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

Roberto Prado-Rodríguez[a], Patricia González[a], Julio R. Banga[b], Ramón Doallo[a]

Email addresses: roberto.prado@udc.es, patricia.gonzalez@udc.es, j.r.banga@csic.es, ramon.doallo@udc.es

[a] *CITIC, Computer Arquitecture Group, University of A Coruña, Spain,*
[b] *Computational Biology Lab, MBG-CSIC, Spanish National Research Council, Pontevedra, Spain,*

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 array 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, BiP-CACO) using a comprehensive benchmark framework, showcasing its performance and, most importantly, its flexibility as a successful all-purpose solver for binary combinatorial problems.

Keywords: Binary Combinatorial Optimization, Metaheuristic, Ant Colony Optimization, 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. Of 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 [1], dimensionality reduction [2, 3], unit commitment [4, 5], manufacturing [6], computational biology [7, 8, 9, 10, 11, 12], and medicine [13, 14] 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 [15].

Although in general these problems are NP-hard, a number of deterministic approaches have been developed for their resolution [16]. 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 [17, 18, 19]. Examples of recent papers exploting metaheuristics to solve problems from the classes listed above include [20], [21], [22] and [23] for feature selection, [24] for dimensionality reduction, [25] for unit commitment, [26] for manufacturing cell formation, or [27] for cell signaling networks.

There are many different metaheuristics used for general combinatorial optimization problems [28, 29, 25, 24, 30, 31]. One of the most popular is Ant Colony Optimization (ACO) [32]. 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 [33, 34, 35, 36].

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 ACO to handle challenging binary combinatorial problems. Our main objective has been to improve its performance, especially for large and difficult instances, without losing its features as a general solver that can be applied to a wide range of problems. To achieve this goal, the improvement of the cooperative scheme in a parallel multicolony implementation previously proposed by the authors [37] have been explored.

The structure of the paper is as follows. Section 2 presents the related work. Section 3 describes the proposed 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 from this study.

2. Related Work

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 [38, 39, 40, 41, 42]. A taxonomy of the different parallel models can be found in [43]. In this paper we adopt a multicolony model, where several colonies explore the search space isolately but including cooperation steps where information is exchanged among them. Other authors have previously explored this model for the parallelization of the ACO algorithm [44, 45, 46, 37]. 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 [47, 48, 49, 50, 51, 52, 53, 54]. Based on this previous experience, we have proposed in [27] 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 [55, 56].

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) [57]. 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.

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

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

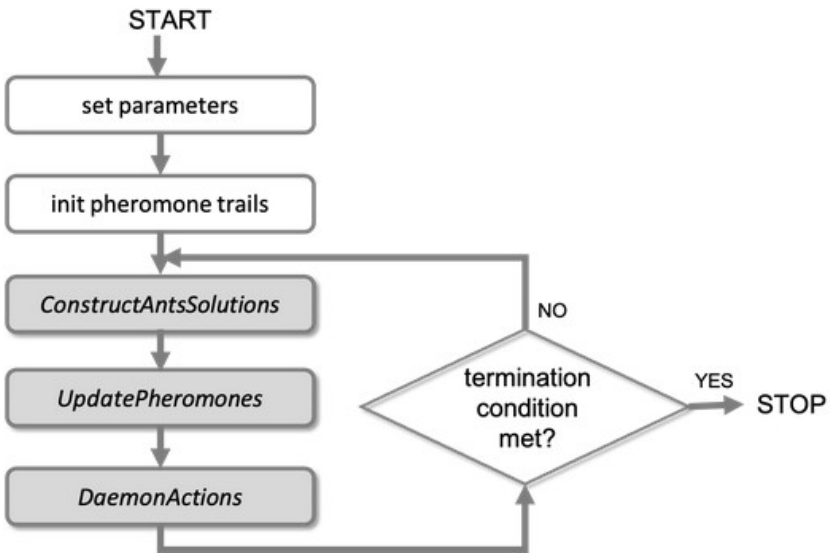

Figure 1: Basic scheme of the ACO algorithm

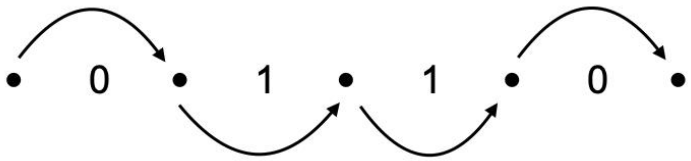

Figure 2: In a binary combinatorial problem, ants choose path 0 or path 1 at each step.

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:

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

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 [58] 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:

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

where $f(S^{best})$ 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 [27].

In ACO, when the algorithm gets stagnated, i.e. 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 [59], and ACO in particular [37]. 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 ususally stand out for approaches that include colony cooperation.

The cooperative approach proposed in this work is based on the model proposed in [37], adapted to a binary combinatorial problem in [27]. 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 [37] 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

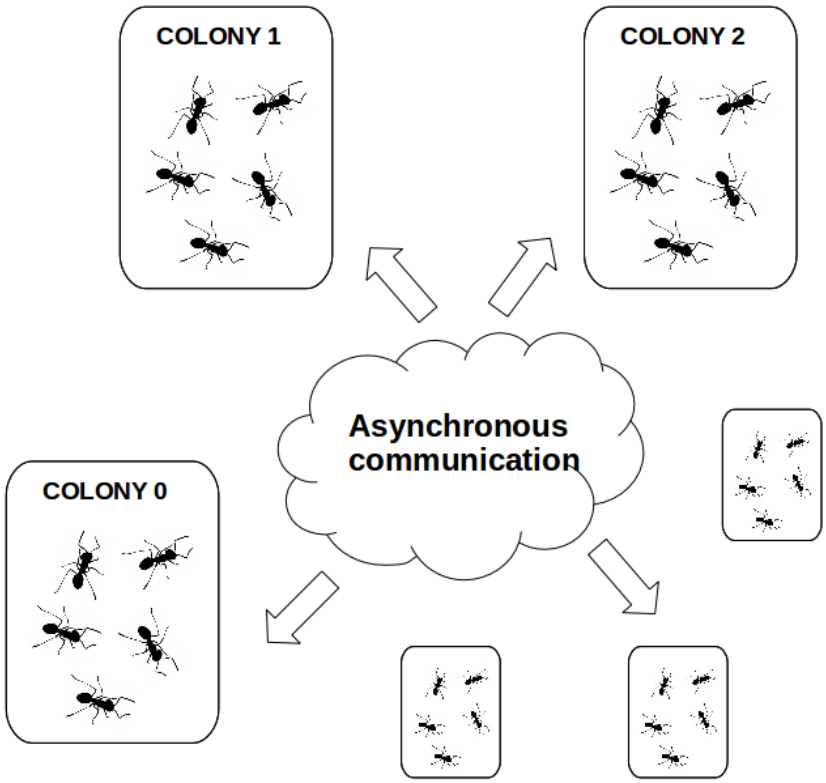

Figure 3: Scheme of a multiconolony implementation

waiting for information from other colonies. In the cooperative scheme proposed in [37], 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 promissing 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 promis-

ing 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 are 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

Algorithm 1: BiPCACO pseudocode.

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

Algorithm 2: Establishing initial maximum cooperation.

```
1  coop-index ← problem_size/NPROC
   // Note that min_cstall ← 0 means full cooperation.
2  min_cstall ← 0
3  if coop − index > 200 then
4      max_cstall ← 10% restart_iters
5  else if coop − index > 150 then
6      max_cstall ← 20% restart_iters
7  else if coop − index > 100 then
8      max_cstall ← 30% restart_iters
9  else if coop − index > 50 then
10     max_cstall ← 40% restart_iters
11 else
12     max_cstall ← 50% restart_iters
13 cstall ← max_cstall;
```

begining of the execution is, then, scarce. Thus, the algoritm 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 [60]. The W-Model is a tunable black-

Table 1: W-Model parameters for benchmarks 20 to 25.

Benchmark	Problem size	Neutrality	Epistasis	Ruggedness/Deceptiveness
20	640	4	130 (81%)	10000 (78%)
21	720	4	150 (83%)	12000 (75%)
22	1000	4	200 (80%)	25000 (80%)
23	640	4	130 (81%)	0
24	720	4	150 (83%)	0
25	1000	4	200 (80%)	0

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 [60] 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 labeled as 1 to 19 in the following.

Six additional challenging benchmarks labeled as 20 to 25 have been defined too. 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 [60].

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

- *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

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	-

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.

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.

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 [61] is used. A four-digit number is used 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: $cfreq \rightarrow \infty$, that is, a configuration without cooperation between colonies.

- M2: $cfreq \rightarrow 2$, 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 $n/25$.

- M3: $cfreq \rightarrow 1$, 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: $cfreq \rightarrow 1$ and $cstall = n/25$, that is, all the colonies incorporate foreigner solutions to their search when the number of stagnate iterations is $n/25$.

- M5: $cfreq \rightarrow 1$ 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: $cfreq \rightarrow 1$ and $cstall = 0$, that is, all the colonies incorporate foreigner solutions everytime 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.

Figure 4 summarizes the results obtained for experiments using the optimum value as stopping criterium. This figure displays the average of the execution time in seconds, 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 to 22) exhibit features that make them very difficult to solve. 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, i.e. the search is oriented smoothly to the global minimum. Since the algorithm does not get stuck, it does not need cooperation, and therefore in problems 23 and 24 there is hardly any difference in times 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.

		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

Figure 4: Comparison of different cooperative configurations in BiPCACO. Average time (in seconds) achieved in 100 runs of experiments using optimum value as stopping criterium, for the most challenging benchmarks using 4, 8, 12, 24 and 64 processes.

Certain conclusions can be drawn from the results in Figure 4. 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.

Results in Figure 4 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:

- (1) Amount of processes: the smaller the number of processes, the higher the cooperation between colonies should be.
- (2) Multimodality: The higher the bias towards the multimodal landscape, the larger the probability that the algorithm will get stuck when there is an excess of cooperation.
- (3) 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 attractive. Moreover, results of Figure 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. To illustrate this, logarithmic scale plots for benchmarks 20 and 22 are shown in Figure 5, Figure 6, Figure 7 and Figure 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,

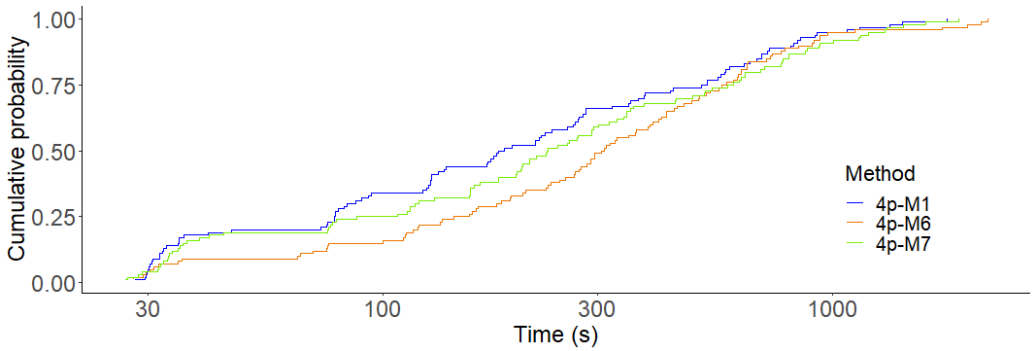

Figure 5: Cumulative probability of reaching the optimum in benchmark 20 using 4 processes, where M1-7 indicates different cooperation schemes as explained in section 5.3.

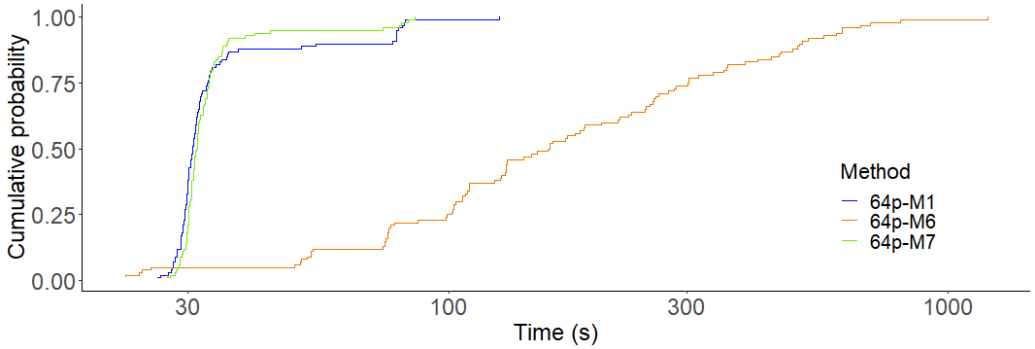

Figure 6: Cumulative probability of reaching the optimum in benchmark 20 using 64 processes, where M1-7 indicates different cooperation schemes as explained in section 5.3.

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.

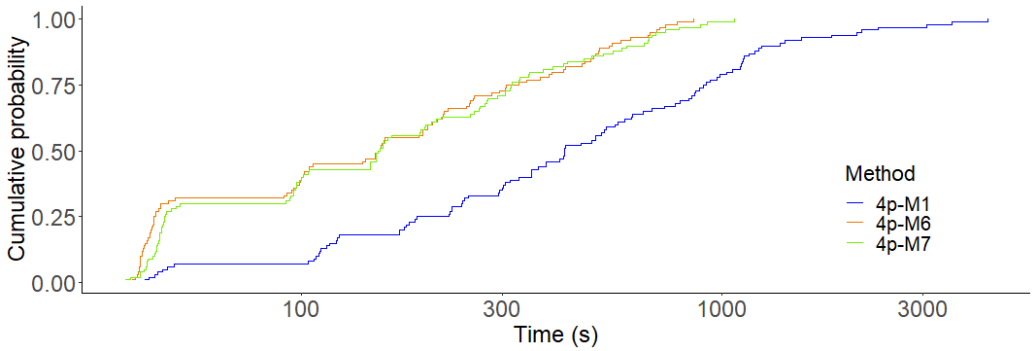

Figure 7: Cumulative probability of reaching the optimum in benchmark 22 using 4 processes, where M1-7 indicates different cooperation schemes as explained in section 5.3.

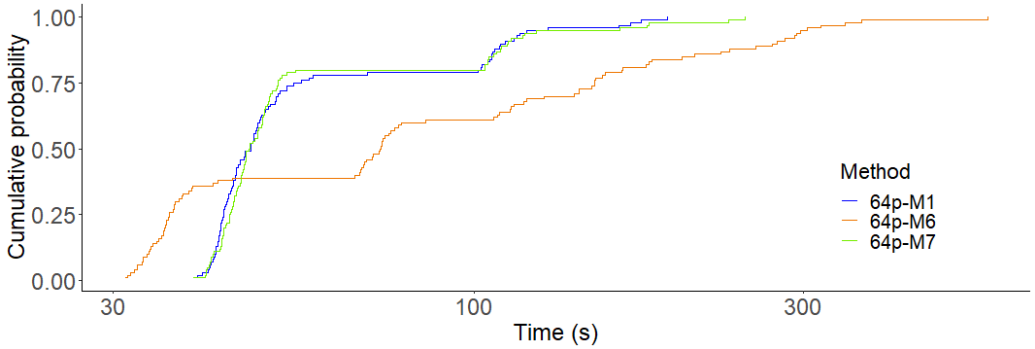

Figure 8: Cumulative probability of reaching the optimum in benchmark 22 using 64 processes, where M1-7 indicates different cooperation schemes as explained in section 5.3.

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.

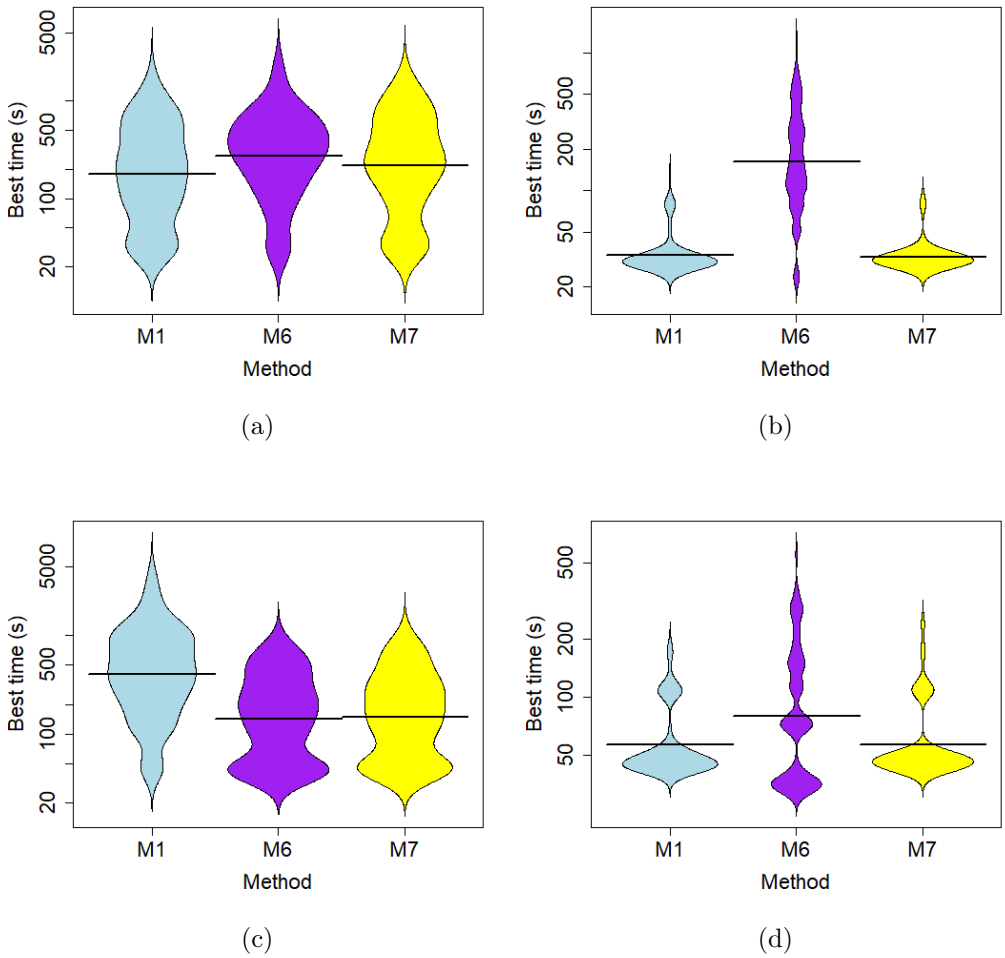

Figure 9: Beanplots with the distribution of the execution time for (a) benchmark 20 using 4 processes, (b) benchmark 20 using 64 processes, (c) benchmark 22 using 4 processes and (d) benchmark 22 using 64 processes, where M1-7 indicates different cooperation schemes as explained in section 5.3.

5.4. Comprehensive assessment

To perform comprehensive assessment of the proposed BiPCACO algorithm, a comparison has been carried out with the results presented in [60] for 17 different algorithms, solving the 19 benchmarks described in 5.1. To perform a fair comparison, the same metric as in [60] 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 [60], aggregating the number of 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 [60] together with the results of the sequential ACO described in Section 3 and BiPCACO proposal.

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 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. Conclusions

This paper introduces an improved multicolony ACO for binary combinatorial problems. This novel method aims to overcome the main handicap of such algorithm, i.e., its tendency to get stuck in local minima. We introduce a cooperative parallel ACO strategy in which colonies share promising solutions with each other, but only incorporate them into their search if certain conditions are met. This approach maintains diversity, making it easier to move out of local minima. Based on tests with different cooperative configurations, a self-tuned version was developed that adjusts the parameters of

	Benchmark																		
Algorithm	**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 [60], where ACO stands for sequential ACO and BiPCACO(N) for BiPCACO algorithm with N colonies.

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 was 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 cooperative strategy has significantly reduced the average time on all tests, especially on large-scale, complex problems similar to those encountered in real-world scenarios.

Acknowledgments

This work was supported by the Ministry of Science and Innovation of Spain (PID2019-104184RB-I00 / AEI / 10.13039/501100011033), and by

Xunta de Galicia and FEDER funds of the EU (Centro de Investigación de Galicia accreditation 2019–2022, ref. ED431G 2019/01; 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 Supercomputing Center (CESGA) for the access to its facilities.

References

[1] Yun Li, Tao Li, and Huan Liu, "Recent advances in feature selection and its applications," *Knowledge and Information Systems*, vol. 53, no. 3, pp. 551–577, 2017.

[2] Miguel A Carreira-Perpinán, "A review of dimension reduction techniques," *Department of Computer Science. University of Sheffield. Tech. Rep. CS-96-09*, vol. 9, pp. 1–69, 1997.

[3] Avishek Pal and Jhareswar Maiti, "Development of a hybrid methodology for dimensionality reduction in mahalanobis–taguchi system using mahalanobis distance and binary particle swarm optimization," *Expert Systems with Applications*, vol. 37, no. 2, pp. 1286–1293, 2010.

[4] Ross Baldick, "The generalized unit commitment problem," *IEEE Transactions on Power Systems*, vol. 10, no. 1, pp. 465–475, 1995.

[5] Xiaohui Yuan, Hao Nie, Anjun Su, Liang Wang, and Yanbin Yuan, "An improved binary particle swarm optimization for unit commitment problem," *Expert Systems with applications*, vol. 36, no. 4, pp. 8049–8055, 2009.

[6] Grammatoula Papaioannou and John M Wilson, "The evolution of cell formation problem methodologies based on recent studies (1997–2008): Review and directions for future research," *European journal of operational research*, vol. 206, no. 3, pp. 509–521, 2010.

[7] Melody K Morris, Julio Saez-Rodriguez, Peter K Sorger, and Douglas A Lauffenburger, "Logic-based models for the analysis of cell signaling networks," *Biochemistry*, vol. 49, no. 15, pp. 3216–3224, 2010.

[8] Julio R Banga, "Optimization in computational systems biology," *BMC systems biology*, vol. 2, no. 1, pp. 1–7, 2008.

[9] Matthew B Biggs and Jason A Papin, "Managing uncertainty in metabolic network structure and improving predictions using ensemblefba," *PLoS computational biology*, vol. 13, no. 3, pp. e1005413, 2017.

[10] Jose M Jimenez-Guardeño, Ana Maria Ortega-Prieto, Borja Menendez Moreno, Thomas JA Maguire, Adam Richardson, Juan Ignacio Diaz-Hernandez, Javier Diez Perez, Mark Zuckerman, Albert Mercadal Playa, Carlos Cordero Deline, et al., "Drug repurposing based on a quantum-inspired method versus classical fingerprinting uncovers potential antivirals against sars-cov-2," *PLoS computational biology*, vol. 18, no. 7, pp. e1010330, 2022.

[11] Somayeh Bakhteh, Alireza Ghaffari-Hadigheh, and Nader Chaparzadeh, "Identification of minimum set of master regulatory genes in gene regulatory networks," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 17, no. 3, pp. 999–1009, 2018.

[12] Mark W Lewis, Amit Verma, and Todd T Eckdahl, "Qfold: a new modeling paradigm for the rna folding problem," *Journal of Heuristics*, vol. 27, no. 4, pp. 695–717, 2021.

[13] Stefan Weber, Antal Nagy, Thomas Schüle, Christoph Schnörr, and Attila Kuba, "A benchmark evaluation of large-scale optimization approaches to binary tomography," in *International Conference on Discrete Geometry for Computer Imagery*. Springer, 2006, pp. 146–156.

[14] Danila Potyagaylo, Elisenda Gil Cortés, Walther HW Schulze, and Olaf Dössel, "Binary optimization for source localization in the inverse problem of ECG," *Medical & biological engineering & computing*, vol. 52, no. 9, pp. 717–728, 2014.

[15] Abraham P Punnen, "The quadratic unconstrained binary optimization problem," Tech. Rep., Springer, 2022.

[16] Abraham P Punnen and Renata Sotirov, "Mathematical programming models and exact algorithms," in *The Quadratic Unconstrained Binary Optimization Problem*, pp. 139–185. Springer, 2022.

[17] Zhipeng Lü, Fred Glover, and Jin-Kao Hao, "A hybrid metaheuristic approach to solving the ubqp problem," *European Journal of Operational Research*, vol. 207, no. 3, pp. 1254–1262, 2010.

[18] Kenneth Sörensen and Fred Glover, "Metaheuristics," *Encyclopedia of operations research and management science*, vol. 62, pp. 960–970, 2013.

[19] Prachi Agrawal, Hattan F Abutarboush, Talari Ganesh, and Ali Wagdy Mohamed, "Metaheuristic algorithms on feature selection: A survey of one decade of research (2009-2019)," *IEEE Access*, vol. 9, pp. 26766–26791, 2021.

[20] Majdi Mafarja, Ibrahim Aljarah, Ali Asghar Heidari, Hossam Faris, Philippe Fournier-Viger, Xiaodong Li, and Seyedali Mirjalili, "Binary dragonfly optimization for feature selection using time-varying transfer functions," *Knowledge-Based Systems*, vol. 161, pp. 185–204, 2018.

[21] Qasem Al-Tashi, Said Jadid Abdul Kadir, Helmi Md Rais, Seyedali Mirjalili, and Hitham Alhussian, "Binary optimization using hybrid grey wolf optimization for feature selection," *Ieee Access*, vol. 7, pp. 39496–39508, 2019.

[22] Shokooh Taghian and Mohammad H Nadimi-Shahraki, "A binary metaheuristic algorithm for wrapper feature selection," *Int. J. Comput. Sci. Eng.(IJCSE)*, vol. 8, pp. 168–172, 2019.

[23] Sankalap Arora and Priyanka Anand, "Binary butterfly optimization approaches for feature selection," *Expert Systems with Applications*, vol. 116, pp. 147–160, 2019.

[24] Abdelazim G Hussien, Diego Oliva, Essam H Houssein, Angel A Juan, and Xu Yu, "Binary whale optimization algorithm for dimensionality reduction," *Mathematics*, vol. 8, no. 10, pp. 1821, 2020.

[25] Srikanth Reddy K, Lokesh Panwar, BK Panigrahi, and Rajesh Kumar, "Binary whale optimization algorithm: a new metaheuristic approach for profit-based unit commitment problems in competitive electricity markets," *Engineering Optimization*, vol. 51, no. 3, pp. 369–389, 2019.

[26] Jeng-Shyang Pan, Pei Hu, and Shu-Chuan Chu, "Binary fish migration optimization for solving unit commitment," *Energy*, vol. 226, pp. 120329, 2021.

[27] Patricia González, Roberto Prado-Rodriguez, Attila Gábor, Julio Saez-Rodriguez, Julio R Banga, and Ramón Doallo, "Parallel ant colony optimization for the training of cell signaling networks," *Expert Systems with Applications*, p. 118199, 2022.

[28] Broderick Crawford, Ricardo Soto, Gino Astorga, José García, Carlos Castro, and Fernando Paredes, "Putting continuous metaheuristics to work in binary search spaces," *Complexity*, vol. 2017, 2017.

[29] Manuel Lozano and Carlos García-Martínez, "Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report," *Computers & Operations Research*, vol. 37, no. 3, pp. 481–497, 2010.

[30] Zakaria Abd El Moiz Dahi, Chaker Mezioud, and Amer Draa, "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*. Springer, 2015, pp. 3–14.

[31] Shokooh Taghian, Mohammad H Nadimi-Shahraki, and Hoda Zamani, "Comparative analysis of transfer function-based binary metaheuristic algorithms for feature selection," in *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*. IEEE, 2018, pp. 1–6.

[32] Marco Dorigo and Thomas Stützle, "Ant colony optimization: overview and recent advances," *Handbook of metaheuristics*, pp. 311–351, 2019.

[33] Min Kong and Peng Tian, "A binary ant colony optimization for the unconstrained function optimization problem," in *International Conference on Computational and Information Science*. Springer, 2005, pp. 682–687.

[34] Ahmed Al-Ani, "Feature subset selection using ant colony optimization," *International journal of computational intelligence*, 2005.

[35] Se-Hwan Jang, Jae-Hyung Roh, Wook Kim, Tenzi Sherpa, Jin-Ho Kim, and Jong-Bae Park, "A novel binary ant colony optimization: Application to the unit commitment problem of power systems," *Journal of Electrical Engineering and Technology*, vol. 6, no. 2, pp. 174–181, 2011.

[36] Shima Kashef and Hossein Nezamabadi-pour, "An advanced aco algorithm for feature subset selection," *Neurocomputing*, vol. 147, pp. 271–279, 2015.

[37] Patricia González, Roberto R Osorio, Xoan C Pardo, Julio R Banga, and Ramón Doallo, "An efficient ant colony optimization framework for HPC environments," *Applied Soft Computing*, vol. 114, pp. 108058, 2022.

[38] Thomas Stützle, "Parallelization strategies for ant colony optimization," in *International Conference on Parallel Problem Solving from Nature*. Springer, 1998, pp. 722–731.

[39] Marcus Randall and Andrew Lewis, "A parallel implementation of ant colony optimization," *Journal of Parallel and Distributed Computing*, vol. 62, no. 9, pp. 1421–1432, 2002.

[40] Pierre Delisle, Marc Gravel, Michaël Krajecki, Caroline Gagné, and Wilson L Price, "Comparing parallelization of an aco: message passing vs. shared memory," in *International Workshop on Hybrid Metaheuristics*. Springer, 2005, pp. 1–11.

[41] José M Cecilia, José M García, Andy Nisbet, Martyn Amos, and Manuel Ujaldón, "Enhancing data parallelism for ant colony optimization on gpus," *Journal of Parallel and Distributed Computing*, vol. 73, no. 1, pp. 42–51, 2013.

[42] Yi Zhou, Fazhi He, Neng Hou, and Yimin Qiu, "Parallel ant colony optimization on multi-core simd cpus," *Future Generation Computer Systems*, vol. 79, pp. 473–487, 2018.

[43] Enrique Alba, Gabriel Luque, and Sergio Nesmachnow, "Parallel metaheuristics: recent advances and new trends," *International Transactions in Operational Research*, vol. 20, no. 1, pp. 1–48, 2013.

[44] Colin Twomey, Thomas Stützle, Marco Dorigo, Max Manfrin, and Mauro Birattari, "An analysis of communication policies for homogeneous multi-colony aco algorithms," *Information Sciences*, vol. 180, no. 12, pp. 2390–2404, 2010.

[45] Ling Chen, Hai-Ying Sun, and Shu Wang, "A parallel ant colony algorithm on massively parallel processors and its convergence analysis for the travelling salesman problem," *Information Sciences*, vol. 199, pp. 31–42, 2012.

[46] Mateusz Starzec, Grażyna Starzec, Aleksander Byrski, Wojciech Turek, and Kamil Pietak, "Desynchronization in distributed ant colony optimization in hpc environment," *Future Generation Computer Systems*, vol. 109, pp. 125–133, 2020.

[47] David R Penas, Julio R Banga, Patricia González, and Ramon Doallo, "Enhanced parallel differential evolution algorithm for problems in computational systems biology," *Applied Soft Computing*, vol. 33, pp. 86–99, 2015.

[48] David R Penas, Patricia González, José A Egea, Julio R Banga, and Ramón Doallo, "Parallel metaheuristics in computational biology: An asynchronous cooperative enhanced scatter search method," *Procedia Computer Science*, vol. 51, pp. 630–639, 2015.

[49] Diego Teijeiro, Xoán C Pardo, Patricia González, Julio R Banga, and Ramón Doallo, "Implementing parallel differential evolution on spark," in *european conference on the applications of evolutionary computation*. Springer, 2016, pp. 75–90.

[50] Diego Teijeiro, Xoán C Pardo, David R Penas, Patricia González, Julio R Banga, and Ramón Doallo, "Evaluation of parallel differential evolution implementations on mapreduce and spark," in *European Conference on Parallel Processing*. Springer, 2016, pp. 397–408.

[51] David R Penas, Patricia González, Jose A Egea, Ramón Doallo, and Julio R Banga, "Parameter estimation in large-scale systems biology models: a parallel and self-adaptive cooperative strategy," *BMC bioinformatics*, vol. 18, no. 1, pp. 1–24, 2017.

[52] Patricia González, Xoán C Pardo, Ramón Doallo, and Julio R Banga, "Implementing cloud-based parallel metaheuristics: an overview," 2018.

[53] Patricia González, Pablo Argüeso-Alejandro, David R Penas, Xoan C Pardo, Julio Saez-Rodriguez, Julio R Banga, and Ramón Doallo, "Hybrid parallel multimethod hyperheuristic for mixed-integer dynamic optimization problems in computational systems biology," *The Journal of Supercomputing*, vol. 75, no. 7, pp. 3471–3498, 2019.

[54] Xoán C Pardo, Pablo Argüeso-Alejandro, Patricia González, Julio R Banga, and Ramón Doallo, "Spark implementation of the enhanced scatter search metaheuristic: Methodology and assessment," *Swarm and Evolutionary Computation*, vol. 59, pp. 100748, 2020.

[55] Simon Parsons, "Ant Colony Optimization by Marco Dorigo and Thomas Stützle, MIT Press, 305 pp., ISBN 0-262-04219-3," *The Knowledge Engineering Review*, vol. 20, no. 1, pp. 92–93, 2005.

[56] Christian Blum, "Ant colony optimization: Introduction and recent trends," *Physics of Life Reviews*, vol. 2, no. 4, pp. 353–373, 2005.

[57] Thomas Stützle, Marco Dorigo, et al., "Aco algorithms for the traveling salesman problem," *Evolutionary algorithms in engineering and computer science*, vol. 4, pp. 163–183, 1999.

[58] Thomas Stützle and Holger H Hoos, "Max–min ant system," *Future generation computer systems*, vol. 16, no. 8, pp. 889–914, 2000.

[59] Enrique Alba, *Parallel metaheuristics: a new class of algorithms*, John Wiley & Sons, 2005.

[60] Thomas Weise, Yan Chen, Xinlu Li, and Zhize Wu, "Selecting a diverse set of benchmark instances from a tunable model problem for black-box discrete optimization algorithms," *Applied Soft Computing*, vol. 92, pp. 106269, 04 2020.

[61] Makoto Matsumoto and Takuji Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 8, no. 1, pp. 3–30, 1998.