence issues when extended to a wider range of problems. Therefore, one of the objectives of the work presented in this paper is to improve the efficiency of the parallel multi-colony algorithm through a self-tuned smart cooperation between colonies.

3 | ANT COLONY OPTIMIZATION FOR BINARY COMBINATORIAL PROBLEMS

The ACO algorithm is based on the observation of the behavior of real ants. In nature, ants follow the trail of pheromones left by the others when looking for food sources. In this algorithm, the artificial ants in a colony build the solutions in each iteration and deposit pheromones in a matrix that guides them through subsequent iterations (Blum, 2005; Parsons, 2005).

Figure 1 shows a simple scheme of ACO. A basic ACO has three main procedures: *ConstructAntsSolutions*, *UpdatePheromones*, and *DaemonActions*. *ConstructAntsSolutions* manages a colony of artificial ants that incrementally build solutions to the optimization problem by means of stochastic local decisions based on pheromone trails and heuristic information. Then, the *UpdatePheromones* procedure modifies the pheromone trails based both on the evaluation of the new solutions and on a pheromone evaporation mechanism. Finally, a *DaemonActions* procedure performs problem specific or centralized actions, which cannot be performed by single ants.

ACO is often used for problems that can be reduced to finding routes in graphs, such as the Traveling Salesman Problem (TSP) (Stützle et al., 1999). This problem is based on discovering the best route for a traveller who has to visit many cities. In a classical TSP problem, the objective is to visit all the cities and return to the origin covering the shortest possible distance. When it comes to binary combinatorial problems, this can be reduced to finding the optimal path that goes from one node to another by choosing between two possible paths: 0 or 1 (see Figure 2). Pheromones will be deposited on paths 0 or 1 in each of the N steps. Ants in the subsequent iterations will be influenced by the previously deposited pheromones.

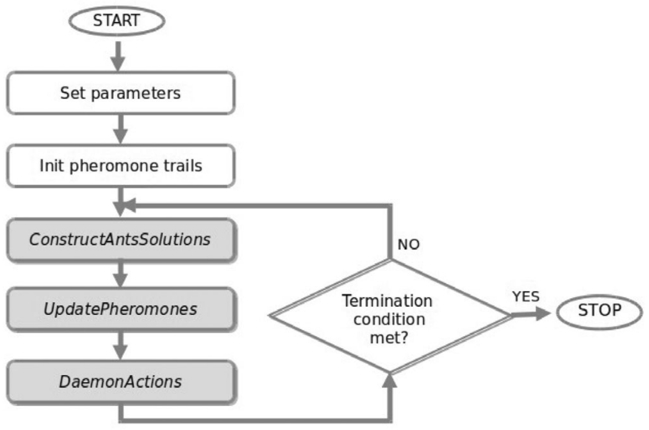


FIGURE 1 Basic scheme of the ACO algorithm.

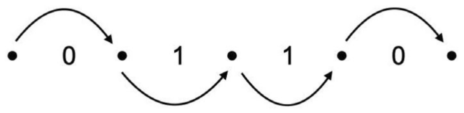


FIGURE 2 In a binary combinatorial problem, ants choose path 0 or path 1 at each step.

To decide the new path, each artificial ant applies the following probabilistic transition rule that depends on pheromone values:

$$p_{ij} = \frac{\tau_{ij}}{\tau_{i0} + \tau_{i1}} \tag{1}$$

where τ_{ij} represents the desirability of using the path j to cross edge i given by the pheromone trails, that is, whether to follow path 0 (τ_{i0}) or path 1 (τ_{i1}).

After the construction of a new solution by each ant, the pheromone trails are updated, increasing their values when ants deposit pheromone on promising paths to guide other ants in constructing new solutions, or decreasing their values due to pheromone evaporation. An evaporation process avoid unlimited accumulation of pheromone trails and also to allow bad choices to be forgotten, preventing the algorithm from premature convergence to suboptimal regions and from getting stuck in a local optimum. The evaporation procedure is implemented by decreasing τ by a constant rate ρ (the pheromone evaporation rate):

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} \tag{2}$$

Then, ants deposit pheromone on the paths they have crossed in their construction:

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\left(\tau_{ij}\right) \tag{3}$$

As shown in the transition rule (Equation 1), the possibility for an ant to cross a path increases with the pheromone trail. Therefore, it is in this step where most of the variants of the ACO algorithm differ. In this work, the *MAX – MIN* Ant System (MMAS) variant (Stützle & Hoos, 2000) has been used. This variant strongly exploits the best paths found, since only the *iteration-best* ant, that is, the ant that constructed the best solution in the current iteration, or the *best-so-far* ant, that is, the ant that constructed the best solution so far, deposits pheromones in each iteration:

$$\Delta\left(\tau_{ij}\right)^{best} = \begin{cases} 1/f\left(S^{best}\right), & \text{if path } j \text{ for edge } i \text{ belongs to } S^{best} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where $f\left(S^{best}\right)$ is the function score of the solution S^{best} found by the *iteration-best* ant or the *best-so-far* ant.

In most general implementations of the MMAS algorithm, the use of *iteration-best* solution and the *best-so-far* solution alternates. The choice of the relative frequency with which the two pheromone update rules are applied has an impact in the search: when pheromone updates are always performed by the *best-so-far* ant, the search focuses very quickly around the *best-so-far* solution, whereas when the *iteration-best* ant updates pheromones the search is less directed. Experimental results indicate that for small problems it may be better to use only *iteration-best* pheromone updates, while for large ones the best performance is obtained by giving an increasingly stronger emphasis to the *best-so-far* solution. This can be achieved by gradually increasing the frequency with which the best-so-far ant updates the pheromone trails. For more details on the particular implementation took as a basis in this work, the reader is referred to (González, Prado-Rodriguez, et al., 2022).

In ACO, when the algorithm gets stagnated, that is, after a certain number of iterations without improving the best solution, a restart is executed. In addition to re-initialising the pheromone matrix, the fitness of the best-so-far ant is assigned the worst possible score. This ensures that in the next step, in all probability, the best-so-far ant is replaced by the best-iteration ant. As the pheromone matrix is reset and all the paths have the same probability to be chosen, the best-iteration ant is completely random, thus achieving a complete restart. Each time a restart is done, new paths are explored, avoiding repeating deficient solutions.

As mentioned before, ACO offers good results for solving unimodal problems. However, when it comes to solving difficult problems, especially those with many local minima, the ACO tends to get stuck easily. To avoid premature convergence to local minima and, thus, the stagnation of metaheuristics, previous studies indicate the need of increasing the diversity in the search. A good way to achieve this is the use of parallel strategies. This is especially effective if it is accomplished using a parallel infrastructure, which will speed up the execution time. The following section describes the solution used in this work and the enhancement introduced in the cooperation strategy to improve its performance.

4 | PARALLEL ACO

The fact that the *ConstructAntsSolutions* procedure consists of tasks that can be performed independently by each ant, facilitates the implementation of parallel ACO approaches. These parallel proposals are especially appealing for solving problems with a large computational cost, since they may lead to shorten the execution time significantly. However, the ACO parallel proposals can not only shorten the execution time by performing tasks in parallel, but could also modify the systemic properties of the algorithm and enhance its convergence.

Different parallel strategies can be applied to metaheuristics in general (Alba, 2005), and ACO in particular (González, Osorio, et al., 2022). Most of them can be classified into fine-grained and coarse-grained strategies. A fined-grained parallelization attempts to find parallelism in the sequential algorithm. In the case of the ACO metaheuristic, finding the parallelism in the sequential algorithm is straightforward, since most of the time-consuming operations are placed in loops that can be performed in parallel within the *ConstructAntsSolutions* procedure. However, in fined-grained approaches the parallel algorithm maintains the sequential behavior in terms of convergence.

A different solution is a coarse-grained approach, which involves looking for a parallel variant of the sequential algorithm. The most popular coarse-grained solution consists of implementing an island-based model. In these models, different distributed colonies exist where the original algorithm is executed in isolation and, from time to time, these colonies exchange information that allow them update their results with the information received from the rest. This parallel implementation is usually known as multicolony model. A schematic representation can be seen in Figure 3.

Multicolony approaches aim to take advantage of distributed resources to extend the search for solutions. The most trivial multicolony solution consists of a parallel search on multiple non-cooperating colonies. Although this solution was found to yield good results, results usually stand out for approaches that include colony cooperation.

The cooperative approach proposed in this work is based on the model proposed in González, Osorio, et al. (2022), adapted to a binary combinatorial problem in González, Prado-Rodriguez, et al. (2022). In order to improve the efficiency of the algorithm in common multimodal problems, we propose an improvement of the cooperative strategy.

4.1 | Cooperative scheme

When designing a parallel multicolony ACO approach, some key issues must be addressed, such as what information is exchanged between colonies, which of them are involved in the communication process, when and how this process is carried out, and how the information received in each colony is used. In this work, the multicolony approach proposed in González, Osorio, et al. (2022) have been followed. The exchange of information between colonies is driven by the quality of the solutions, that is, when a colony finds a promising solution, it broadcasts it to other colonies. For this, an asynchronous communication protocol is used, thus avoiding some colonies remaining inoperative while waiting for information

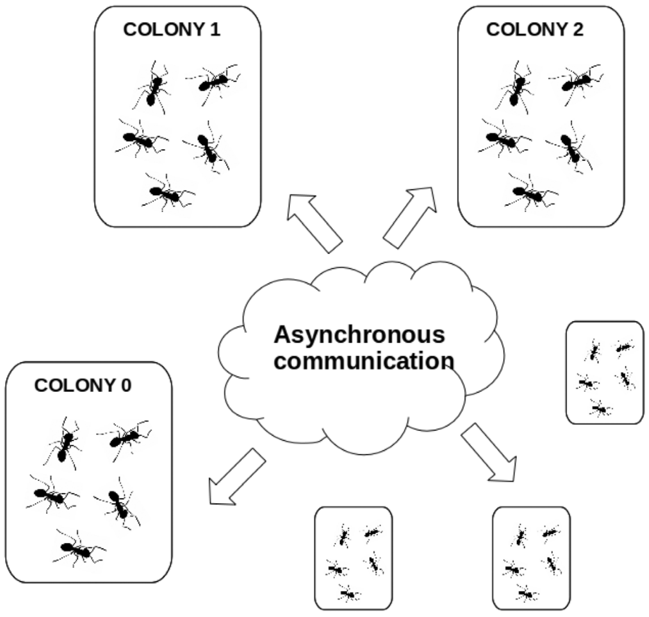


FIGURE 3 Scheme of a multiconolony implementation.

from other colonies. In the cooperative scheme proposed in González, Osorio, et al. (2022), when a promising new solution arrives at a colony and improves the *best-solution-so-far*, the latter is always replaced by the former. However, although that scheme performs well for some problems, it may not be effective in many cases. Foremost, the goal of the parallel cooperative schemes is that promising colonies help those that are stuck in local minima. But in practice, the opposite outcome often occurs: a colony that receives a better solution is diverted from its own search. All colonies converge to the same local solution, reinforcing the same path, and eventually getting stuck at the same local minimum. In short, a full cooperation strategy can often lead to the loss of the diversity that the multicolony parallel strategy claims for.

In this work, an efficient selective cooperative scheme is presented. The proposal maintains the all-purpose property and the key features of the former, namely: a cooperation driven by the quality of the solutions and a completely asynchronous implementation. When a colony obtains a promising solution, this solution is spread to the rest and all processes receive the promising solutions. However, to avoid the problem mentioned, only a few processes introduce these solutions into their colony, modifying the pheromone matrix. To determine if a solution that has just arrived in a process should be included in the colony, two aspects are taken into account.

First, although all the colonies cooperate by spreading their promising solutions, some colonies keep their execution outside the influence of the rest, for which they never use the solutions received from outside the colony. This ensures the desired diversity in the ACO progression. The number of colonies that remain independent depends on an integer parameter called *cfreq*. It could be set from $cfreq = 1$ (full cooperation since 1 out 1 colonies – all of them – use the incoming solutions) to $cfreq \rightarrow \infty$ (no-cooperation between colonies).

Second, to further avoid the danger of premature convergence due to early cooperation, the processes will only use the solutions received from other colonies once they have been stalled for a certain number of iterations. This number of iterations is defined by the *cstall* parameter. This ensures that, even the processes that use the incoming solutions, introduce them in the colony when their execution is stalled.

By introducing these two parameters, the tier of cooperation required can be controlled. This cooperation depends on the size of the problem, its hardness, and the resources available to address the problem at hand. The pseudocode of the proposal, called Binary Parallel Cooperative ACO (BiPCACO), is shown in Algorithm 1.

As a general rule, it is very complicated to know the optimal level of cooperation for a problem before dealing with it. To address this situation, a self-adaptive approach to automatically determine the tier of cooperation is also proposed in this paper. To do this, the size of the problem and the number of resources to be used are taken as a basis, and the parameters are tuned at runtime. Note that the hardness of the problem, which also influences the level of cooperation, is impossible to determine beforehand. Let us call cooperation-index to the ratio between the size of the problem and the number of processes to be used. The higher this index is, the more intensive the cooperation between the colonies should be. Ranges of this index are established to set an upper limit to the *cstall* parameter. The minimum *cstall* is always 0 (full cooperation). Algorithm 2 shows a pseudocode of the proposal. At the start of the execution, *cstall* takes its maximum value, according to the cooperation-index of the problem. The cooperation at the beginning of the execution is, then, scarce. Thus, the algorithm pursues for the diversity of multiple colonies. However, each time the algorithm gets stuck and a restart is triggered, *cstall* is reduced by 10% of restart-iterations size, so the algorithm increases their rely on incoming solutions after being reinitialized. If reboots continue to occur, the *cstall* will continue to drop and, thus, the algorithm increases the cooperation between colonies. When the minimum *cstall* is reached (0 iterations), the algorithm returns to the maximum value. In this way, even in multimodal problems, where the characteristics can change as the execution goes through the different search spaces, different cooperation degrees are explored in a round-robin fashion.

5 | EXPERIMENTAL RESULTS

In this section, a series of experiments are shown to assess the value of the strategies proposed in this work.

5.1 | Testbed

All the benchmarks used to carry out the experiments reported in this paper are obtained from the W-Model (Weise et al., 2020). The W-Model is a tunable black-box discrete optimization benchmarking problem (BB-DOB) that uses a bit-string representation of the data. The W-Model framework creates different benchmarks by means of different input parameters that modulate different features for the problems.

In Weise et al. (2020), a set of 19 diverse benchmarks was selected as a representative pool because they exhibit very different features, hardness and algorithm performance. In this work, the same set of benchmarks have been used, in order to be able to compare the results with that outstanding work. These benchmarks are labelled as 1 to 19 in the following.

Six additional challenging benchmarks labelled as 20–25 have been defined as well. The W-model parameters used to define these challenging problems can be seen in Table 1. The reader can consult the parameters for the original 19 benchmarks in Weise et al. (2020).

Table 2 summarizes the features of this set of benchmarks, classified as:

ALGORITHM 1 BiPCACO pseudocode.

```
// Initialize each colony and MPI environment
MPI_Init
Initialize colony parameters
Initialize colony pheromone trails
// Prepare a reception buffer for asynchronous communications in each colony
MPI_IRecv(promising-solution,request)
while termination condition not met do
    // Each colony constructs a set of new solutions isolately
    ConstructSolutions
    // When a new local best solution is found, it is spread asynchronously to the colonies
    if local-best-solution-so-far < global-best-solution-so-far then
        MPI_ISend(local-best-solution-so-far)
    // Each colony check the reception of foreign promising solutions
    repeat
        // MPI check asynchronous reception
        MPI_Test(request, recvflag)
        if recvflag then
            // Only some colonies include foreign solutions to their search
            if promising-solution < global-best-solution-so-far && process_id%cfreq == 0 && stall_iters >= cstall then
                global-best-solution-so-far ← promising solution
            // Prepare a new reception buffer for next receptions
            MPI_IRecv(promising-solution,request)
    until (!recvflag)
    // Each colony updates its pheromone matrix
    UpdatePheromones
    DaemonActions
```

ALGORITHM 2 Establishing initial maximum cooperation

```
coop-index ← problem_size/NPROC
// Note that min_cstall ← 0 means full cooperation.
min_cstall ← 0
if $coop - index > 200$ then
    max_cstall ← 10% restart_iters
else if $coop - index > 150$ then
    max_cstall ← 20% restart_iters
else if $coop - index > 100$ then
    max_cstall ← 30% restart_iters
else if $coop - index > 50$ then
    max_cstall ← 40% restart_iters
else
    max_cstall ← 50% restart_iters
cstall ← max_cstall;
```

TABLE 1 W-model parameters for benchmarks 20–25.

Benchmark	Problem size	Neutrality	Epistasis	Ruggedness/Deceptiveness
20	640	4	130 (81%)	10,000 (78%)
21	720	4	150 (83%)	12,000 (75%)
22	1000	4	200 (80%)	25,000 (80%)
23	640	4	130 (81%)	0
24	720	4	150 (83%)	0
25	1000	4	200 (80%)	0

TABLE 2 Features of the W-Model benchmarks used in this section.

Benchmark	Problem size	Neutrality	Epistasis	Ruggedness/deceptiveness
1	20	Low	Medium	Low
2	20	Low	Medium	Medium
3	16	-	Low	Medium
4	48	Medium	Medium	Medium
5	25	-	High	Medium
6	32	-	-	High
7	128	High	Low	-
8	128	High	Medium	-
9	128	High	Low	Low
10	50	-	High	Low
11	100	Low	Medium	Low
12	150	Medium	Low	Medium
13	128	Low	Medium	Low
14	192	Medium	Low	Very low
15	192	Medium	Low	Low
16	192	Medium	Low	Low
17	256	High	High	Very low
18	75	-	High	Very low
19	150	Low	Medium	Very low
20	640	High	High	High
21	720	High	High	High
22	1000	High	High	High
23	640	High	High	-
24	720	High	High	-
25	1000	High	High	-

- *neutrality*: when a search operation applied to a solution candidate yields no change in objective value.
- *epistasis*: when the contribution of some decision variables to the objective value depends on the value of other decision variables.
- *ruggedness*: when small changes in a solution cause large changes in its fitness.
- *deceptiveness*: when a move to a gradient descend leads the search away from the global optimum.

All the experiments were performed at the Galicia Supercomputing Center (CESGA) using the FinisTerrae-III supercomputer. Each FinisTerrae-III node is composed of two Intel Xeon Ice Lake 8352Y CPUs running at 2.2 GHz, with 32 cores per processor (64 cores per node), and 256 GB of RAM. The nodes are connected using an Mellanox InfiniBand HDR 100 Gbps interconnect using a fat-tree topology.

TABLE 3 BiPCACO parameters used in the experiments of this section.

Parameter	Value	Description
Population	200	Number of solutions per iteration
Evaporation rate	0.05	Percentage of pheromone reduced per iteration
Restart	50,100	Iterations with no improvement that trigger a restart. 50 for sequential algorithm and 100 for parallel one
cfreq	1,2	Frequency of colonies receiving solutions if stagnated. Here we try 1 of 1 and 1 of 2 colonies
cstall	0–∞	Receive other colonies solutions after cstall iterations of stagnation

5.2 | Methodology and reproducibility

To allow the reproducibility of the experiments shown in this paper, authors make public available the code of the BiPCACO algorithm in the https://gitlab.com/RobertoPradoRodriguez/bipcaco repository.

Different tests have been carried out in this work. Experiments have been performed using as stopping criteria both a maximum effort (in execution time) or a quality objective (that is, achieving the optimum value in each experiment, which in the case of the W-Model benchmarks is the value 0).

The user-defined parameters of the proposed BiPCACO algorithm are shown in Table 3, where *n* is the problem size.

For the generation of random numbers, the MT19937 variant of the Mersenne Twister algorithm (Matsumoto & Nishimura, 1998) is used. A four-digit number is taken as the initial seed.

Given the stochastic nature of these methods, a total of 100 executions have been carried out for each experiment, and a statistical study has been carried out on the reported data.

5.3 | Assessment of BiPCACO proposal

To assess the cooperative scheme, the results of seven different configurations have been compared, from no cooperation to intensive cooperation.

- M1: $cfreq \to \infty$, that is, a configuration without cooperation between colonies.
- M2: $cfreq \to 2$, that is, only one out of two (50%) of the colonies incorporate foreigner solutions to their search; and $cstall = n/25$, that is, the foreigner solutions are incorporated only when the number of stagnate iterations is greater than or equal to $n/25$.
- M3: $cfreq \to 1$, that is, all the colonies incorporate foreigner solutions to their search; and $cstall = n/10$, that is, the foreigner solutions are incorporated only when the number of stagnate iterations is large ($n/10$).
- M4: $cfreq \to 1$ and $cstall = n/25$, that is, all the colonies incorporate foreigner solutions to their search when the number of stagnate iterations is greater than or equal to $n/25$.
- M5: $cfreq \to 1$ and $cstall = n/50$, that is, all the colonies incorporate foreigner solutions to their search when the number of stagnate iterations is small ($n/50$).
- M6: $cfreq \to 1$ and $cstall = 0$, that is, all the colonies incorporate foreigner solutions every time they received it. This is the most intensive cooperative configuration.
- M7: self-adapted, that is, the degree of cooperation is dynamically adjusted in execution time.

Note that M1 configuration corresponds to a non-cooperative parallel solution, while M6 corresponds to the cooperation scheme proposed in González, Osorio, et al. (2022), that is, a full cooperation between colonies. Then, M7 configuration corresponds to the self-adaptive solution proposed in this paper.

Figure 4 summarizes the results obtained for experiments using the optimum value as stopping criterium. This figure displays the boxplots of the execution time in seconds of 100 runs per experiment, and attempts to compare different cooperative configurations using the most challenging benchmark problems of the testbed. These challenging benchmarks are not only large, but also three of them (20–22) exhibit features that make them very difficult to solve, as they are *multimodal* and are defined by having large number of local minima. In those instances, the cooperation between colonies favors the convergence rate of the algorithm. Problems 23, 24 and 25 are unimodal, that is, the search is oriented smoothly to the global minimum. Since the algorithm does not get stuck, it does not need cooperation and, therefore, there is hardly any difference in problems 23 and 24 among all methods. In practice, all behave in a non-cooperative fashion. Problem 25, although unimodal, is too large in size and cooperation becomes necessary.

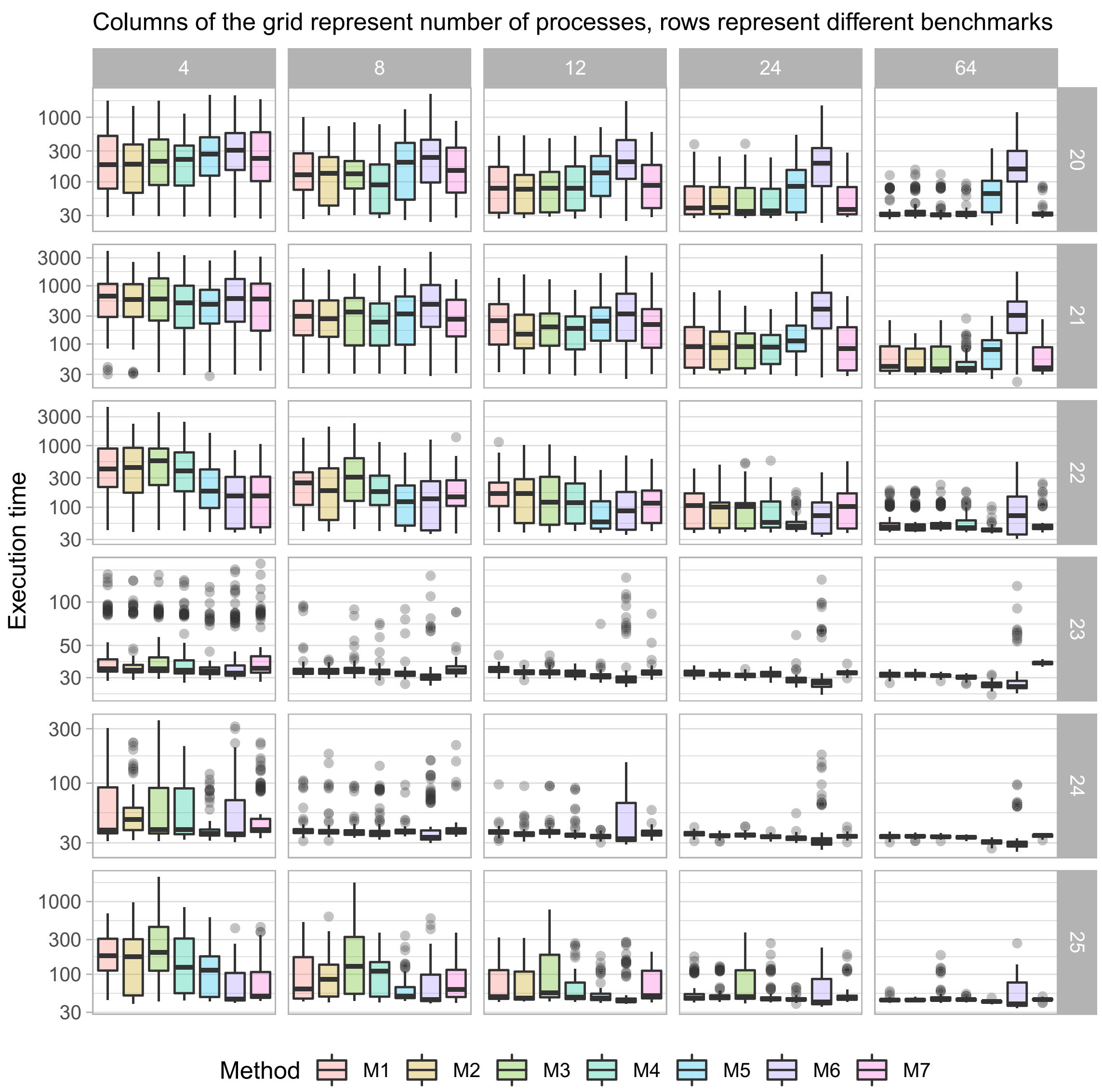


FIGURE 4 Comparison of different cooperative configurations in BiPCACO. Boxplots of the execution times achieved in 100 runs of experiments using optimum value as stopping criteria, for the most challenging benchmarks using 4, 8, 12, 24 and 64 processes.

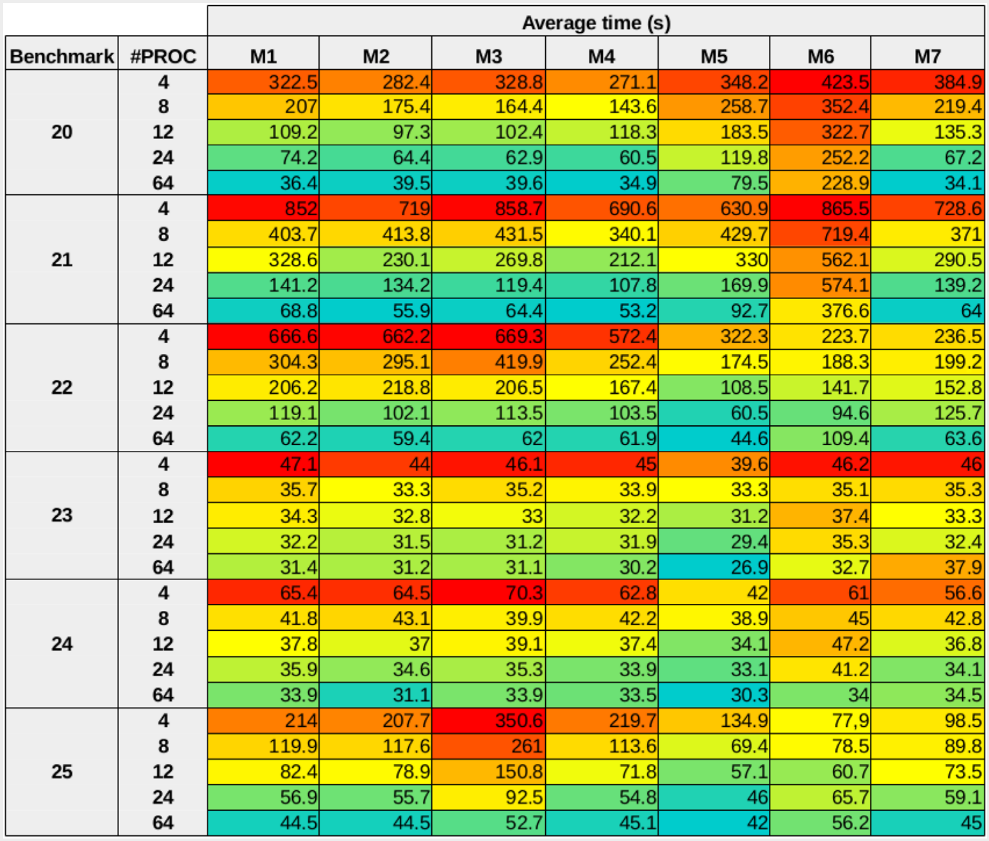
In Figure 4, the dispersion of the results can be observed, and the median is highlighted. In order to clarify, Table 4 shows the results of the average execution time for each experiment. The cells are colored to show, for the same benchmark and number of colonies, the best results obtained through different cooperation levels.

Certain conclusions can be drawn from these results. A mid cooperation scheme, for example the M4 configuration, outperforms the other configurations for 4, 8, 24 and 64 colonies in terms of average time for benchmark 20. In benchmark 21, tougher that benchmark 20, there is a need for increasing the cooperation, and results obtained for M5 configuration outperform the rest when few processes are involved. This situation is reaffirmed in problem 22, larger and more difficult than benchmark 21. The degree of cooperation that achieves the best performance is M6. That is, the larger and more difficult to solve the problem, the larger the need for cooperation between the colonies.

These results also demonstrate that an intensive cooperation (M6 configuration) is effective when using few processes to solve very difficult problems. In those experiments, the opportunities of finding a good solution are low, so the need to share promising solutions as soon as possible is high in order to accelarate the progress of the search. However, when the number of processes increases, the chances for one of them to find a good solution on its own increases, and cooperation can interfere with this search and harm diversity. If a colony frequently accepts external solutions, the process deviates from their own search and ends up converging towards the same local minimum as the rest of the colonies. This behavior is more accentuated the more processes exist.

Based on previous experiments, it can be concluded that there are essentially three features of the problem at hand that will determine the degree of cooperation that benefits BiPCACO execution:

TABLE 4 Comparison of different cooperative configurations in BiPCACO.



		Average time (s)						
Benchmark	#PROC	M1	M2	M3	M4	M5	M6	M7
20	4	322.5	282.4	328.8	271.1	348.2	423.5	384.9
	8	207	175.4	164.4	143.6	258.7	352.4	219.4
	12	109.2	97.3	102.4	118.3	183.5	322.7	135.3
	24	74.2	64.4	62.9	60.5	119.8	252.2	67.2
	64	36.4	39.5	39.6	34.9	79.5	228.9	34.1
21	4	852	719	858.7	690.6	630.9	865.5	728.6
	8	403.7	413.8	431.5	340.1	429.7	719.4	371
	12	328.6	230.1	269.8	212.1	330	562.1	290.5
	24	141.2	134.2	119.4	107.8	169.9	574.1	139.2
	64	68.8	55.9	64.4	53.2	92.7	376.6	64
22	4	666.6	662.2	669.3	572.4	322.3	223.7	236.5
	8	304.3	295.1	419.9	252.4	174.5	188.3	199.2
	12	206.2	218.8	206.5	167.4	108.5	141.7	152.8
	24	119.1	102.1	113.5	103.5	60.5	94.6	125.7
	64	62.2	59.4	62	61.9	44.6	109.4	63.6
23	4	47.1	44	46.1	45	39.6	46.2	46
	8	35.7	33.3	35.2	33.9	33.3	35.1	35.3
	12	34.3	32.8	33	32.2	31.2	37.4	33.3
	24	32.2	31.5	31.2	31.9	29.4	35.3	32.4
	64	31.4	31.2	31.1	30.2	26.9	32.7	37.9
24	4	65.4	64.5	70.3	62.8	42	61	56.6
	8	41.8	43.1	39.9	42.2	38.9	45	42.8
	12	37.8	37	39.1	37.4	34.1	47.2	36.8
	24	35.9	34.6	35.3	33.9	33.1	41.2	34.1
	64	33.9	31.1	33.9	33.5	30.3	34	34.5
25	4	214	207.7	350.6	219.7	134.9	77.9	98.5
	8	119.9	117.6	261	113.6	69.4	78.5	89.8
	12	82.4	78.9	150.8	71.8	57.1	60.7	73.5
	24	56.9	55.7	92.5	54.8	46	65.7	59.1
	64	44.5	44.5	52.7	45.1	42	56.2	45

Note: Average time (in seconds) achieved on experiments reported in Figure 4. Color coding per benchmark and colonies from blue (best time) to red (worst time).

- Amount of processes: the smaller the number of processes, the higher the cooperation between colonies must be.
- Multimodality: The higher the bias towards the multimodal landscape, the larger the probability that the algorithm will get stuck and the greater the need for cooperation.
- Problem size: the larger the problem, the greater the need for cooperation.

The problem of the previous conclusions is the difficulty for the user to know the features of the problem in advance, and therefore the difficulty of adjusting the configuration parameters before the execution. It is at this point where the self-tuned approach is especially appealing. Moreover, results of Table 4 evidence that it is also competitive when compared with the solution obtained with the best configuration in each problem.

Results in previous figures were obtained from 100 independent runs of each experiment. However, displaying only the average execution time does not give a clear view of the behavior of the algorithm according to the level of cooperation introduced with the different configurations. Therefore, an additional statistical study was performed on these experimental data.

To demonstrate the significance of the results, a nonparametric statistical analysis has been applied to the run times of the experiments for each benchmark (B20–B25) and each configuration (M1–M7). Nonparametric procedures are popular methods to compare the performance of different metaheuristics (Carrasco et al., 2020; Derrac et al., 2011). In this work we have applied the Kruskal–Wallis test (Kruskal & Wallis, 1952) followed by Dunn's test (Dunn, 1964). The goal is to explain whether the observed differences among the final execution times of each problem are due to the different cooperative configuration or to pure chance.

Table 5 shows, for each problem, the *p*-values of the Kruskal–Wallis test for the comparison of all configurations with different number of processes. The Kruskal–Wallis test is used to determine whether or not there is a statistically significant difference among three or more independent groups. When the *p*-value is less than 0.05, the Kruskal–Wallis test concludes, with a confidence level of 95%, that the groups are indeed different. It can be seen that, in most the cases, the *p*-values are so low that we can assume that the differences are due to the cooperation setup and not for pure random. The only exceptions are benchmarks 20 and 21 using only 4 processors, where the *p*-values are larger.

TABLE 5 Kruskal–Walis test *p*-values for a confidence level of 95%, comparing the seven different configuracions (M1–M7) for each benchmark and number of processes (tested using the results reported in Table 4).

	#PROC				
Benchmark	**4**	**8**	**12**	**24**	**64**
20	0.03898	3.56×10^{-6}	1.11×10^{-12}	$<2.2 \times 10^{-16}$	$<2.2 \times 10^{-16}$
21	0.2883	0.001403	8.46×10^{-5}	$<2.2 \times 10^{-16}$	$<2.2 \times 10^{-16}$
22	$<2.2 \times 10^{-16}$	1.13×10^{-11}	8.94×10^{-8}	5.09×10^{-6}	$<1.05 \times 10^{-14}$
23	1.58×10^{-6}	$<2.2 \times 10^{-16}$	$<2.2 \times 10^{-16}$	$<2.2 \times 10^{-16}$	$<2.2 \times 10^{-16}$
24	1.75×10^{-12}	1.68×10^{-14}	$<2.2 \times 10^{-16}$	$<2.2 \times 10^{-16}$	$<2.2 \times 10^{-16}$
25	$<2.2 \times 10^{-16}$	$<2.2 \times 10^{-16}$	$<2.2 \times 10^{-16}$	1.08×10^{-15}	$<2.2 \times 10^{-16}$

The Kruskal–Wallis test does not indicate which groups are different, so a post-hoc test is needed. We have used Dunn's test, which reports the results between multiple pairwise comparisons after the Kruskal–Wallis test. The *p*-values obtained with Dunn's test have been converted into a compact letter representation to facilitate discussion. The compact letter representation allows to display *p*-values whereby pairwise comparisons that share a letter do not reveal statistically significant differences. Table 6 shows, for each problem, the classification of the methods into groups obtained by Dunn's test for each pairwise comparison after application of the Kruskal–Wallis test.

In these results, it can be seen how by varying the configuration of the cooperation between the processes, the results differ significantly, being considered into different groups. Note again that the M1 configuration and the M6 configuration are the two extremes between non-cooperation and full cooperation between processes. In Table 6, it can be seen that, in general, they always belong to different groups, while the other configurations vary depending on the benchmarks and the number of processes. It can also be seen how the M7 configuration, that is, the self-adaptive one, gives similar results to other configurations depending on the benchmark and the number of processes used. This is precisely what this work pursues, that the self-adaptive configuration achieves competitive results with the best configuration for the problem and number of processes at hand.

It can be seen, for example, that in benchmark 20 with 4 processes the differences between the different configurations are not relevant according to these statistical tests. However, for 64 processes, results of M7 are in the group of M1, that is, when increasing the number of processes results for benchmark 20 benefit of avoiding the cooperation between colonies and maintaining the diversity in the search. If we look at benchmark 22, with 4 processes the best result is achieved by the M6 configuration, that is, the inter-colony cooperation is particularly good, and the results of M7 belong to this group. However, when increasing to 64 processes, the best results are achieved by the M1 configuration, again by maintaining the diversity of the colonies through restriction in the cooperation, and it can be seen that M7 configuration has adapted to this circumstance and again shares group with M1.

To better illustrate the behavior of the proposed M7 self-tuned cooperation method, logarithmic scale plots for benchmarks 20 and 22 are shown in Figures 5–8. Those figures show the cumulative probability of reaching the optimum related to the execution time, for the 100 runs of each experiment. We choose benchmarks 20 and 22 to show how, despite being both large and difficult problems, in one case cooperation is more beneficial than in the other, however, this is somewhat difficult to know beforehand.

As it can be seen, for benchmark 20, a configuration with no cooperation (M1) is better than an intensive cooperation (M6). However, a self-tuned cooperation (M7) adapts during runtime and offers a competitive result versus the non-cooperation solution.

On the other hand, for benchmark 22, the best solution turns out to be a configuration with an intensive cooperation (M6) compared to lack of cooperation (M1) for 4 processes. And, on the contrary, the absence of cooperation (M1) is better compared to an intense cooperation (M6) for 64 processes. Besides, the self-tuned solution (M7) is always competitive when compared with the best solution in each situation.

These previous results show that the self-tuned solution, although it may not improve the superior configuration, allows the user to get rid of the responsibility of choosing the most appropriate parameters, making the algorithm reconfigure itself, at execution time, depending on the progress of the search. Results of the self-tuned solution stands out as a competitive alternative.

Finally, Figure 9 shows the beanplots of the previous executions, to graphically illustrate the distribution of the results. Note that the bean of the self-tuned execution always achieves competitive results versus the best of the other configurations.

5.4 | Comprehensive assessment

To perform comprehensive assessment of the proposed BiPCACO algorithm, a comparison has been carried out with the results presented in (Weise et al., 2020) for 17 different algorithms, solving the 19 benchmarks described in Section 5.1. To perform a fair comparison, the same metric as in (Weise et al., 2020) was used, the ERT (Expected Run Time). ERT allows to decouple the runtime results from the infrastructure on which the experiments are executed. This metric is calculated using the same procedure described in Weise et al. (2020), aggregating the number of

TABLE 6 Compact letter display of groups after using the Dunn's test as post-hoc of Kruskal–Wallis, comparing the seven different configurations (M1–M7) for each benchmark and number of processes (tested using the results reported in Table 4).

Benchmark	#PROC	M1	M2	M3	M4	M5	M6	M7
20	4	a	a	a	a	a	a	a
	8	abc	ab	ab	a	bc	c	bc
	12	a	a	a	a	bc	b	ac
	24	ab	a	a	a	b	c	ab
	64	a	a	a	a	b	c	a
21	4	a	a	a	a	a	a	a
	8	a	a	ab	a	ab	b	a
	12	ab	a	ab	a	ab	b	ab
	24	a	a	a	a	a	b	a
	64	ab	a	a	a	b	c	ab
22	4	a	a	a	a	b	b	b
	8	ab	a	b	a	c	c	ac
	12	a	a	a	ab	c	bc	ab
	24	a	ab	a	ab	c	bc	a
	64	a	a	a	a	b	b	a
23	4	a	ab	a	abc	bc	c	a
	8	a	a	a	a	b	c	a
	12	a	b	b	b	c	c	b
	24	ab	ac	c	abc	d	d	b
	64	a	a	a	b	c	d	e
24	4	a	a	a	a	b	b	a
	8	a	ab	bc	c	ab	d	ab
	12	a	bc	a	bd	e	de	ac
	24	a	bc	ab	d	e	e	cd
	64	ab	a	ab	b	c	c	a
25	4	a	ab	a	a	bc	d	c
	8	ab	a	c	a	b	d	ab
	12	a	ab	c	ab	b	d	ac
	24	ab	a	a	bc	c	c	ab
	64	a	ab	b	ab	c	c	ab

Note: Groups followed by the same letter are not significantly different.

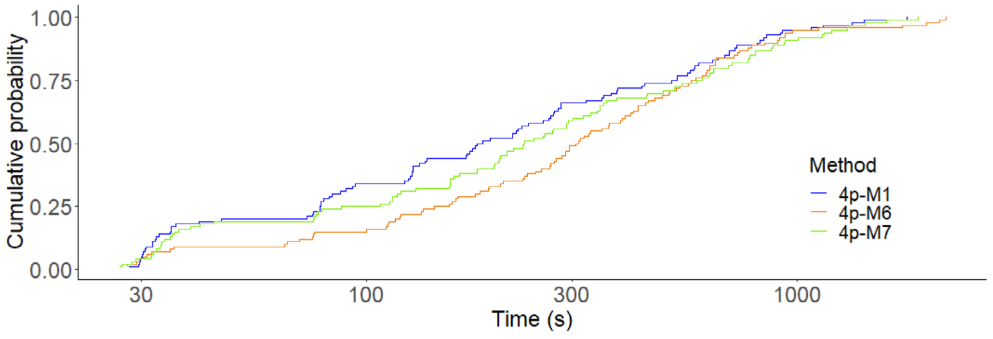


FIGURE 5 Cumulative probability of reaching the optimum in benchmark 20 using 4 processes, where M1-7 indicates different cooperation schemes as explained in Section 5.3.

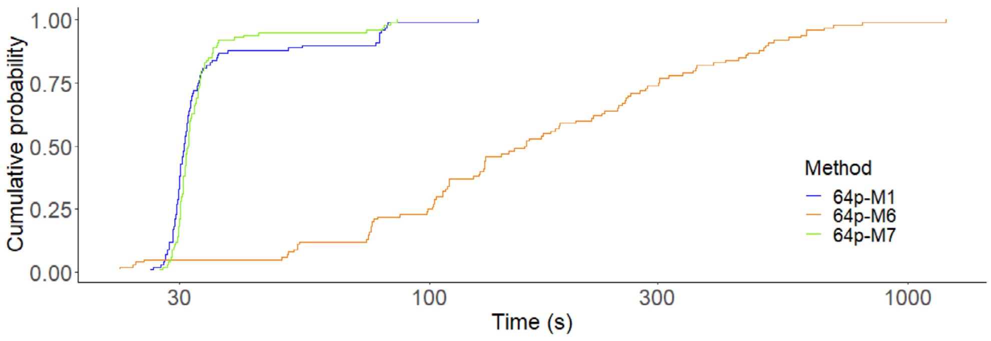


FIGURE 6 Cumulative probability of reaching the optimum in benchmark 20 using 64 processes, where M1-7 indicates different cooperation schemes as explained in Section 5.3.

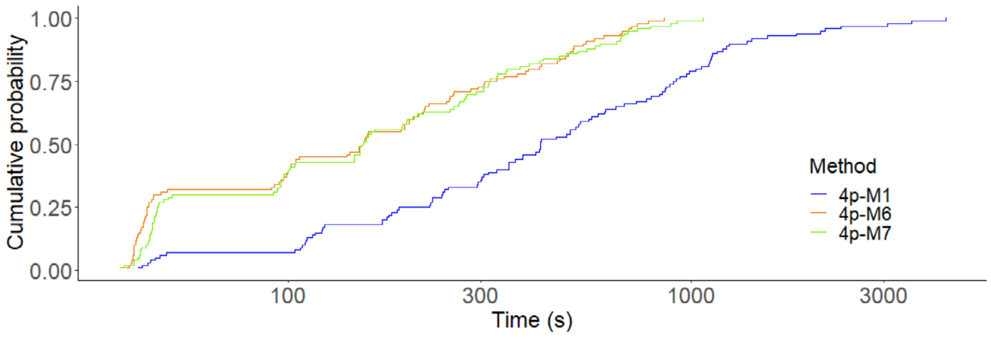


FIGURE 7 Cumulative probability of reaching the optimum in benchmark 22 using 4 processes, where M1-7 indicates different cooperation schemes as explained in Section 5.3.

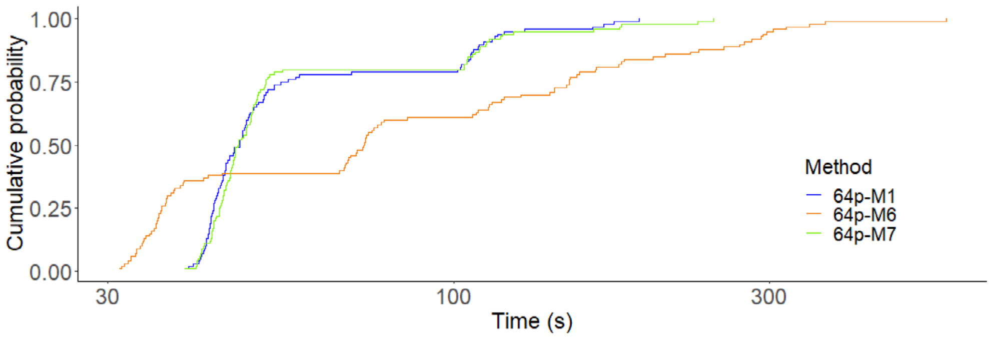


FIGURE 8 Cumulative probability of reaching the optimum in benchmark 22 using 64 processes, where M1-7 indicates different cooperation schemes as explained in Section 5.3.

evaluations of all the runs in a experiment and dividing by the number of those runs that found the optimum. The base 2 logarithm is applied and rounded off. For each run the number of evaluations included in the ERT calculation is:

- the number of evaluations of the colony that achieves it, when the optimum is reached.
- 2^{20} evaluations, when the optimum is not reached.

Figure 10 shows a table with the ERT of the 17 algorithms that were compared in Weise et al. (2020) together with the results of the sequential ACO described in Section 3 and BiPCACO proposal. Figure 11 highlights the differences between the ACO, BiPCACO variants and the best competitor for each benchmark among the remaining algorithms from Figure 10.

In these results it can be seen that the sequential ACO outperforms the rest of the algorithms in 9 of the 19 problems (specifically in the most difficult and large ones). BiPCACO with the self-tuned configuration outperforms most of them, even using only 4 colonies. The more colonies

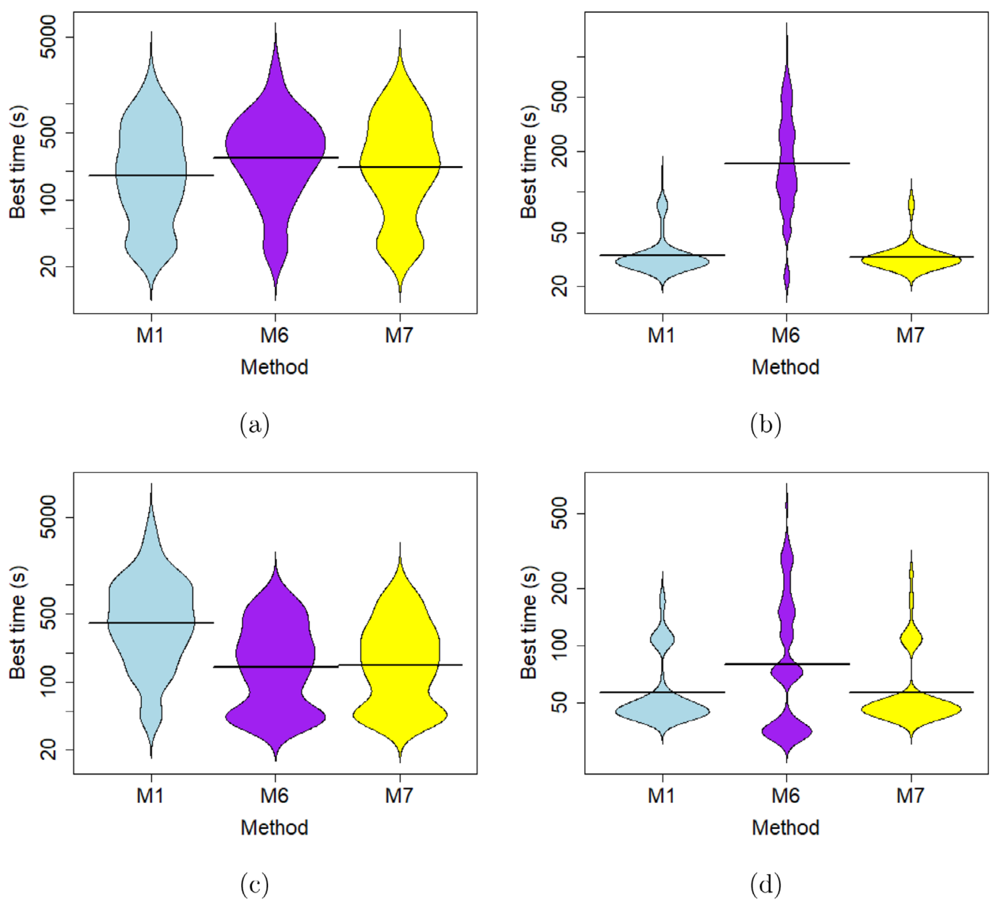


FIGURE 9 Beanplots with the distribution of the execution time for (a) benchmark 20 using 4 processes, (b) benchmark 20 using 64 processes, (c) benchmark 22 using 4 processes and (d) benchmark 22 using 64 processes, where M1-7 indicates different cooperation schemes as explained in Section 5.3.

used in the BiPCACO proposal, the better. Note that getting a small improvement on the ERT result means a significant amount of execution time, given the definition of the metric.

6 | CONCLUSION

This paper presents an improved multicolony ACO for binary combinatorial problems. This novel method aims to overcome the main handicap of such algorithm, that is, its tendency to get stuck in local minima. We introduce a parallel cooperative ACO strategy in which colonies share promising solutions with each other, but only incorporate them into their search if certain conditions are met. In other words, they collaborate only when needed. This approach maintains diversity, while avoiding rapid convergence to the same local minimum of all colonies. Based on tests with different cooperative configurations, a self-tuned version was developed that adjusts the parameters of the cooperative algorithm at runtime. This allows finding a good solution for each problem without the need to know a priori the features of the problem at hand.

The design and implementation of this proposal were guided by the goal of preserving the versatility of the original ACO algorithm. We strove to create an implementation that is easy to understand, configurable, and applicable to a wide range of binary problems, avoiding ad hoc solutions tailored exclusively to specific problem types.

To evaluate the approach, we used a set of benchmarks from the W-Model framework, which allows us to evaluate algorithms on a wide range of problem sets.

The new self-adapted cooperative strategy presents the following advantages:

- avoids the user having to know in advance which configuration is the most suitable for the problem in question
- saves the user the time to adjust the parameters of the new algorithm
- always ensures competitive runtime for wide spectrum problems

Algorithm	Benchmark 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
fea-256-7	10	10	12	12	12	12	14	14	14	17	15	17	14	17	18	17	14	17	16
fea-256-3	10	10	12	12	12	12	17	15	16	18	17	18	15	20	19	19	18	18	18
fea-32-3	9	9	11	12	12	10	17	17	16	17	19	19	20	21	21	21	22	18	23
fea-32-7	9	8	10	12	11	10	17	17	16	16	19	20	22	22	23	22	23	18	24
hcp	11	11	13	14	13	24	21	21	21	17	21	22	21	21	22	22	21	17	22
ea-256-7	10	10	12	13	13	-	19	17	20	17	20	22	17	20	21	24	18	17	19
fea-8-3	10	9	11	13	11	10	22	22	22	19	25	24	24	25	26	25	25	20	-
hc2	13	11	15	16	12	25	21	22	25	17	21	-	19	22	24	26	21	16	19
ea-256-3	11	11	13	13	14	-	21	20	23	19	25	-	21	26	27	-	24	19	27
fea-8-7	11	10	12	14	12	11	24	25	27	18	-	-	-	-	-	-	-	20	-
ea-32-7	15	16	20	19	19	-	27	26	26	24	-	-	-	-	-	-	-	-	-
ea-32-3	17	16	20	21	19	-	-	-	-	26	-	-	-	-	-	-	-	27	-
rs	11	11	16	16	24	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ea-8-7	22	21	24	23	26	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ea-8-3	22	21	25	25	27	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ee	17	17	15	-	25	-	-	-	-	-	-	-	-	-	-	-	-	-	-
hc1	23	24	27	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ACO	10,3	10,3	12,6	12,8	13,2	15,7	14	13,7	15	14,4	16	16,7	14,6	15,5	16,3	16,6	14,5	14,7	15
BiPCACO(4)	8,7	8,8	11,5	11,7	12,3	13,8	13,5	13,3	13,7	13,4	14,5	14,5	13,9	14,3	14,8	14,8	14,1	13,7	14,3
BiPCACO(8)	8,1	8,3	11,3	11,5	12,2	13,5	13,4	13,3	13,5	13,2	13,9	14,2	13,8	14,2	14,3	14,6	14,1	13,7	14,3
BiPCACO(12)	7,8	7,8	11	11,2	12	12,8	13,4	13,2	13,4	13,1	13,9	14,2	13,8	14,2	14,2	14,2	14,1	13,7	14,3
BiPCACO(24)	7,6	7,6	10,7	10,5	11,9	12,4	13,3	13,2	13,2	13,1	13,8	14,1	13,8	14,1	14,1	14,2	14	13,6	14,2
BiPCACO(64)	7,6	7,6	9,7	9,9	11,8	12,2	13,2	13,1	13,2	12,9	13,7	13,9	13,7	14	14	14,1	13,9	13,5	14,3

FIGURE 10 Comparison of ERT achieved by ACO and BiPCACO with other 17 algorithms reported in Weise et al. (2020), where ACO stands for sequential ACO and BiPCACO(N) for BiPCACO algorithm with N colonies. Color coding per benchmark from blue (best ERT) to red (worst ERT).

Algorithm	Benchmark 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Best competitor	9	8	10	12	11	10	14	14	14	16	15	17	14	17	18	17	14	16	16
ACO	10,3	10,3	12,6	12,8	13,2	15,7	14	13,7	15	14,4	16	16,7	14,6	15,5	16,3	16,6	14,5	14,7	15
BiPCACO(4)	8,7	8,8	11,5	11,7	12,3	13,8	13,5	13,3	13,7	13,4	14,5	14,5	13,9	14,3	14,8	14,8	14,1	13,7	14,3
BiPCACO(8)	8,1	8,3	11,3	11,5	12,2	13,5	13,4	13,3	13,5	13,2	13,9	14,2	13,8	14,2	14,3	14,6	14,1	13,7	14,3
BiPCACO(12)	7,8	7,8	11	11,2	12	12,8	13,4	13,2	13,4	13,1	13,9	14,2	13,8	14,2	14,2	14,2	14,1	13,7	14,3
BiPCACO(24)	7,6	7,6	10,7	10,5	11,9	12,4	13,3	13,2	13,2	13,1	13,8	14,1	13,8	14,1	14,1	14,2	14	13,6	14,2
BiPCACO(64)	7,6	7,6	9,7	9,9	11,8	12,2	13,2	13,1	13,2	12,9	13,7	13,9	13,7	14	14	14,1	13,9	13,5	14,3

FIGURE 11 Comparison of ERT achieved by ACO, BiPCACO and best result of 17 algorithms reported in Weise et al. (2020), where ACO stands for sequential ACO and BiPCACO(N) for BiPCACO algorithm with N colonies. Color coding per benchmark from blue (best ERT) to red (worst ERT).

As the closest future work, we are preparing a hybridization of the ACO with methods that perform as local search. Our main objective is to maintain a general-purpose solution by incorporating the ideas from this work on cooperation and self-adaptation at runtime.

ACKNOWLEDGEMENTS

RPR, PGG and RDB acknowledges funding from Grants PID2019-104184RB-I00 and PID2022-136435NB-I00, funded by MCIN/AEI/10.13039/501100011033, PID2022 also funded by 'ERDF A way of making Europe', EU; Xunta de Galicia and FEDER funds of the EU (Consolidation Program of Competitive Reference Groups, ref. ED431C 2021/30). JRB acknowledges funding from the Ministry of Science and Innovation of Spain MCIN/AEI/10.13039/501100011033 through grant PID2020-117271RB- C22 (BIODYNAMICS). Authors also acknowledge the Galician Supercom- puting Center (CESGA) for the access to its facilities. Funding for open access charge: Universidade da Coruña/CISUG

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in BiPCACO at https://gitlab.com/RobertoPradoRodriguez/BiPCACO.

ORCID

Roberto Prado-Rodríguez https://orcid.org/0000-0003-2651-8804

REFERENCES

Agrawal, P., Abutarboush, H. F., Ganesh, T., & Mohamed, A. W. (2021). Metaheuristic algorithms on feature selection: A survey of one decade of research (2009-2019). *IEEE Access*, *9*, 26766–26791.

Al-Ani, A. (2005). Feature subset selection using ant colony optimization. *International Journal of Computational Intelligence*, *2*(1), 53–58.

Alba, E. (2005). *Parallel metaheuristics: A new class of algorithms* (pp. 105–346). John Wiley & Sons Ch. 5–14. https://www.wiley.com/en-gb/Parallel+Metaheuristics%3A+A+New+Class+of+Algorithms-p-9780471678069

Alba, E., Luque, G., & Nesmachnow, S. (2013). Parallel metaheuristics: Recent advances and new trends. *International Transactions in Operational Research*, *20*(1), 1–48.

Al-Tashi, Q., Kadir, S. J. A., Rais, H. M., Mirjalili, S., & Alhussian, H. (2019). Binary optimization using hybrid grey wolf optimization for feature selection. *IEEE Access*, *7*, 39496–39508.

Arora, S., & Anand, P. (2019). Binary butterfly optimization approaches for feature selection. *Expert Systems with Applications*, *116*, 147–160.

Bakhteh, S., Ghaffari-Hadigheh, A., & Chaparzadeh, N. (2018). Identification of minimum set of master regulatory genes in gene regulatory networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *17*(3), 999–1009.

Baldick, R. (1995). The generalized unit commitment problem. *IEEE Transactions on Power Systems*, *10*(1), 465–475.

Banga, J. R. (2008). Optimization in computational systems biology. *BMC Systems Biology*, *2*(1), 1–7.

Biggs, M. B., & Papin, J. A. (2017). Managing uncertainty in metabolic network structure and improving predictions using ensemble FBA. *PLoS Computational Biology*, *13*(3), e1005413.

Blum, C. (2005). Ant colony optimization: Introduction and recent trends. *Physics of Life Reviews*, *2*(4), 353–373.

Carrasco, J., García, S., Rueda, M., Das, S., & Herrera, F. (2020). Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review. *Swarm and Evolutionary Computation*, *54*, 100665.

Carreira-Perpinán, M. A. (1997). *A review of dimension reduction techniques* (Vol. 9, pp. 1–69). Department of Computer Science. University of Sheffield. Tech. Rep. CS-96-09. https://www.semanticscholar.org/paper/A-Review-of-Dimension-Reduction-Techniques-Carreira-Perpi

Cecilia, J. M., García, J. M., Nisbet, A., Amos, M., & Ujaldón, M. (2013). Enhancing data parallelism for ant colony optimization on gpus. *Journal of Parallel and Distributed Computing*, *73*(1), 42–51.

Chen, L., Sun, H.-Y., & Wang, S. (2012). A parallel ant colony algorithm on massively parallel processors and its convergence analysis for the travelling salesman problem. *Information Sciences*, *199*, 31–42.

Crawford, B., Soto, R., Astorga, G., Garca, J., Castro, C., & Paredes, F. (2017). Putting continuous metaheuristics to work in binary search spaces. *Complexity*, *2017*, 1–19.

Dahi, Z. A. E. M., Mezioud, C., & Draa, A. (2015). Binary bat algorithm: On the efficiency of mapping functions when handling binary problems using continuous-variable-based metaheuristics. In *IFIP International Conference on Computer Science and its Applications* (pp. 3–14). Springer.

Delisle, P., Gravel, M., Krajecki, M., Gagné, C., & Price, W. L. (2005). Comparing parallelization of an aco: Message passing vs. shared memory. In *International Workshop on Hybrid Metaheuristics* (pp. 1–11). Springer.

Derrac, J., García, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, *1*(1), 3–18.

Dorigo, M., & Stützle, T. (2019). Ant colony optimization: Overview and recent advances. In *Handbook of Metaheuristics* (pp. 311–351). Springer.

Dunn, O. J. (1964). Multiple comparisons using rank sums. *Technometrics*, *6*(3), 241–252.

González, P., Argüeso-Alejandro, P., Penas, D. R., Pardo, X. C., Saez-Rodriguez, J., Banga, J. R., & Doallo, R. (2019). Hybrid parallel multimethod hyperheuristic for mixed-integer dynamic optimization problems in computational systems biology. *The Journal of Supercomputing*, *75*(7), 3471–3498.

González, P., Osorio, R. R., Pardo, X. C., Banga, J. R., & Doallo, R. (2022). An efficient ant colony optimization framework for HPC environments. *Applied Soft Computing*, *114*, 108058.

González, P., Pardo, X. C., Doallo, R., & Banga, J. R. (2018). Implementing cloud-based parallel metaheuristics: An overview. *Journal of Computer Science Technology*, *18*(3), 228–238.

González, P., Prado-Rodriguez, R., Gábor, A., Saez-Rodriguez, J., Banga, J. R., & Doallo, R. (2022). Parallel ant colony optimization for the training of cell signaling networks. *Expert Systems with Applications*, *208*, 118199.

Hussien, A. G., Oliva, D., Houssein, E. H., Juan, A. A., & Yu, X. (2020). Binary whale optimization algorithm for dimensionality reduction. *Mathematics*, *8*(10), 1821.

Jang, S.-H., Roh, J.-H., Kim, W., Sherpa, T., Kim, J.-H., & Park, J.-B. (2011). A novel binary ant colony optimization: Application to the unit commitment problem of power systems. *Journal of Electrical Engineering and Technology*, *6*(2), 174–181.

Jimenez-Guardeño, J. M., Ortega-Prieto, A. M., Menendez Moreno, B., Maguire, T. J., Richardson, A., Diaz-Hernandez, J. I., Diez Perez, J., Zuckerman, M., Mercadal Playa, A., Cordero Deline, C., et al. (2022). Drug repurposing based on a quantum-inspired method versus classical fingerprinting uncovers potential antivirals against sars-cov-2. *PLoS Computational Biology*, *18*(7), e1010330.

Karimi-Mamaghan, M., Mohammadi, M., Meyer, P., Karimi-Mamaghan, A. M., & Talbi, E.-G. (2022). Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. *European Journal of Operational Research*, *296*(2), 393–422.

Kashef, S., & Nezamabadi-pour, H. (2015). An advanced aco algorithm for feature subset selection. *Neurocomputing*, *147*, 271–279.

Kong, M., & Tian, P. (2005). A binary ant colony optimization for the unconstrained function optimization problem. In *International Conference on Computational and Information Science* (pp. 682–687). Springer.

Kruskal, W. H., & Wallis, W. A. (1952). Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, *47*(260), 583–621.

Lewis, M. W., Verma, A., & Eckdahl, T. T. (2021). Qfold: A new modeling paradigm for the RNA folding problem. *Journal of Heuristics*, *27*(4), 695–717.

Li, Y., Li, T., & Liu, H. (2017). Recent advances in feature selection and its applications. *Knowledge and Information Systems*, *53*(3), 551–577.

Lozano, M., & García-Martínez, C. (2010). Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report. *Computers & Operations Research*, *37*(3), 481–497.

Lü, Z., Glover, F., & Hao, J.-K. (2010). A hybrid metaheuristic approach to solving the ubqp problem. *European Journal of Operational Research*, *207*(3), 1254–1262.

Mafarja, M., Aljarah, I., Heidari, A. A., Faris, H., Fournier-Viger, P., Li, X., & Mirjalili, S. (2018). Binary dragonfly optimization for feature selection using time-varying transfer functions. *Knowledge-Based Systems*, *161*, 185–204.

Matsumoto, M., & Nishimura, T. (1998). Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, *8*(1), 3–30.

Morris, M. K., Saez-Rodriguez, J., Sorger, P. K., & Lauffenburger, D. A. (2010). Logic-based models for the analysis of cell signaling networks. *Biochemistry*, *49*(15), 3216–3224.

Pal, A., & Maiti, J. (2010). Development of a hybrid methodology for dimensionality reduction in mahalanobis–taguchi system using mahalanobis distance and binary particle swarm optimization. *Expert Systems with Applications*, *37*(2), 1286–1293.

Pan, J.-S., Hu, P., & Chu, S.-C. (2021). Binary fish migration optimization for solving unit commitment. *Energy*, *226*, 120329.

Papaioannou, G., & Wilson, J. M. (2010). The evolution of cell formation problem methodologies based on recent studies (1997–2008): Review and directions for future research. *European Journal of Operational Research*, *206*(3), 509–521.

Pardo, X. C., Argüeso-Alejandro, P., González, P., Banga, J. R., & Doallo, R. (2020). Spark implementation of the enhanced scatter search metaheuristic: Methodology and assessment. *Swarm and Evolutionary Computation*, *59*, 100748.

Parsons, S. (2005). Ant Colony Optimization by Marco Dorigo and Thomas Stützle, MIT Press, 305 pp., ISBN 0-262-04219-3. *The Knowledge Engineering Review*, *20*(1), 92–93.

Penas, D. R., Banga, J. R., González, P., & Doallo, R. (2015). Enhanced parallel differential evolution algorithm for problems in computational systems biology. *Applied Soft Computing*, *33*, 86–99.

Penas, D. R., González, P., Egea, J. A., Banga, J. R., & Doallo, R. (2015). Parallel metaheuristics in computational biology: An asynchronous cooperative enhanced scatter search method. *Procedia Computer Science*, *51*, 630–639.

Penas, D. R., González, P., Egea, J. A., Doallo, R., & Banga, J. R. (2017). Parameter estimation in large-scale systems biology models: A parallel and self-adaptive cooperative strategy. *BMC Bioinformatics*, *18*(1), 1–24.

Potyagaylo, D., Cortés, E. G., Schulze, W. H., & Dössel, O. (2014). Binary optimization for source localization in the inverse problem of ecg. *Medical & Biological Engineering & Computing*, *52*(9), 717–728.

Punnen, A. P. (2022). *The Quadratic Unconstrained Binary Optimization Problem*. Ch. 2 (pp. 39–56). Springer.

Punnen, A. P., & Sotirov, R. (2022). Mathematical programming models and exact algorithms. In *The Quadratic Unconstrained Binary Optimization Problem* (pp. 139–185). Springer.

Randall, M., & Lewis, A. (2002). A parallel implementation of ant colony optimization. *Journal of Parallel and Distributed Computing*, *62*(9), 1421–1432.

Reddy, K. S., Panwar, L., Panigrahi, B., & Kumar, R. (2019). Binary whale optimization algorithm: A new metaheuristic approach for profit-based unit commitment problems in competitive electricity markets. *Engineering Optimization*, *51*(3), 369–389.

Sörensen, K., & Glover, F. (2013). Metaheuristics. *Encyclopedia of Operations Research and Management Science*, *62*, 960–970.

Starzec, M., Starzec, G., Byrski, A., Turek, W., & Pietak, K. (2020). Desynchronization in distributed ant colony optimization in hpc environment. *Future Generation Computer Systems*, *109*, 125–133.

Stützle, T. (1998). Parallelization strategies for ant colony optimization. In *International Conference on Parallel Problem Solving from Nature* (pp. 722–731). Springer.

Stützle, T., Dorigo, M., et al. (1999). Aco algorithms for the traveling salesman problem. *Evolutionary Algorithms in Engineering and Computer Science*, *4*, 163–183. https://www.researchgate.net/publication/2771967_ACO_Algorithms_for_the_Traveling_Salesman_Problem

Stützle, T., & Hoos, H. H. (2000). Max–min ant system. *Future Generation Computer Systems*, *16*(8), 889–914.

Taghian, S., & Nadimi-Shahraki, M. H. (2019). A binary metaheuristic algorithm for wrapper feature selection. *International Journal of Computer Science Engineering (IJCSE)*, *8*, 168–172.

Taghian, S., Nadimi-Shahraki, M. H., & Zamani, H. (2018). Comparative analysis of transfer function-based binary metaheuristic algorithms for feature selection. In: 2018 International Conference on Artificial Intelligence and Data Processing (IDAP). IEEE, pp. 1–6.

Teijeiro, D., Pardo, X. C., González, P., Banga, J. R., & Doallo, R. (2016). Implementing parallel differential evolution on spark. In *European Conference on the Applications of Evolutionary Computation* (pp. 75–90). Springer.

Teijeiro, D., Pardo, X. C., Penas, D. R., González, P., Banga, J. R., & Doallo, R. (2016). Evaluation of parallel differential evolution implementations on mapreduce and spark. In *European Conference on Parallel Processing* (pp. 397–408). Springer.

Twomey, C., Stützle, T., Dorigo, M., Manfrin, M., & Birattari, M. (2010). An analysis of communication policies for homogeneous multi-colony aco algorithms. *Information Sciences*, *180*(12), 2390–2404.

Wang, D., Tan, D., & Liu, L. (2018). Particle swarm optimization algorithm: An overview. *Soft Computing*, *22*, 387–408.

Weber, S., Nagy, A., Schüle, T., Schnörr, C., & Kuba, A. (2006). A benchmark evaluation of large-scale optimization approaches to binary tomography. In *International Conference on Discrete Geometry for Computer Imagery* (pp. 146–156). Springer.

Weise, T., Chen, Y., Li, X., & Wu, Z. (2020). Selecting a diverse set of benchmark instances from a tunable model problem for black-box discrete optimization algorithms. *Applied Soft Computing*, *92*, 106269.

Yuan, X., Nie, H., Su, A., Wang, L., & Yuan, Y. (2009). An improved binary particle swarm optimization for unit commitment problem. *Expert Systems with Applications*, *36*(4), 8049–8055.

Zhou, Y., He, F., Hou, N., & Qiu, Y. (2018). Parallel ant colony optimization on multi-core simd cpus. *Future Generation Computer Systems*, *79*, 473–487.

AUTHOR BIOGRAPHIES

Roberto Prado-Rodríguez received his BSc degree in Computer Science (2017) and MSc degree in High Performance Computing (2021) from the University of La Coruña (Spain). Since 2021 he is working as a predoctoral researcher. His work focuses on the improvement and parallelism of algorithms for combinatorial optimization in the field of bioinformatics.

Patricia González received the B.S. (1996), M.S. (1996) and Ph.D. (2001) degrees in physics from the University of Santiago de Compostela (Spain). Currently, she is an Associate Professor in the Department of Computer Engineering at the Universidade da Coruña (Spain). Her main area of research is high-performance computing, and more specifically, parallel and distributed computing, fault tolerance for parallel applications, and cloud computing. Within these lines of research she is co-author of more than 70 international papers. She has supervised 5 PhD students. She has participated in several national and international research networks and projects in the field of high-performance computing.

Julio R. Banga received a PhD in Chemical Engineering in 1991 from the University of Santiago de Compostela (Spain). During 1992 he was a postdoc at the University of California, Davis. During the period 1994-1996, he also spent several short periods as visiting researcher at the University of Pennsylvania, MIT and UC Davis. In 1995, he obtained a tenured researcher position at the Spanish Council for Scientific Research (CSIC). Since 2009, he is a CSIC Research Professor, and is currently leading the Computational Biology Lab at MBG-CSIC (Pontevedra, Spain). Julio has extensive experience in computational systems biology, with emphasis on reverse engineering (dynamic modelling) and dynamic optimisation (optimal control and optimal design) of biological systems. He is the author of more than 200 archival publications. He has supervised 15 PhD students (with two more ongoing) and has participated as principal investigator in 11 projects funded by the European Union and over 20 Spanish national projects. He was a member of the editorial committee of BMC Systems Biology (2007-2018). Currently he belongs to the Editorial Board of BMC Bioinformatics and to the IFAC Technical Committee on Control of Biotechnological Processes.

Ramón Doallo received the Ph.D. degree in physics from the Universidade de Santiago de Compostela (Spain). He has been a Full Professor of the Universidade da Coruña (Spain) since 1999, where he has been the Head of Computer Architecture Research Group. He has over 30 years of experience in research and development in high-performance computing (HPC), covering a wide range of topics such as compilers and programming languages for HPC, parallel and distributed algorithms and applications, management of HPC infrastructures, cluster and grid computing, processor architecture, computer graphics, and distributed geographic information systems. He has published more than 120 technical papers on these topics.

How to cite this article: Prado-Rodríguez, R., González, P., Banga, J. R., & Doallo, R. (2024). Improved cooperative Ant Colony Optimization for the solution of binary combinatorial optimization applications. *Expert Systems*, e13554. https://doi.org/10.1111/exsy.13554