Using a Hybrid Honey Bees Mating Optimisation Algorithm for solving SONET/SDH Design Problems

Anabela Moreira Bernardino
Eugénia Moreira Bernardino
Computer Science and Communication Research Centre, School of Technology and Management, Polytechnic Institute of Leiria
Leiria, Portugal
{anabela.bernardino, eugenia.bernardino}@ipleiria.pt

Juan M. Sánchez-Pérez
Juan A. Gómez-Pulido
Miguel A. Vega-Rodríguez
Department of Technologies of Computers and Communications, Polytechnic School, University of Extremadura
Cáceres, Spain
{sanperez, jangomez, mavega}@unex.es

ABSTRACT
In this paper we propose a hybrid Honey Bees Mating Optimisation (HBMO) algorithm to solve two problems that arise in the design of optical telecommunication networks known as SONET/SDH Ring Assignment Problem (SRAP) and Intraring Synchronous Optical Network Design Problem (IDP). In SRAP the objective is to minimise the number of rings. In IDP the objective is to minimise the number of Add-Drop Multiplexers (ADMs). Both problems are subject to a ring capacity constraint. HBMO algorithm simulates the mating process of real honey bees. We apply a hybridisation of HBMO to solve these two combinatorial optimisation problems. The feasibility of Hybrid HBMO is demonstrated and compared with the solutions obtained by other algorithms from literature.

Categories and Subject Descriptors
C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design – *Circuit-switching networks, Network communications.*

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search – *Heuristic methods.*

General Terms
Algorithms, Design.

Keywords
Communication Networks, SONET Design Problems, Honey Bees Mating Optimisation algorithm, Bio-inspired algorithms.

1. INTRODUCTION
Nowadays, Synchronous Optical NETworking (SONET) in North America, and Synchronous Digital Hierarchy (SDH) in Europe and Japan are the current transmission and multiplexing standards for high speed signals within the carrier infrastructure.

In this paper we consider two problems that arise in the design of optical telecommunication networks, specifically SRAP and IDP, known to be NP-hard.

In SRAP each customer has to be assigned to exactly one SONET ring and a special ring called federal ring interconnects the other rings together through a special device, the Digital Cross Connect (DXC), which is the most costly network component. In this problem a capacity constraint on each ring is imposed. The capacity of each bidirectional ring of the network, including the federal ring has to accommodate the sum of bandwidth requests between all pairs of nodes connected by that ring. The problem is to find a feasible assignment of the customers minimising the total number of rings used (i.e. DXCs used) [3, 10, 15].

In IDP, customers can be connected to more than one ring. If two customers want to communicate, they have to be connected to the same ring. In this case, the DXCs and the federal ring are not necessary, however the number of ADMs increase substantially in this topology. Since ADMs are the most expensive component in this network topology, it is important to obtain the smallest number of ADMs to reduce the network cost. Similarly to SRAP, the capacity of each ring is limited [3, 11, 12, 15].

SRAP and IDP are NP-hard combinatorial optimisation problems and to solve them we propose a Honey Bees Mating Optimisation (HBMO) algorithm, combined with a Local Search (LS) method. HBMO is inspired by the marriage process in real honey bees [1, 2]. A honey bee colony typically is composed of three kinds of bees: the queen, the drones (the fathers of the colony) and the workers (specialised in brood care). We use two workers which perform a neighbourhood search and a LS in order to achieve a very fast and efficient algorithm, able to reduce the computational time, making the algorithm suitable for solving very large scaled problems in short computational times.

We compare our results with several algorithms used in literature to solve the same problems, namely: Diversification by Multiple Neighbourhoods (DMN and DMN2), and Hybrid Scatter Search (HSS). For the SRAP we also compare our results with: standard Genetic Algorithm (GA), GA with Evolutionary Path-Relinking (GA-EvPR), Greedy Randomised Adaptive Search Procedure (GRASP), GRASP with Path-Relinking (GRASP-PR), and Memetic Algorithm with Vocabulary Building (MA+VB).

The paper is structured as follows: in Section 2 we present the definition of the problems; in Section 3 we describe the implemented HSS algorithm; in Section 4 we present the studied instances; in Section 5 we discuss the computational results obtained and in Section 6 we report about the conclusions.

ISABEL '11, October 26 - 29 2011, Barcelona, Spain.

2. SONET/SDH DESIGN PROBLEMS

In our work we consider the two topologies described by Aringhieri and Dell'Amico [3] and we use the same model based on graph theory for SRAP and IDP. In both topologies, the objective is to minimise the network cost so as to guarantee that the customer's demands, in term of bandwidth, are satisfied.

Considering a set of n customers and a symmetric traffic matrix [duv], where u, $v = \{1, ..., n\}$ and $u \neq v$, each matrix element specifies the amount of traffic between customers u and v. Note that $duv = dvu$ and that $duu = 0$. So, given an undirected graph $G = (V,E)$, the node set V contains one node for each customer and the edge set E contains one edge (u, v) for each pair of customers u, v such that $duv > 0$.

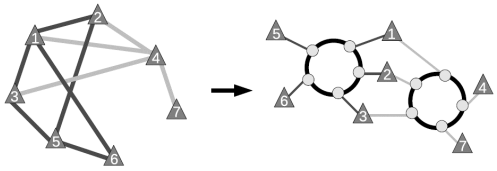

Figure 1. Relation between node partitioning and network topology [15].

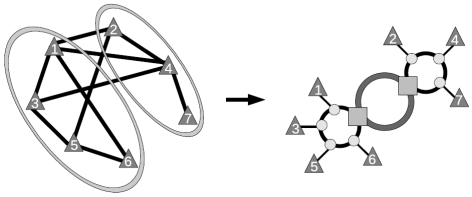

Figure 2. Relation between edge partitioning and network topology [15].

The SRAP and IDP correspond to two different partitioning of the above graph, subject to capacity constraints. In particular, SRAP involves a node partitioning (see Figure 1) and IDP (see Figure 2) an edge partitioning.

2.1 SRAP

Formally, let $V_1, V_2, \ldots V_k$, be a partitioning of V in k subsets, the corresponding SRAP network is obtained by defining k local rings and a federal ring. All customers of the subset V_i are associated to the i-th local ring by means of an ADM and to the federal ring that uses a DXC to connect each local ring. As a result the corresponding SRAP network uses n ADMs and k DXCs (see Figure 1).

Solving SRAP corresponds to find the partition $V_1, \ldots V_k$ that minimises k (number of rings), taking into account that the traffic volume on any ring is limited by the link capacity, called B. In other words, the total traffic demands of all customers connected to a ring must be lower or equal to the bandwidth (see Eq. 1) and moreover the total traffic of the federal ring cannot be larger than the bandwidth B (see Eq. 2).

$$\sum_{u \in V_i} \sum_{v \in V, v \neq u} d_{uv} \leq B, \qquad \forall i = 1,...k \tag{1}$$

$$\sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \sum_{u \in V_i} \sum_{v \in V_j} d_{uv} \leq B \tag{2}$$

2.2 IDP

In IDP, it is not necessary to assign each customer to a particular ring since customers can be connected to several rings. For this problem the used model is based on the edges of the graph, where a subset of the partition corresponds to a ring (see Figure 2).

Given a partition of P into k subsets $P_1, P_2, \ldots P_k$, the IDP network can be obtained by defining k rings and connecting each customer of $V(E_i)$ to the i-th ring by means of an ADM. In this case, the DXC are no longer needed and neither is the federal ring. Solving IDP corresponds to find the partition $P_1, P_2, \ldots P_k$ for each edge in order to minimise the number of ADMs [11, 12].

In this problem, like in SRAP, the traffic volume inside each ring is also limited by the link capacity B (see Eq. 3).

$$\sum_{(u,v) \in P_i} d_{uv} \leq B, \quad \forall i = 1,...,k \tag{3}$$

SRAP and IDP solutions are represented using integer vectors. Each vector position corresponds to a customer (in SRAP) or edge (in IDP) and the value of that position corresponds to the ring / partition which it is connected to. We assume that weights of the traffic demands cannot be split.

3. HYBRID HBMO

The mating process of honey bees gave rise to the HBMO algorithm. Honey bees mating can be considered as a typical swarm-based approach to optimisation, in which the search algorithm is inspired by the process of mating in real honey bees. Abbass [1, 2] developed an optimisation algorithm based on the honey bees mating process.

The standard HBMO algorithm consists of the following main steps:

1. Initial population: the drones are randomly generated;
2. Mating Flight: a queen (best solution) selects drones probabilistically to form the spermatheca (list of drones);
3. Creation of broods: a drone is randomly selected from the spermatheca and a new brood is created by crossing the drone's genotypes with the queen's genotype;
4. Workers: a worker is selected to conduct local search on each brood;
5. Adaptation of workers' fitness: the fitness of a worker is updated based on the amount of improvements achieved on broods;
6. Replacement of weaker queens by fitter broods. If one brood does not replace the queen, then in the next mating flight of the queen this brood will be one of the drones.

We have adopted a different model proposed by Bernardino et al. [8]. Our algorithm only works with one queen. In nature there is only one queen at each moment in a bee colony.

The mating flight can be considered as a set of transitions in the environment where the queen moves among different states in some speed and mates probabilistically with the drone at each step. A drone mates with a queen probabilistically using an annealing function as follows [1], [2]:

$$ProbQD = exp(-AvgQDFit()/speedQ) \tag{4}$$

ProbQD is the probability of adding the sperm of drone D to the queen's spermatheca (i.e., the probability of a successful mating);

AvgQDFit() is the absolute difference between the drone's fitness and the queen's fitness; and *speedQ* is the queen's speed at a time.

We verify that this probability will exclude the most part of the drones since the differences in terms of fitness values (queen and drone) can be very high. It was changed the equation (Eq. 4) to consider the absolute average difference between all drones and the queen (*AvgTotalFit*). A drone with a smaller fitness value will have more probability to be selected [8].

$$ProbQD = exp((-AvgQ\,DFit()/Avg\,TotalFit())/speedQ) \quad (5)$$

The probability of mating is high either when the queen is still at the beginning of her mating flight, therefore with higher speed, or when the drone's fitness is good compared to the queen's and to other drones' fitness in the population. After each transition in space, the queen's speed and energy decays (see subsection 3.4).

In nature, the workers are specialised in brood care and are not separate members of the population. In HBMO the workers are used as LS procedures in order to improve the broods [1], [2]. In our hybrid implementation, we use two workers to improve the broods' quality. The two workers are always applied to each brood. For that reason, it was deleted the 5th step from our algorithm [8]. The first worker performs a neighbourhood search and the second a LS.

In the 6th step, the best broods that do not replace the queen will be selected as drones in the next matting flight. In our implementation the number of broods can be smaller than the number of drones. In case of *numBroods<numDrones* all the broods (except the brood that replaces the queen) will be selected as drones to the next mating flight. The remaining drones are created by means of the deterministic algorithm used to create the initial population.

Our Hybrid HBMO algorithm consists of the following steps:

Hybrid HBMO algorithm

```
Initialisation of parameters
Generation of initial population of drones
Evaluation of Drones
Select best solution Q (queen)
mf=0
WHILE (mf<numMatingFlights)
   Matting Flight:
      speedQ=generateSpeed(0.1, 1)
      energyQ=generateEnergy(0.5, 1)
      elemS=0
      WHILE (elemS<sizeSper and energyQ>0)
         D=selectDrone()
         probQD=exp((-AvgQDFit()/AvgTotalFit())
                                          /speedQ)
         IF (rand()<=probQD)
            addSpermatheca(D)
            elemS=elemS+1
         speedQ=speedQ*alfa
         energyQ=energyQ-energyReduce
   Creation of broods and Workers Phase:
      b=0
      WHILE (b<numBroods)
         D=selectDroneSpermatheca()
         addBrood(crossover(Q,D))
         applyWorker1()
         aplplyWorker2()
         b=b+1
   Best brood selection:
      Evaluation of Broods
      orderBroods()
      bestB=selectBestBrood()
      b=0
      IF (fitness(bestB)<fitness(Q))
         Q=bestB /*replace queen*/
         b=1
   Drones replacement:
      d=0
      WHILE (d<numDrones and b<numBroods)
         replaceDrone(d, selectBrood(b))
         b=b+1
         d=d+1
      WHILE (d<numDrones)
         generateGreedyDrone(d)
         d=d+1
   mf++
```

Parameters (see subsection 3.1): *numMatingFlights*, *alfa*, *energyReduce*, *sizeSper*, *numDrones* and *numBroods*.
Variables: *mf* – number of matting flights at a moment; *energyQ* – queen's energy; *speedQ* – queen's speed; *elemS* – number of drones in the spermatheca; *b* – number of broods and *d* – number of drones.

The next subsections describe each step of the algorithm in detail.

3.1 Initialisation of Parameters

The following parameters, must be defined by the user: (1) *numMatingFlights* – number of iterations or number of mating flights; (2) *alfa* – factor between *0* and *1*; (3) *energyReduce* – amount of energy reduction after each transition; (4) *sizeSper* – spermatheca size representing the maximum number of mating in a single mating flight; (5) *numDrones* – number of possible fathers; (6) *numBroods* – number of broods that will be born by the queen; (7) *cr* – crossover operator; and (8) *nm* – number of modifications.

3.2 Generation of initial population of drones

For SRAP and IDP the solutions can be created randomly or in a deterministic form. For the deterministic form we use the same algorithm proposed by Bernardino et al. [9]. It is a simple heuristic that builds balanced solutions based on the number of customers for each ring. We consider an initial $k = k_{lb} + 1$ (Eq. 6).

$$k_{lb} = \left\lceil \sum_{u=1}^{n-1} \sum_{v=u+1}^{n} d_{uv} \Big/ B \right\rceil \quad (6)$$

3.3 Evaluation of bees

For SRAP and IDP, Aringhieri and Dell'Amico [3] introduced four objective functions. In our work we used the fitness function z_5 (Eq. 7) introduced by Pelleau et al. [15]. The authors have implemented the five fitness functions. The z_5 function finds better solutions than the other ones.

$$z_5 = z_0 + \sum_{p \in \ partitions} violations(p) \quad (7)$$

$$violations(p) = \begin{cases} capacity(p) - B & if\ exceed\ B \\ 0 & otherwise \end{cases} \quad (8)$$

Let z_0 be the basic objective function counting the number of rings of a solution for SRAP, and the total number of ADMs for IDP. The objective function z_5 minimises the basic function z_0 and penalises the unfeasible solutions. In this objective function every constraint violated (partition with total load higher than *B*) has a certain cost (the current load of a partition minus *B*). Adding all violations of the current solution, the total violation is obtained. As z_0 is much smaller than the load of a ring, a feasible solution with 4 rings will be preferred to an unfeasible solution with 3 rings. In z_5 partitions are all the rings (in the case of the SRAP the federal ring is also included).

The evaluation process is the most time-consuming step, as it is the case in many real-life problems. Our algorithm has some important improvements. After creating/modifying a solution, the algorithm does not perform any full examination for computing the new fitness value; it only updates the fitness value based on the modifications made for creating the solution. The running time is considerably reduced.

3.4 Mating Flight

Initially the queen's speed is randomly generated between *0.1* and *1*, and the queen's energy is randomly generated between *0.5* and *1*.

$$speedQ = generateSpeed(0.1,1) \quad (9)$$

$$energyQ = generateEnergy(0.5,1) \quad (10)$$

In the beginning of each mating flight the queen selects a drone using the probabilistic rule in Eq. 5. If the mating is successful (i.e, the randomly generated number is less or equal than the probabilistic value), the drone's sperm is stored in the queen's spermatheca.

After each transition in the space, the queen's speed and energy decreases according to the following equations:

$$speedQ = speedQ * alfa \quad (11)$$

$$energyQ = energyQ - energyReduce \quad (12)$$

The mating flight ends when the queen's energy reaches the value zero or when the spermatheca is full.

3.5 Creation of broods and Workers Phase

For a required number of broods (*numBroods*) the queen is mated with a randomly selected sperm from her spermatheca. By using a crossover operator, the queen's and the drone's genotypes are recombined and a new brood is created. The new brood can be later improved by employing workers to drive the local search.

In this paper we have studied 5 crossover operators: one point, 2-points, 4-points, uniform and "exchange ring". One point, 2-points, 4-points and uniform are very well-known and widely used in practice. In "exchange ring", one ring is randomly selected and the costumers/edges of that ring are exchanged.

As explained before, the algorithm uses two workers, which are always used in an effort to improve the broods. The first worker uses a neighbourhood search algorithm. A neighbour is obtained by performing multiple moves whose length is specified as *nm*. We create two different procedures for SRAP and IDP. In SRAP we first try to exchange the ring of a random customer, to the ring with the minimum amount of traffic. If the solution does not improve we exchange the rings of two customers randomly selected. If the fitness of the solution still does not improve, we exchange the ring of a customer assigned to the ring with the maximum amount of traffic, to the ring with the lowest amount of traffic.

In IDP, we split the neighbourhood into two sets (70% and 30%). In 70% of the cases, it changes the solutions based on the modifications of two edges of different partitions or based on the modifications of the edges of two partitions randomly selected. In 30% of the cases, it changes one edge randomly selected to the partition with the lowest traffic or it changes the edges of a random partition.

The above procedures are repeated until at least one exchange with improvement is made or until *nm* is reached.

The second worker uses a LS algorithm. The LS algorithm applies a partial neighbourhood search. In our implementation, we create several LS methods. For SRAP and IDP we create the methods "Exchange one customer" and "Exchange two customers".

In "Exchange one customer", we generate a neighbour by assigning one customer (randomly chosen) to another ring (randomly chosen). The size of the neighbourhood used was *sizeSolution/10*. The algorithm searches for a better solution in the initial set of neighbours. If the best neighbour improves the actual solution, then the LS algorithm replaces the actual solution with the best neighbour.

In "Exchange two customers", we generate a neighbour by swapping two customers between two rings, *r1* and *r2* (randomly chosen). The algorithm searches for a better solution in the initial set of neighbours. If the best neighbour improves the actual solution, then the LS algorithm replaces the actual solution with the best neighbour. Otherwise, the algorithm creates another set of neighbours. In this case, one neighbour results on assigning one customer of *r1* to *r2*, or *r2* to *r1*. The neighbourhood size is *n(r1)*n(r2)*, or *n(r1)*n(r2) + n(r1)+n(r2)*, where *n* is the number of customers assigned to that ring.

For IDP we also implement the LS methods: "Exchange Max Partition" and "Exchange Min Partition". In "Exchange Max Partition", a customer of the partition with the maximum amount of traffic is randomly selected and assigned to other ring. In "Exchange Min Partition", a customer not assigned to the partition with the lowest amount of traffic is randomly selected and assigned to that partition. The size of the neighbourhood used for the last two methods was *sizeSolutio /20*. If the best neighbour improves the actual solution, then the LS algorithm replaces the actual solution with the best neighbour.

3.6 Best brood selection

The algorithm selects the best brood. If this brood is better than the current queen it replaces the queen. The best drones that do not replace the queen will be selected to the next matting flight as drones (see subsection 3.7).

3.7 Drones Replacement

The algorithm selects the best broods to be drones in the next mating flight. If the number of drones is higher than the number of broods, the remaining drones are generated using the deterministic algorithm (see subsection 3.2).

3.8 Termination Criterion

The algorithm stops when a predefined number of mating flights (*numMatingFlights* – number of iterations) is reached.

4. BENCHMARK INSTANCES

In order to test the performance of our approach, we use a collection of instances of different sizes.

For SRAP we used four sets of instances. The first one, class C1, has been introduced in [10]. Class C2 is composed of all instances tested in [10], excluding those already in C1, introduced in [3]. The third set of instances, class C3, has been presented in [12].

The nomenclature of the classes C1, C2 and C3 is described in Table 1.

Table 1. Nomenclature and Capacity of the instances of classes C1, C2 and C3.

Class	Group	Capacity (*B*)	Total instances
C1	GL	155 *Mbs*	23 + 17*
	GH	622 *Mbs*	27 + 13*
	RL	155 *Mbs*	31 + 9*
	RH	622 *Mbs*	37 + 3*
	Total instances of Class C1:		118 + 42*= 160
C2	new.GL	155 *Mbs*	70
	new.GH	622 *Mbs*	70
	new.RL	155 *Mbs*	70
	new.RH	622 *Mbs*	20
	Total instances of Class C2:		230
C3	LSHK	48 *Mbs*	2 + 38* =40
	Total instances of Class C3:		40
GL (Geometric low)		GH (Geometric High)	
RL (Random Low)		RH (Random High)	
* Unfeasible instances for SRAP			

Bastos [4] generated the instances of class C4. Two algorithms to generate higher instances have been used. The nomenclature of these instances is not related with the ability of the rings given by *B*, but by the number of customers in the instance. Thus, there are solutions in class C4 named by prefixes: RTNR_100, RTNR_150, RTNR_200, RTNR_250, RTNR_300, and RTNR_400 followed by the instance number. For example, instance RTNR_300_06 has 300 customers and is the sixth instance of this group. The values of *B* used to generate these levels ranged between *51.84 Mbs* (STS-1) and *39813.12 Mbs* (STS-768) (see Table 2).

Table 2. Nomenclature and Capacity of the instances of class C4.

Group	Instance number	Capacity (*B*)	Total instances
RTNR_100	[01 ... 10]	2488 *Mbs*	46
	[12 ... 16]	9953.28 *Mbs*	
	[11 and 17 ... 26]	51.84 *Mbs*	
	[27 ... 31]	155.52 *Mbs*	
	[32 ... 36]	622.08 *Mbs*	
	[37 ... 41]	2488.32 *Mbs*	
	[42 ... 46]	39813.32 *Mbs*	
RTNR_150	[01 ... 10]	2488 *Mbs*	10
RTNR_200	[01 ... 10]	9953 *Mbs*	31
	[11 ... 15	] 51.84 *Mbs*	
	[16 ... 21]	155.52 *Mbs*	
	[22 ... 26]	2488.32 *Mbs*	
	[27 ... 31]	39813.32 *Mbs*	
RTNR_250	[01 and 02]	622 *Mbs*	6
	[03, 05 and 06]	2488 *Mbs*	
	04	9953.28 *Mbs*	
RTNR_300	[01 ... 06]	622.08 *Mbs*	6
RTNR_400	[01 ... 06]	2488.32 *Mbs*	6
		Total:	105

Because they are larger and denser than the instances of C1, C2 and C3, the instances of the class C4 are also more difficult. For these instances, the values of optimal solutions are not yet known.

For IDP, we only studied the instances of C1 and C3 sets. We want to compare our results with others from literature. Pelleau et al. [15] provided their results for this two set of instances (they have specified the number of ADMs obtained for each instance). In [3], the authors also solved IDP using class C2, however they only present the number of optimal solutions. For our work, it is necessary to know the exact number of ADMs obtained for each instance in order to make an effective comparison.

As all instances can be feasible for IDP, it is possible to assign each demand to a different partition.

5. RESULTS

Aringhieri and Dell'Amico [3] described the results obtained for SRAP and IDP on the benchmark sets C1, C2 and C3 studied in this paper, by the algorithms BTS, PR1, PR2, XTS, SS, and DMN. Pelleau et al. [15] implemented all the algorithms described by Aringhieri and Dell'Amico, including a new one - DMN2 - and compared them (for the instances C1 and C3) using the new objective function z_5. DMN and DMN2 were the heuristics that obtained the best results. Bastos et al. [4, 5, 6, 7] solved SRAP using GA, GA-EvPR, GRASP, and GRASP-PR, and obtained good results (for the instances C1 and C2) with them. Silva et al. [16] developed the algorithm MA+VB to solve SRAP. They compared this algorithm with GA and MA, and MA+VB obtained better results for instances C1 and C2.

Recently, Bernardino et al. [9] solved the SRAP and IDP using the HSS. This algorithm has also achieved very good results for the instances C1, C2 and C3. In this paper we also apply this algorithm to class C4 using the best combination of parameters obtained in [9]. We only perform comparisons with class C4 for SRAP using the algorithms HSS and hybrid HBMO. In literature, we have found two works that have used this class [4, 14]. However, for some instances the final result was smaller than k_{lb} and in our work this is not possible. It seems that they are using different evaluation methods.

In this paper we compare our IDP results with DMN, DMN2 (algorithms that obtained the best results in [15]), and HSS, and our SRAP results with DMN, DMN2, GA, GA-EvPR, GRASP, GRASP-PR, MA-VB, and HSS.

The algorithms HSS and hybrid HBMO have been coded in C++ and tested on Intel based, dual-core T2300, running under Windows.

Bernardino el al. [8] have applied the Hybrid HBMO to TAP. TAP is a similar assignment problem where concentrators in TAP are rings in SRAP, and partitions in IDP, and terminals in TAP are customers in SRAP and IDP. In TAP, the solutions are also represented using integer vectors. The suggestions from literature, namely the TAP parameter setting, helped us to guide our choice of parameter values for SRAP and IDP.

The best results obtained with the hybrid HBMO algorithm use *alfa>=0.90*, *energyReduce<=0.1*, *numDrones* between *10* and

100, *numBroods>=100*, *sizeSper* between *10* and *30*, *cr*={one-point, uniform} and *nm<=10*. In general, experiments have shown that the proposed parameter setting is very robust to small modifications.

The initial population was created using the deterministic algorithm. Bernardino et al [9] verified that the deterministic algorithm obtained a slightly better average fitness comparing with a random initial population.

We also perform comparisons between the LS methods used for SRAP and IDP. The best LS method for SRAP is "Exchange two customers" and the best LS methods for IDP are "Exchange two customers" and "Exchange Min Partition".

The parameter values used to evaluate the efficiency of the algorithms are summarised in Table 3.

Table 3. Parameter values used for SRAP and IDP.

Algorithm	Parameter	Values
HSS	Number of initial solutions	*30..50*
	Number of best solutions in reference set	*8..10*
	Number of most different solutions in the reference set	*8..10*
	Number of iterations without improvement	*n/15..n/2*
Hybrid HBMO	Alfa	*[0.9…1]*
	Energy reduce	*[0.01…0.1]*
	Number of drones	*30*
	Number of broods	*100*
	Spermatheca size	*10*
	Crossover operator	*uniform*
	nm	*{2, 3, 4}*

For each SRAP C1 and C2 instance considered in our experiments, we fix a solution target value equal to the optimal solution. In [13], the authors implemented an exact algorithm that provided optimal solutions for the two classes C1 and C2 (in almost all instances $k_{lb}+1$). In class C3 there are only two feasible instances, and for them we consider the lower bound k_{lb}. In class C4 we also consider the lower bound k_{lb}. For IDP, we consider the lower bound k_{lb}.

We gave a time limit of seconds to each run of our algorithms (see Table 4). To solve an IDP instance, it normally takes much more iterations/time to improve the fitness value than in SRAP. In SRAP, the size of the solutions is smaller. Obviously, the algorithm terminates if the current best solution found is equal to the optimal solution – class C1 and C2. If the optimal solution is not reached, we define as a high-quality solution a solution for which the evaluation of the objective is equal to $k_{lb}+1$.

Table 4. Number of seconds used to test the instances.

Class	Group	HSS		HBMO	
		SRAP	IDP	SRAP	IDP
		Time Limit (in sec.)	Time Limit (in sec.)	Time Limit (in sec.)	Time Limit (in sec.)
C1	-	*	*	30	60
C2	-	*	-	40	-
C3	-	*	*	30	60
C4	RTNR_100	40	-	40	-
	RTNR_150	120	-	120	-
	RTNR_200	1000	-	1000	-
	RTNR_250	1000	-	1000	-
	RTNR_300	1500	-	1500	-
	RTNR_400	2000	-	2000	-
* We only perform new examinations for class C4. The remaining results for HSS were taken from literature [23].					

In Table 5, we compare the SRAP results obtained by hybrid HBMO with the results taken from [7, 9, 11, 15, 16]. The first two columns show the instance class and number of feasible solutions, the following columns give the percentage of optimal solutions found in literature by other algorithms. The presented hybrid HBMO values have been computed based on *100* different executions for each test instance, using the best combination of parameters found and different seeds.

The hybrid HBMO did not find the optimal solution for just one instance – *new.GH_50_3.4*. We do not present in Table 5 the results for class C4, since HSS and hybrid HBMO have obtained the same fitness values.

Table 5. Results for SRAP.

Class	**FS**	**GRASP**	**GRASP-PR**	**GA**	**GA-EvPR**	**EB. CB. NB**	**DMN**	**DMN2**	**MA+VB**	**HSS**	**HBMO**
C1	118	100 *	100*	98.3	100	97.46	100	97.46	100	100	100
C2	230	98.69	99.57	85.7	98.26	-	98.26	-	99.57	99.57	99.57
C3	2	-	-	-	-	-	100	100	-	100	100
* The authors only consider 111 feasible instances											

Table 6. Average time to reach the best solution.

Class	**Group**	**HSS**		**HBMO**	
		SRAP	**IDP**	**SRAP**	**IDP**
		Time Limit (in seconds)	Time Limit (in seconds)	Time Limit (in seconds)	Time Limit (in seconds)
C1	-	0.06	18.24	0.06	16.09
C2	-	1.87	-	1.62	-
C3	-	0.04	6.97	0.04	5.85
C4	RTNR_100	6.11	-	5.93	-
	RTNR_150	17.96	-	16.27	-
	RTNR_200	121.20	-	118.41	-
	RTNR_250	96.55	-	82.76	-
	RTNR_300	200.21	-	189.39	-
	RTNR_400	289.38	-	268.72	-

For IDP, we performed identical examinations. With hybrid HBMO, we have obtained identical number of ADMs for all the instances studied by Pelleau et al. [15] using DMN and DMN2, and by Bernardino et al. [9] using HSS. Pelleau et al. [15] indicate that for IDP, they have obtained better results for 15 instances - we have obtained the same values in a smaller execution time. In their work they used a limit of 5 minutes - we have obtained our results with less than 60 seconds. In 7 instances of the 200 instances studied for this problem, it was necessary $k_{lb}+1$ rings to obtain the smallest number of ADMs.

The results show that GRASP-PR, MA-VB, HSS and hybrid HBMO present better results in terms of solution quality for SRAP and DMN, HSS and hybrid HBMO obtain better results also in terms of solution quality for IDP. Comparing computational times (to achieve the best/optimal solution) among HSS and hybrid HBMO, hybrid HBMO is the fastest algorithm (see Table 6). MA-VB obtains identical SRAP results, however with higher times (around *15* seconds for C1 instances and *34* seconds for C2 instances). We do not perform comparisons in terms of computational time with other algorithms, because the experiments were performed in different equipment, but we believe that our algorithm is faster. Our average times are significantly smaller in comparison with the ones obtained in literature to the same instances. Even for C4 instances the results were produced very fast.

6. CONCLUSION

In this paper we present a Hybrid HBMO algorithm to solve the SRAP and IDP. Extensive computational experiments were done with several instances from literature. The performance of' HBMO algorithm is compared with several algorithms used to solve the same problems studied in this paper.

The computational results show that Hybrid HBMO has a stronger performance, improving in some cases, the results obtained by previous approaches. We cannot say that for IDP and SRAP the HBMO is the fastest algorithm, because we have implemented and executed it in other platform, however we can say that it is competitive with the best heuristics in literature in terms of solution quality. It is faster in comparison with HSS. We didn't find in literature all the necessary results to fully compare our results. It will be interesting to do it in the future. In literature, the application of the Hybrid HBMO algorithm for these problems is nonexistent. For that reason, this article shows its enforceability in the resolution of these problems.

7. REFERENCES

[1] Abbass, H. A. 2001. Marriage in honey-bee optimization (MBO): a haplometrosis polygynous swarming approach. In I*EEE Congress on Evolutionary Computation* (Seoul, Korea, May 27 - 30, 2001). IEEE Press. 207–214.

[2] Abbass, H. A. 2001. A monogenous MBO approach to satisfiabiality. In *Proceedings of the Int. Conference on Computational Intelligence for Modelling, Control and Automation* (Las Vegas, NV, USA, July 9-11, 2001).

[3] Aringhieri, R. and Dell'Amico, M. 2005. Comparing Metaheuristic Algorithms for Sonet Network Design Problems. *Journal of Heuristics.* 11, 1 (Jan. 2005), 35-57.

[4] Bastos, L. O., Ochi, L. S. and Macambira, E. M. 2005. A relative neighbourhood GRASP for the SONET ring assignment problem. In *Proceedings of the International Network Optimization Conference* (Lisboa, Portugal, March, 2005) 833-838.

[5] Bastos, L. O., Ochi, L. S. and Macambira, E. M. 2005. GRASP with Path-Relinking for the SONET Ring Assignment Problem. In *Proceedings of the Fifth International Conference on Hybrid Intelligent Systems - HIS'05* (Rio de Janeiro, Brazil, November 6-9, 2005). IEEE Computer Society. 239-244.

[6] Bastos, L. O. 2005. *Heuristic Solutions for the SONET Ring Assignment Problem*. Master Thesis. UFF, Niterói/RJ.

[7] Bastos, L. O., Ochi, L. S. 2008. A genetic algorithm with evolutionary path-relinking for the Sonet Ring Assignment Problem. In P*roceedings of the International Conference on Engineering Optimization - EngOpt2008* (Rio de Janeiro, Brazil, June 1-5, 2008).

[8] Bernardino, E. M., Bernardino, A. M., Sánchez-Pérez, J. M., Vega-Rodríguez, M. A. and Gómez-Pulido, J. A. 2010. Hybrid HBMO Algorithm to assign terminals to concentrators. In *Proceedings of the 3rd International Symposium on Applied Sciences on Biomedical and Communication Technologie - ISABEL 2010* (Rome, Italy, November 7-10, 2010). IEEE Press.

[9] Bernardino, A. M., Bernardino, E. M., Sánchez-Pérez, J. M., Vega-Rodríguez, M. A. and Gómez-Pulido, J. A. 2011. Solving SONET problems using a Hybrid Scatter Search Algorithm. In *Studies in Computational Intelligence*. Springer, Berlin / Heidelberg.

[10] Goldschmidt, O., Laugier, A. and Olinick, E. V. 2003. SONET/SDH Ring Assignment with Capacity Constraints. *Discrete Applied Mathematics*. 129, 1 (June. 2003), 99-128.

[11] Goldschmidt, O., Hochbaum, D. S., Levin, A. and Olinick, E. V. 20003. The Sonet Edge-Partition Problem. *Networks*. 41, 1 (Dec. 2003), 3-23.

[12] Lee, Y., Sherali, H. D., Han, J. and Kim, S. 2000. A Branch-and-Cut Algorithm for Solving an Intraring Synchronous Optical Network Design Problem. *Networks*. 35, 3 (Apr. 2000), 223-232.

[13] Macambira, E. M., Maculan, N. and Souza, C. C. 2006. A column generation approach for SONET ring assignment. *Networks*. 47, 3 (Feb. 2006), 157-171.

[14] Oliveira, W. 2010. *Parallel Evolutionary Algorithm to the SONET/SDH Ring Assignment Problem*. Master Thesis. UFRN, Natal/RN.

[15] Pelleau, M., Van Hentenryck, P. and Truchet, C. 2009. Sonet Network Design Problems. In *Proceedings of the Sixth International Workshop on Local Search Techniques in Constraint Satisfaction* (Lisbon, Portugal, September 20, 2009). EPTCS 5, 81-95.

[16] Silva, A. C. G., Marinho, E. S., Oliveira, W. and Aloise, D.J. 2009. Genetic Algorithm and Memetic Algorithm with Vocabulary Building for the SONET Ring Assignment Problem. In *Proceedings of the EU/MEeting 2009 - European Chapter on Metaheuristics* (Porto, Portugal, April 29-30, 2009). 69-74.