Journal of Network and Computer Applications 34 (2011) 668–685

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Solving ring loading problems using bio-inspired algorithms

Anabela Moreira Bernardino [a,*], Eugénia Moreira Bernardino [a], Juan Manuel Sanchez-Perez [b], Juan Antonio Gomez-Pulido [b], Miguel Angel Vega-Rodriguez [b]

[a] Department of Computer Science—Research Center for Informatics and Communications, School of Technology and Management, Polytechnic Institute of Leiria, 2400 Leiria, Portugal
[b] Department of Technologies of Computers and Communications, Polytechnic School, University of Extremadura, 10071 Cáceres, Spain

ARTICLE INFO

Article history:
Received 22 May 2010
Received in revised form
4 November 2010
Accepted 13 November 2010
Available online 23 November 2010

Keywords:
Communication networks
Bio-inspired algorithms
Optimisation algorithms
Swarm Intelligence
Weighted Ring Arc-Loading Problem
Weighted Ring Edge-Loading Problem

ABSTRACT

In the last years, several combinatorial optimisation problems have arisen in the communication networks field. In many cases, to solve these problems it is necessary the use of emergent optimisation algorithms. The Weighted Ring Loading Problem (WRLP) is an important optimisation problem in the communication optical network field. When managed properly, the ring networks are uniquely suited to deliver a large amount of bandwidth in a reliable and inexpensive way. An optimal load balancing is very important, as it increases the system's capacity and improves the overall ring performance. The WRLP consists on the design, in a communication network of a transmission route (direct path) for each request, such that high load on the arcs/edges is avoided, where an arc is an edge endowed with a direction. In this paper we study this problem in two different ring types: Synchronous Optical NETworking (SONET) rings and Resilient Packet Ring (RPR). In RPR the purpose is to minimise the maximum load on the ring Arcs (WRALP). In SONET rings the purpose is to minimise the maximum load on the ring Edges (WRELP). The load of an arc is defined as the total weight of those requests that are routed through the arc in its direction and the load of an edge is the total weight of the routes traversing the edge in either direction. In this paper we study both problems without demand splitting and we propose three bio-inspired algorithms: Genetic Algorithm with multiple operators, Hybrid Differential Evolution with a multiple strategy and Hybrid Discrete Particle Swarm Optimisation. We also perform comparisons with other algorithms from literature. Simulation results verify the effectiveness of the proposed algorithms.

1. Introduction

Nowadays Synchronous Optical NETworking (SONET) rings are widely used in telecommunications. In a SONET ring, nodes (typically telephone central offices) are connected among them by a ring of fiber. Nodes send, receive and relay messages by means of a device called an Add-Drop-Multiplexer (ADM) that determines the actual bandwidth available along any edge of the SONET ring (Goralski, 2002). An important optimisation problem arising in this context is the Weighted Ring Edge-Loading Problem (WRELP). Given a network and a set of communication requests, a fundamental problem is to design a transmission route (direct path) for each request such that the high load on the edges will be avoided. The load of an edge is the number of routes traversing the edge in either direction. In general each request is associated with a non-negative integer weight, and the load of an edge is defined as the total weight of those requests that are routed through the edge in both directions. The weight of a request can be interpreted as a traffic demand or the size of the data to be transmitted (Bernardino et al., 2008, 2009a; Cosares and Saniee, 1994; Dell'Amico et al., 1999; Goldschmidt et al., 2003; Karunanithi and Carpenter, 1994; Lee and Chang, 1997; Myung et al., 1997; Myung and Kim, 2004; Schrijver et al., 1998; van Hoesel, 2005; Wang, 2005).

Resilient Packet Ring (RPR), also known as IEEE 802.17, is a standard, designed to optimise the transport of data traffic through optical fiber ring networks (Davik et al., 2004; RPR Alliance, 2004; Yuan et al., 2004). The RPR aims to combine the appealing functionalities of Synchronous Optical Network/Synchronous Digital Hierarchy (SONET/SDH) networks with the advantages of Ethernet networks.

The load balancing model for RPR differs from the SONET ring loading in a number of significant aspects. Namely, SONET demands are bidirectional and demands assigned to go clockwise compete for common span capacity with the demands assigned to go counter-clockwise. In RPR two distinct rings occur (clockwise and counter-clockwise) and the demands do not compete for the common capacity. In SONET the demands are "circuit-switched", and deterministic, while the RPR is based on "packet" stream technology with Multi Protocol Label Switching (MPLS) like "tunnels", statistical multiplexing and different level of service.

* Corresponding author.
E-mail addresses: anabela.bernardino@ipleiria.pt (A.M. Bernardino), eugenia.bernardino@ipleiria.pt (E.M. Bernardino), sanperez@unex.es (J.M. Sanchez-Perez), jangomez@unex.es (J.A. Gomez-Pulido), mavega@unex.es (M.A. Vega-Rodriguez).

1084-8045/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2010.11.003

In this paper we also consider the Weighted Ring Arc-Loading Problem (WRALP) which arises in engineering and planning of the RPR systems. Specifically, for a given set of non-splitable and unidirectional commodities (point-to-point demands), the purpose is to find the routing for each commodity (i.e., assignment of the commodity to either clockwise or counter-clockwise ring) so that the maximum link segment load is minimised (Bernardino et al., 2009b; Cho et al., 2005; Kim et al.; 2008; Kubat and Smith, 2005; Yuan and Zhou, 2004).

There are three variants to solve these problems: (i) demands can be split into two parts, and then each one is sent in a different direction; (ii) demands are allowed to be split into two parts, but restricted to be integrally split; (iii) each demand must be entirely routed in either one of the two directions—clockwise or counter-clockwise. In this paper we study the third variant for the two loading problems, which NP-hardness can be drawn from the results in literature (Cosares and Saniee, 1994; Kubat and Smith, 2005). In this paper we do not present solutions for the split variants because they can be solved in polynomial time.

The WRLP is very difficult to solve with the use of exact algorithms, especially if the number of nodes and number of positive demands between pairs are very large due to the fact that this problem belongs to the category of NP-hard problems, i.e. no polynomial time algorithms are known for their solution. However, a number of exact algorithms have been proposed to solve this problem with a small number of nodes and demands. As it is a NP-hard problem, the instances with a large number of nodes and positive demand values between pairs cannot be optimally solved within a resonable time. Thus, many heuristic algorithms have been developed in order to find a near optimal solution in a resonable computational time.

For research on the non-split WRELP, Cosares and Saniee (1994) and Dell'Amico et al. (1999) studied the problem on SONET rings. Cosares and Saniee (1994) proved that the formulation without demand splitting is NP-complete. This means that we cannot guarantee to find the best solution in a reasonable amount of time. For the split problem, various approaches are summarised by Schrijver et al. (1998) and their algorithms are compared in Myung and Kim (2004) and Wang (2005).

The non-split WRALP considered in the present paper is identical to the one described by Kubat and Smith (2005) (non-split WRALP), Cho et al. (2005) (non-split WRALP and WRALP) and Yuan and Zhou (2004) (WRALP).

We verify that the main purpose of the previous works was to produce near optimal solutions for WRELP/WRALP in a reduced amount of time. Our purpose is different—we want to compare the performance of our algorithms with others in the achievement of the best-known solution. Using the same principle, Bernardino et al. (2008, 2009a, b, 2010a, b, c) presented several Evolutionary Algorithms (EAs) and a Tabu Search (TS) algorithm to solve the non-split WRELP and several EAs and Swarm Optimisation algorithms to solve the non-split WRALP. Karunanithi and Carpenter (1994) used a GA using the same purpose and tried to solve smaller WRELP instances. Recently, Kim et al. (2008) presented an Ant Colony Optimisation (ACO) algorithm using different strategies to solve WRELP.

Nonlinearities and complex interactions among design and operation variables in engineering problems form a search space with multiple optimal solutions, in which most local optima have inferior objective function values. Thus, gradient-based methods may not be good candidates for efficient optimisation algorithms when applied to a broad range of engineering design and operation problems. Over the last decades, nature inspired algorithms have been extensively used as search and optimisation tools in various problem domains. The broad applicability, ease of use, and global perspective of this bio-inspired algorithms may be considered as the primary reason for their extensive applications and success as search and optimisation tools in various problems domains. Bio-inspired randomised search heuristics such as EAs, hybridisations with Local Search (LS), and Swarm Intelligence (SI) are very popular among practitioners as they can be applied in the case of problems for which the formulation of explicit fitness function is difficult. EAs simulate the natural evolution of species by iteratively applying evolutionary operators such as mutation, recombination, and selection to a set of solutions for a given problem (Eiben and Smith, 2003). SI comprises ACO, Artificial Bee Colony (ABC) as well as Particle Swarm Optimisation (PSO) (Kennedy et al., 2001). These modern search paradigms rely on the collective intelligence of many single agents to find good solutions. A recent trend is to hybridise bio-inspired heuristics with LS to refine newly constructed solutions (Memetic Algorithms Website).

This paper is a continuation of previous papers, which considered the use of Hybrid Differential Evolution (HDE) algorithm (Bernardino et al., 2009a) and GA (Bernardino et al., 2008) to solve the non-split WRELP. Currently, we consider an improved HDE algorithm with a multiple strategy, including the incorporation of a different LS method to solve the WRELP and the WRALP. In this paper, the GA with multiple operators is considered to solve the WRALP. We also apply it to higher WRELP instances and the operators were adapted to use the binary representation. Bernardino et al. (2009b) had proposed several variants of PSO to solve the WRALP. In this paper we present a Hybrid Discrete PSO (HDPSO) algorithm, which uses a different algorithm model and is coupled with a different LS method. We compare the performance of our algorithms with TS, LS-Probability Binary PSO (LS-PBPSO), HDE, ABC, Hybrid ACO (HACO) and Discrete DE (DDE) used in literature to solve the same problems.

The paper is structured as follows. In Section 2 we describe the WRLP problem; in Section 3 we describe the implemented algorithms; 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.

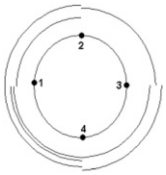
2. Weighted ring loading problems

In this paper we consider two loading problems. For both problems we consider a ring with:

- n nodes
  Let R_n be a n-node bidirectional ring with nodes $\{n_1, n_2, \ldots, n_n\}$ labelled clockwise. Each edge $\{e_k,e_{k+1}\}$ of R_n, $1 \le k \le n$, is taken as two arcs with opposite directions, in which the data streams can be transmitted in either direction (Schrijver et al., 1998): $a_k^+ = (e_k,e_{k+1})$, $a_k^- = (e_{k+1},e_k)$.
  A communication request on R_n is an ordered pair (s,d) of distinct nodes, where s is the source and d is the destination. We assume that data can be transmitted clockwise or counter-clockwise on the ring, without splitting. We use $P^+(s,d)$ to denote the directed (s,d) path clockwise around R_n, and $P^-(s,d)$ the directed (s,d) path counter-clockwise around R_n.
- Unidirectional weights
  Often a request (s,d) is associated with an integer weight $w >= 0$; we denote this weighted request by $(s,d; w)$. Let $Z=\{(s_1,d_1; w_1),(s_2,d_2; w_2),\ldots,(s_m,d_m; w_m)\}$ be a set of integrally weighted requests on R_n. For each request/pair (s_i,d_i) we need to design a directed path P_i of R_n from s_i to d_i. A collection $P=\{P_i:i=1, 2, \ldots, m\}$ of such directed paths is called a routing for Z.
- Weights cannot be split
  Each demand must be entirely routed in either one of the two directions, clockwise or counter-clockwise.

Table 1
Solution representation.

Pair(s, d) demand						
1: (1, 2) → 15	15 C					
2: (1, 3) → 3	3 CC					
3: (1, 4) → 6	6 CC					
4: (2, 3) → 15	15 C					
5: (2, 4) → 6	6 CC					
6: (3, 4) → 14	14 C					
n=number nodes=4	C—clockwise					
m=number pairs=6	CC—counter-clockwise					
Representation (L)	$Pair_1$ 1	$Pair_2$ 0	$Pair_3$ 0	$Pair_4$ 1	$Pair_5$ 0	$Pair_6$ 1



In this work, the solutions are represented using binary vectors (Table 1). If a position has the value 1, the demand flows by the clockwise direction, 0 otherwise.

The decision (assignment) variables are defined as

$$L_i = \begin{cases} 1 \text{ if demand flows by the clockwise direction} \\ 0 \text{ otherwise} \end{cases}.$$

We assume that weights cannot be split, i.e., for some integer $L_i=1$, $1 \le i \le m$, the total amount of data is transmitted along $P^+(s_i,d_i)$; $L_i=0$, the total amount of data is transmitted along $P^-(s_i,d_i)$. The vector $L=(L_1, L_2, \ldots, L_m)$ determines a routing scheme for Z.

2.1. WRELP formulation

WRELP is formulated as follows

$$W_i,\ldots,w_m \rightarrow \text{demands of the pairs } (s_i,d_i),\ldots,(s_m,d_m) \tag{1a}$$

$$L_i,\ldots,L_m = 0 \rightarrow P^-(s_i,d_i);\ 1 \rightarrow P^+(s_i,d_i) \tag{1b}$$

Load on arcs:

$$\text{Load}(L,a_k^+) = \sum_{i:a_k^+ \in P^+(s_i,d_i)} w_i \quad \text{Load}(L,a_k^-) = \sum_{i:a_k^- \in P^-(s_i,d_i)} w_i \tag{2a}$$

Load on edges:

$$\text{Load}(L,e_k) = \text{Load}(L,a_k^+) + \text{Load}(L,a_k^-) \quad \forall k = 1,\ldots,n; \quad \forall i = 1,\ldots,m \tag{2b}$$

Fitness function:

$$\max\{\text{Load}(L,e_k)\} \tag{3}$$

For a given ring, between each node pair (s_i,t_i) there is a demand value $>=0$. Constraint sets (1) state that each positive demand value is routed in either clockwise (C) or counter-clockwise (CC) direction.

For an edge, the load is the sum of w_i for clockwise and counter-clockwise between nodes e_k and e_{k+1} (2a and 2b). The purpose is to minimise the maximum load on the edges of a ring (3).

2.2. WRALP formulation

Likewise, for the WRALP, constraint sets (1) state that each demand is routed in either clockwise (C) or counter-clockwise (CC) direction.

For an arc, the load is the sum of w_i for clockwise or counter-clockwise between nodes e_k and e_{k+1} (2a). The purpose is to minimise the maximum load on the arcs of a ring (4).

WRALP fitness function:

$$\max\{\max \text{Load}(L,a_k^+), \max \text{Load}(L,a_k^-)\} \tag{4}$$

3. Bio-inspired algorithms

During the last years, nature inspired intelligence became increasingly popular through the development and use of intelligent methods in advanced information systems' design and optimisation. These methods are driven by concepts from nature and biology. *To solve the non-split WRELP and the non-split WRALP, we implement three well-known bio-inspired algorithms: GA, HDEM and HDPSO.* Relevant contributions include the design of new genetic operators for the GA, the application of a new strategy for the DE and the application of hybrid algorithms.

3.1. Genetic algorithm

GAs are EAs inspired by the natural process of reproduction (Holland, 1975). Metaphors as chromosomes and population stand for solutions and solution set, respectively. Analogously, a single variable is often indicated as a gene. Mechanisms as recombination and mutation give rise to new offspring by manipulating the current population of solutions. Specially, mutation applies to a single solution (chromosome) while crossover creates new solutions from a pair of solutions selected in the current population. Following a standard Darwinian approach, selection extracts the most promising individuals from the current population (Goldberg, 1989).

The main steps of the GA are:

```
Initialise Parameters
Initialise Solutions (suitable solutions for the
 problem)
Evaluate Solutions
While number of generations <=MAXG:
  Create a new population by repeating the following
  steps:
    Selection→Select two parent chromosomes from a
  population according to their fitness (the better
  fitness, the bigger chance to be selected)
    Crossover→With a crossover probability,
  crossover the parents to form a new offspring. If no
  crossover was performed, offspring is an exact copy
  of the parents
    Mutation→With a mutation probability, mutate
  new offspring
    Place new offspring in a new population
  Evaluate solutions in the population
Output the best solution
```

3.1.1. Initialisation of parameters

The following parameters must be defined by the user: population size (NP), selection operator, crossover operator, mutation operator, crossover probability, mutation probability and maximum number of generations (MAXG).

3.1.2. Initialisation of solutions

The initial solutions can be created randomly or in a deterministic form. The deterministic form is based in a Shortest-Path Algorithm (SPA). The SPA is a simple traffic demand assignment rule in which the demand will traverse the smallest number of segments.

3.1.3. Evaluation of solutions

The fitness function is responsible for performing the evaluation and returning a positive number (fitness value) that reflects how good the solution is. We use the fitness function (3) for the WRELP and fitness function (4) for the WRALP.

3.1.4. Selection

Some individuals are selected from the population to be parents to crossover. Parents are selected according to their fitness. We implement three well-known selection methods (Eiben and Smith, 2003): "Roulette", "Tournament" and "Tournament with Elitism" (20% of the best individuals of the population are maintained in the following generation; the others are selected with the "Tournament" method).

3.1.5. Crossover

We implement seven crossover operators: "One point", "2-points", "4-points" and "Uniform" are well-known and widely used in practice. The crossover operator "Exchange positions" is based on the "2-points" operator. However in this case, 2-points are obtained for each parent (Fig. 1). In the "Differential" crossover operator, if the genes in the same position of the two parents have different values, their values are exchanged, otherwise both values are changed to the opposite direction (Fig. 2).

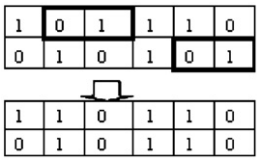


Fig. 1. Exchange positions.

In previous studies, we verified that the success of applying a crossover operator is highly dependent of the problem studied and the instance used. To dynamically apply the best crossover operator to solve an instance, we create a new crossover operator—"Multiple". In the "Multiple" crossover operator, a set of available crossover operators is initially established. The crossover set contains the six crossover operators applicable to the problem. In the initialisation phase, each crossover operator has the same probability of being selected. From thereon and after each recombination, a fitness value is assigned to the respective crossover operator based on its contribution to individuals' fitness. This operator fitness value is used for recombination selection. The recombination operators are selected using a tournament selection. The program chooses three random operators. The operator with the highest fitness will win. After a predefined number of recombinations (NUM_PREV_OPS_RECOMB), the probabilities of each crossover operator are updated, based on their contribution in the last recombinations.

Multiple Recombination:

```
FitIni1 = fitness(individual1)
FitIni2 = fitness(individual2)
IF(first time)
  initialise numberOperationsR
  initialise fitPrevGensR
  initialise fitnessActualR
  num_operations_totalR = 0
ELSE
  IF (num_operations_totalR = NUM_PREV_OPS_RECOMB)
    num_operations_totalR = 0
    FOR i=1 TO NUM_OPERATORS DO
      fitnessActualR[i] = fitPrevGensR [i] / numberOperationsR[i]
    initialise numberOperationsR
    initialise fitPrevGensR
num_operations_totalR = num_operations_totalR + 1
operator = random(NUM_RECOMB_OPERATORS)
FOR op = 1 TO 3 DO
  op = chooseRandomOperator()
  IF (fitnessActualR[op] > fitnessActualR[operator])
     operator = op
SWITCH(operator)
  CASE 1: recomb = Recombination1Point
  CASE 2: recomb = Recombination2Points
  CASE 3: recomb = Recombination4Points
  CASE 4: recomb = RecombinationUniform
  CASE 5: recomb = RecombinationExchangePositions
  CASE 6: recomb = RecombinationDifferential
run(recomb, individual1, individual2)
fitPrevGensR [operator]= fitPrevGensR [operator] +
                  max(fitIni1,fitIni2) - max(fitness(individual1),fitness (individual2))
numberOperationsR[operator] = numberOperationsR[operator] + 1
```

Vector `numberOperationsR`– number of times where each operator is applied in previous generations.
Vector `fitPrevGensR`– total fitness achieved by each strategy in previous generations.
Vector `finessActualR`– fitness achieved by each operator using the `numberOperationsR`.

3.1.6. Mutation

We implement four mutation operators: "Change direction", "Change order", "Change direction-order" and "Multiple".

In "Change direction" one gene is randomly selected and its direction is exchanged. In "Change order" two genes are randomly selected and their directions are exchanged. In "Change direction-order", in 70% of the cases "Change direction" is applied in the remaining "Change order".

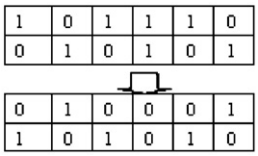


Fig. 2. Differential.

We also apply a "Multiple" mutation operator to automatically select the best mutation operator. A set of available mutation operators is initially established. The mutation set contains the three mutation operators. In the initialisation phase, each mutation operator has the same probability of being selected. From thereon and after each mutation a fitness value is assigned to the respective mutation operator, based on its contribution to individual fitness. This operator fitness value is used for mutation selection. The mutation operators are selected using a tournament selection. The program chooses two random operators. The operator with the highest fitness wins. After a predefined number of mutations (NUM_PREV_OPS_MUT) the probabilities of each mutation operator are updated based on their contribution.

Multiple Mutation:

```
fitIni = fitness(individual)
IF(first time)
  initialise numberOperationsM
  initialise fitPrevGensM
  initialise fitnessActualM
  num_operations_totalM = 0
ELSE
  IF(num_operations_totalM = NUM_PREV_OPS_MUT
    num_operations_totalM = 0
    FOR i=1 TO NUM_OPERATORS DO
      fitnessActualM[i] = fitPrevGensM [i] / numberOperationsM[i]
    initialise numberOperationsM
    initialise fitPrevGensM
num_operations_totalM = num_operations_totalM + 1
operator = random(NUM_MUT_OPERATORS)
FOR op = 1 TO 2 DO
  op = chooseRandomOperator()
  IF(fitnessActualM[op] > fitnessActualM[operator])
     operator = op
SWITCH(operator)
  CASE 1: mut = MutationChangeDirection
  CASE 2: mut = MutationChangeOrder
  CASE 3: mut = MutationChangeDirectionOrder
run(mut, individual)
fitPrevGensM [operator]= fitPrevGensM [operator] + fitIni - fitness(individual)
numberOperationsM[operator] = numberOperationsM[operator] + 1
```

Vector `numOperationsM`– number of times where each mutation operator is applied in previous generations.
Vector `fitPrevGensM`– total fitness achieved by each mutation operator in previous generations.
Vector `fitnessActualM`– fitness achieved by each mutation operator considering the `numOperationsM`.

We have used crossover/mutation operators in the multiple operators that can be applied to all kind of binary problems, since the best crossover/mutation operators are highly dependent of the problem or instance to be solved. The purpose is to extract the best operators adapted to the instance resolution.

3.1.7. Termination criterion

The algorithm stops when a maximum number of generations (MAXG) is reached.

This algorithm was first applied to the WRELP by Bernardino et al. (2008) using an integer representation. In this paper we apply it to larger instances and also to the WRALP. All the GA operators used in this paper have been recreated to be applied to the binary representation.

We have tried to apply the LS method used in HDPSO and in HDEM to GA, but that led to poor performance, mainly due to the size of the GA population.

3.2. Hybrid differential evolution algorithm

The DE algorithm was introduced by Storn and Price in 1995. It is a method of mathematical optimisation of multidimensional functions and belongs to the class of EAs. DE explores the candidate solutions encoded in chromosomes and iteratively exploits those with better fitness until it reaches the stop condition (Storn and Price, 1995, 1997).

The HDE algorithm combines global and local search using an EA to perform explorations while the LS method performs the exploitation. Combining global and local search is a strategy used by many successful global optimisation approaches, and these types of algorithms have in fact been recognised as powerful algorithmic paradigms for evolutionary computing (Memetic Algorithms Website). This method has proved to be of practical success in several problem domains. This algor^{ith}m is also known as Memetic Algorithm, Hybrid EA, etc. (Moscato, 1989). This algorithm was first applied to the WRELP by Bernardino et al. (2009a). In this paper we apply a different LS method and we propose a new strategy to solve WRELP and WRALP.

DE is a population-based algorithm like GAs, using similar operators: crossover, mutation and selection. It resembles the structure of a GA, but differs in the generation of new candidate solutions and by the use of a 'greedy' selection scheme. The main difference in building solutions is that our implementation of the GA relies on the crossover operation, while DE relies on the mutation operation. The algorithm uses the mutation operation as a search mechanism and the selection operation to direct the search toward the prospective regions in the search space.

The DE algorithm also uses a non-uniform crossover. The crucial idea behind DE is a scheme to generate trial parameter vectors. Using the components of the existing population members to build trial vectors, the recombination operator effectively shuffles information

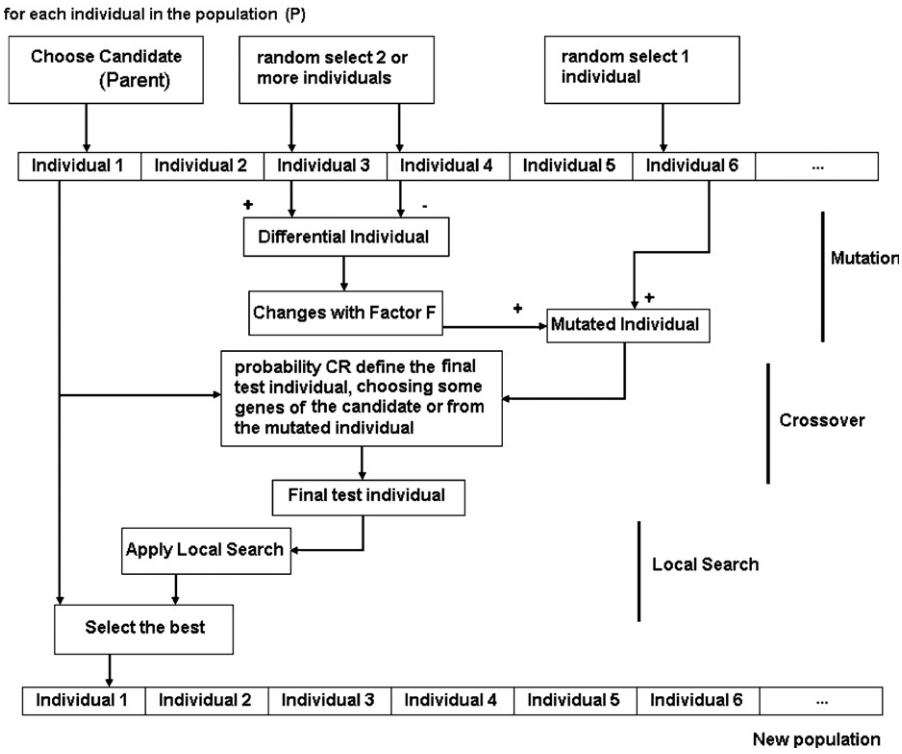


Fig. 3. HDEM algorithm—creating a new solution.

Table 2
DE strategies.

Name	Mutation
Rand1bin	P_i=ind1+F^* (ind2–ind3)
Best1bin	P_i=best+F^* (ind1–ind2)
Rand2bin	P_i=ind1+F^* (ind2+ind3–ind4–ind5)
Best2bin	P_i=best+F^* (ind1+ind2–ind3–ind4)
RandBest2Bin	P_i=old+F^* ((best–old)+(ind1–ind2))
Rand1exp	P_i=ind1+F^* (ind2–ind3)
Best1exp	P_i=best+F^* (ind1–ind2)
Rand2exp	P_i=ind1+F^* (ind2+ind3–ind4–ind5)
Best2exp	P_i=best+F^* (ind1+ind2–ind3–ind4)
RandBest2Exp	P_i=old+F^* ((best–old)+(ind1–ind2))
Current2Rand	P_i=old+CR* (ind1–old)+F^* (ind2–ind3)
Best3Bin	P_i=best+F^* (ind1+ind2+ind3–ind4–ind5–ind6)
RandRandBin	P_i=ind1+z^* (ind2–ind3); z is $N(0,1)$ Gaussian variable

about successful combinations, enabling the search for a better solution space (Fig. 3). There are several strategies with different approaches that can be used to solve the loading problems (Table 2).

The proposed HDEM algorithm can be summarised in the following pseudo-code steps:

```
Initialise parameters
Initialise solutions (suitable solutions for the
  problem)
While number of generations <=MAXG:
   Perform mutation and crossover operations using a
   defined strategy (Table 2) as explained in the DE
   algorithm (Price and Storn, 2005) on all the
   population members:
      Selection→For each parent, select three (or
      more) distinct vectors randomly from the current
      population. The selected vectors must not be the
      same as the parent vector. These vectors combine to
      produce an offspring
      Mutation→Calculate new mutation vector using
      the formula of the corresponding strategy selected
      initially
      Crossover→Perform crossover using one of the two
      crossover methods: Bin—Binomial crossover or
      Exp—Exponential crossover
      Apply LS method
      Evaluate new offspring and check if it is better or
      equal to the parent. Replace the parent with
      offspring in the next generation if the offspring is
      better or equal to the parent, otherwise, the parent
      proceed to the next generation
Output the best solution
```

3.2.1. Initialisation of parameters

The following parameters must be defined by the user: population size (NP), crossover constant (CR), scaling factor (F), maximum number of generations (MAXG) and strategy.

3.2.2. Initialisation of solutions

The initial solutions can be created randomly or in a deterministic form using the SPA.

3.2.3. Evaluation of solutions

To evaluate the solutions we use the fitness function (3) for the WRELP and fitness function (4) for the WRALP.

3.2.4. Selection

Parameter vectors ind1, ind2, ind3, … (usually three, depending on the strategy chosen) are selected from the population. Note that all vectors in this step are distinct from each other.

3.2.5. Mutation

Basically, DE adds the weighted difference between two population vectors to a third vector to create a mutated vector P_i (Table 2, Fig. 3). F is the mutation factor and is selected between 0 and 2.

3.2.6. Crossover

At the recombination step, new individuals are created by combining the mutated vector with the old individual vector (parent). The combination takes place according to the applied strategy (Price et al., 2005).

After recombination, if a gene (pair) of the final test individual (Fig. 3) has a value outside of the allowed range [0,1], it is necessary to apply the following transformation:

```
IF currentValuegene > 1 currentValuegene=1
ELSE IF currentValuegene < 0 currentValuegene=0
```

3.2.7. Local search

HDEM uses a DE algorithm to explore several regions of the search space and it simultaneously incorporates a mechanism (LS algorithm) to enhance the search around some selected regions (Fig. 3). The LS algorithm by itself explores the solution space making specific moves in its neighbourhood. The LS algorithm only applies a partial neighbourhood examination. Some pairs of the solution are selected and their directions are exchanged (partial search). The LS method only accepts neighbourhoods that improve the actual solution (deterministic).

This method can be summarised in the following pseudo-code steps:

```
For t=0 to numberNodesRing/4
  p1=random (number of pairs)
  p2=random (number of pairs)
  N=neighbourhoods of ACTUAL-SOLUTION (one
  neighbourhood results of interchange the direction
  of p1 and/or p2)
  SOLUTION=FindBest (N)
  If ACTUAL-SOLUTION is worst than SOLUTION
    ACTUAL-SOLUTION=SOLUTION
```

The performance of the child vector and its parent is compared and the best one is selected, as shown in Fig. 3. If the parent is still better, it is maintained in the population.

3.2.8. Termination criterion

The algorithm stops when a maximum number of generations (MAXG) is reached.

Bernardino et al. (2009a) proved that the best strategies to the formulation WRELP are the binomial strategies, but to solve some instances, some strategies are better than others. To automatically select the best strategy in this paper, we present a new "Multiple" strategy. In "Multiple" strategy, a set of available strategies is initially established. The strategy set contains the best six binomial strategies applicable to the problems. In the initialisation phase, each strategy has the same probability of being selected. From thereon and after each generation a fitness value is assigned to the respective strategy based on its contribution to individual's fitness. This fitness value is used for strategy selection. The strategies are selected using a tournament selection. The application chooses three random strategies and the strategy with the highest fitness wins. After a predefined number of generations (NUMPREVGENS), the probabilities of each strategy are updated based on their contribution in the last generations.

Multiple Strategy:

```
fitIni = fitness(individual)
IF (first time)
  initialise numGensS
  initialise fitPrevGens
  initialise fitnessActual
  numGens = 0
ELSE
  IF (numGens = NUMPREVGENS)
    numGens = 0
    FOR i=1 to NUMSTRATEGIES
      fitnessActual[i] = fitPrevGens[i]/numGensS[i]
    initialise numGensS
    initialise fitPrevGens
numGens = numGens + 1
strat = random(NUMSTRATEGIES)
FOR op = 1 to 3
  op = random(NUMSTRATEGIES)
  IF (fitnessActual[op] > fitnessActual[strat])
    strat = op
SWITCH(strat)
  CASE 1: st = Best1Bin()
  CASE 2: st = Best2Bin()
  CASE 3: st = Best3Bin()
  CASE 4: st = Rand1Bin()
  CASE 5: st = Rand2Bin()
  CASE 6: st = RandBest2Bin()
run(st, individual)
fitEnd = fitness(individual)
fitPrevGens[strat] = fitPrevGens[strat] + fitIni - fitEnd
numGensS[strat] = numGensS[strat] + 1
```

Vector `numGensS`– number of times where each strategy is applied in previous generations.
Vector `fitPrevGens`– total fitness achieved by each strategy in previous generations.
Vector `fitnessActual`– fitness achieved by each strategy considering the `numGensS`.

More information on DE can be found in (Differential Evolution Website; Price et al., 2005).

3.3. Discrete particle swarm optimisation algorithm

The PSO is a randomised, population-based optimisation method that was inspired by the flocking behaviour of birds and human social interactions. This technique was developed by

Eberhart and Kennedy in 1995 and it has been successfully applied to several real-world problems (Eberhart and Kennedy, 1995; Kennedy and Eberhart, 1995). There is little literature reported regarding its application to loading problems. In this paper, PSO is redefined and modified by introducing genetic operators, such as crossover and mutation to update the particles.

PSO shares many similarities with evolutionary computation techniques such as GAs. The system is initialised with a population of random solutions and searches for optimal solutions by updating the generations. In PSO, the potential solutions, called particles, fly through the problems' solution space following the current optimum particles. Whether continuous or discrete, the original and most essential idea of PSO is: difference in position leads to velocity and velocity leads to search.

The DPSO algorithm was proposed by Pan et al. (2008) to solve the no-wait flowshop scheduling problem. The algorithm updates the particle position, based on discrete permutations (Guner and Sevkli, 2008; Tasgetiren et al., 2007). In this paper, we present a Hybrid DPSO algorithm (HDPSO)—a population-based optimisation method coupled with a LS method.

The algorithm HDPSO can be summarised in the following steps:

```
Initialise parameters
Initialise Solutions (particles)
Evaluate Solutions (fitness of each particle)
While number of iterations <=MAXI:
  Create a new population by repeating following
  steps until the new population is complete:
    Update position using the genetic operators
    Evaluate Solution
    Apply LS method
    Evaluate Solution
    Update Best Position of each particle
    Update Global Best Position
Output the best solution
```

We decided to implement this algorithm, since in previous works the application of PSO proved to be very efficient to solve the WRALP (Bernardino et al., 2009b). The DPSO presented in this paper uses a different algorithm model.

3.3.1. Initialisation of parameters

The following parameters must be defined by the user: number of particles (NP), mutation probability (W), mutation operator ($F1$), crossover probabilities ($C1$ and $C2$), crossover operators ($F2$ and $F3$), and maximum number of iterations (MAXI).

3.3.2. Initialisation of solutions

The initial solutions can be randomly created or in a deterministic form using the SPA.

3.3.3. Evaluation of solutions

To evaluate the solutions, we use the fitness function (3) for the WRELP and fitness function (4) for the WRALP.

3.3.4. Updating the positions

Supposing that the searching space is D-dimensional and the p particles form a swarm, each particle is looked as a point in the D-dimensional space, and the ith particle represents a D-dimensional vector $X_i=(X_{i1}, X_{i2},\ldots, X_{iD})$.

According to the fitness value, the particle is updated by the corresponding operators to move towards the best area, until the best point is found. In the iterative process, each particle's previous best position is remembered and denoted $P_i=(P_{i1}, P_{i2},\ldots, P_{iD})$, and the globally best position in the whole swarm is recorded as $G=(G_1, G_2,\ldots, G_D)$.

The ith particle's "flying" velocity is also a D-dimensional vector, represented as $V_i=(V_{i1}, V_{i2},\ldots, V_{iD})$ $(i=1,2,\ldots, p)$.

At each step, the velocity of all particles is adjusted as a sum of its local best value, global best value and its present velocity, multiplied by the three parameters W, $C1$ and $C2$ respectively, demonstrated in (5); the position of each particle is also modified by adding its velocity to the current position, see (6).

$$V_{ij}^{t} = W \times V_{ij}^{t-1} + C1 \times r_1 \times (P_{ij}^{t-1} - X_{ij}^{t-1}) + C2 \times r_2 \times (G_j^{t-1} - X_{ij}^{t-1}) \quad (5)$$

$$X_{ij}^{t} = X_{ij}^{t-1} + v_{ij}^{t-1} \quad (6)$$

In (5) and (6) t represents the iteration number; r_1, r_2 are two random numbers selected from a uniform distribution in $[0.0-1.0]$; W is the inertia weight. $C1$ and $C2$ are two constant numbers, which are often called the acceleration coefficients.

Standard PSO equations cannot be used to generate binary/discrete values, since the positions are real-valued. Pan et al. (2008) have presented a DPSO optimisation algorithm to tackle the discrete spaces. The particles are updated as follows:

$$X_{ij}^{t} = C2 \oplus F3(C1 \oplus F2(W \oplus F1(X_{ij}^{t-1}), P_{ij}^{t-1}), G_j^{t-1}) \quad (7)$$

Given that λ_{ij} , and δ_{ij} are two temporary particles, the update Eq. (7) consists of three operators:

- The first operator is $\lambda_{ij}^{t} = W \oplus F1(X_{ij}^{t-1})$, where $F1$ represents the destruction and construction operator (mutation) with the probability of W. In other words, a uniform random number r is generated between 0 and 1. If r is less than W, then the destruction and construction operator is applied to generate a perturbed particle by $\lambda_{ij}^{t} = F1(X_{ij}^{t-1})$, otherwise, the current particle is kept as $\lambda_{ij}^{t} = X_{ij}^{t-1}$.
- The second operator is $\delta_{ij}^{t} = C1 \oplus F2(\lambda_{ij}^{t}, P_{ij}^{t-1})$, where $F2$ represents the crossover operator with the probability of $C1$. Note that λ_{ij}^{t} and P_{ij}^{t-1} will be the first and second parents for the crossover operator, respectively. It results either in $\delta_{ij}^{t} = F2(\lambda_{ij}^{t}, P_{ij}^{t-1})$ or in $\delta_{ij}^{t} = \lambda_{ij}^{t}$, depending on the choice of a uniform random number.
- The third operator is $X_{ij}^{t} = C2 \oplus F3(\delta_{ij}^{t}, G_j^{t-1})$, where $F3$ represents the crossover operator with the probability of $C2$. Note that δ_i^{t} and G_j^{t-1} will be the first and second parents for the crossover operator, respectively. It results either in $X_{ij}^{t} = F3(\delta_{ij}^{t}, G_j^{t-1})$ or $X_{ij}^{t} = \delta_{ij}^{t}$, depending on the choice of a uniform random number.

For the crossover ($F2$ and $F3$) and for the mutation ($F1$) we use the same operators used in the GA.

3.3.5. Local search

We use the same LS algorithm applied in HDEM.

3.3.6. Updating the best position

For the DPSO algorithm, the P and G (best position and global best position) were calculated using the notation of Kennedy and Eberhart (1995). The best positions are updated by taking into account the fitness of the particle X_{ij}^{t}, and also the fitness of the previous best position P_{ij}^{t-1}. The personal best position of each

particle is updated using:

$$P_{ij}^{t} = \begin{cases} P_{ij}^{t-1} & \text{if}(\text{fitness}(P_{ij}^{t-1}) \leq \text{fitness}(X_{ij}^{t})) \\ X_{ij}^{t} & \text{if}(\text{fitness}(P_{ij}^{t-1}) > \text{fitness}(X_{ij}^{t})) \end{cases}$$

3.3.7. Updating the global best position

The global best position is updated by taking into account the fitness of the best particle position P_{ij}^{t}, and the fitness of the previous global best position G_{ij}^{t-1}. The global best position found so far in the swarm population is obtained as:

$$G_{j}^{t} = \begin{cases} G_{j}^{t-1} & \text{if}(\text{fitness}(G_{j}^{t-1}) \leq \text{fitness}(P_{ij}^{t})) \\ P_{ij}^{t} & \text{if}(\text{fitness}(G_{j}^{t-1}) > \text{fitness}(P_{ij}^{t})) \end{cases}$$

3.3.8. Termination criterion

The algorithm stops when a maximum number of generations (MAXI) is reached.

More information about PSO can be found in Eberhart and Shi (2001), Particle Swarm Optimisation Website.

4. Instances

We evaluate the utility of the algorithms using the same instances used in literature (Bernardino et al., 2010a, b, c). The studied instances arisen by considering six different ring sizes: 5, 10, 15, 20, 25 and 30 nodes. A ring in a telecommunication network will typically contain between 5 and 15 nodes. Thus, the instances consider the 5, 10 and 15 node rings to be ordinary-sized rings and the 20, 25 and 30 node rings to be extremely large rings. It is reasonable to assume that in practice there might be many point-to-point demands. Traffic demands "from-to" pattern have been randomly generated using the node list; the traffic demands randomly selected between 1 and 500 Mb/s. The instances consider complete and partial sets of demands. For a ring with n nodes there are $n^{*}(n-1)/2$ possible source and destination pairs. This is a complete set of demands. Partial sets of demands were generated by randomly setting some fraction of the possible demands to zero. In addition, the instances consider two ranges for the demands: a low variance range over 5 and 100 units, and a high variance range over 1 and 500 units. The demand cases used to create the instances were:

- Case 1: complete set of demands between 5 and 100 with uniform distribution;
- Case 2: half of the demands in Case 1 set to zero;
- Case 3: 75% of the demands in Case 1 set to zero;
- Case 4: complete set of demands between 1 and 500 with uniform distribution. This case was only used for the 30 nodes ring.

We studied 19 instances—3 instances for the 5, 10, 15, 20 and 25 nodes ring and 4 instances for the 30 nodes ring. For convenience, they are labelled C_{ij}, where $1 < i < 6$ represents the ring size and $1 < j < 4$ represents the demand case.

Other authors from literature used similar instances, but with smaller ring sizes or only with a low variance range for the demands (Cho et al., 2005; Karunanithi and Carpenter, 1994; Kubat and Smith, 2005).

5. Results

5.1. Best combination of parameters

We perform comparisons between all parameters of the algorithms using all instances. In this paper we only report the results obtained with instance $C41$ (instance with average difficulty) for the WRALP (we verify that for both problems the parameters have an identical influence). For this instance, were used 50 iterations and the initial solutions were randomly created. The purpose was to obtain the best combination of parameters.

5.1.1. HDPSO best results

The best results obtained with HDPSO use $C1 > 0.6$, $C2$ between 0.5 and 0.7, and W between 0.1 and 0.5 (Figs. 4–6). The results

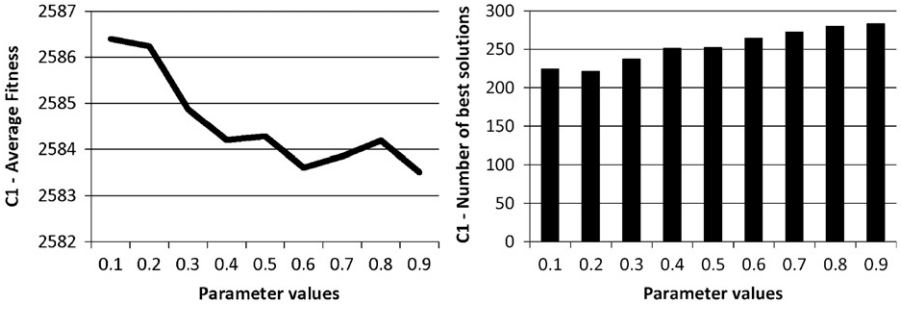


Fig. 4. Influence of the $C1$ parameter on the average fitness and number of best solutions.

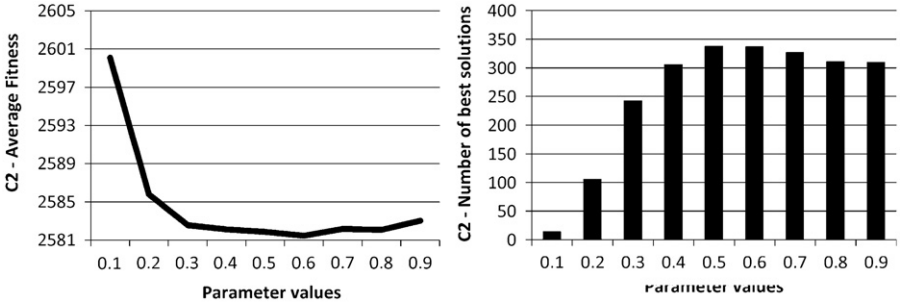


Fig. 5. Influence of the $C2$ parameter on the average fitness and number of best solutions.

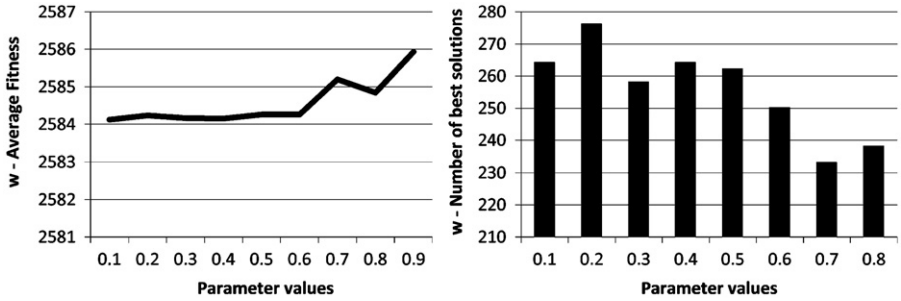


Fig. 6. Influence of the *W* parameter on the average fitness and number of best solutions.

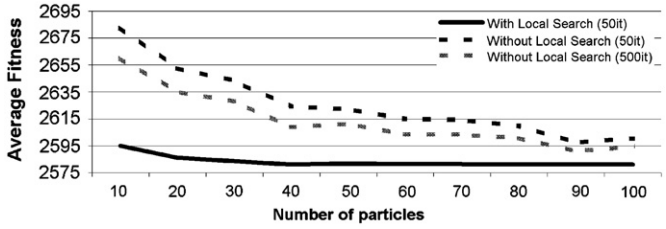


Fig. 7. Influence of the number of particles/LS method on the average fitness.

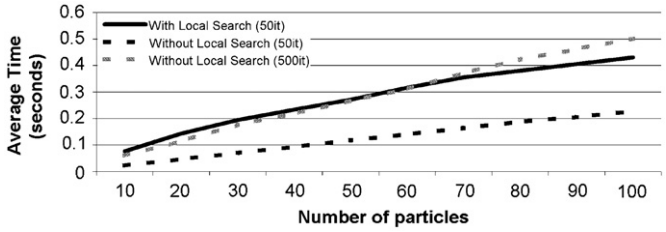


Fig. 8. Influence of the number of particles/LS method on the average time.

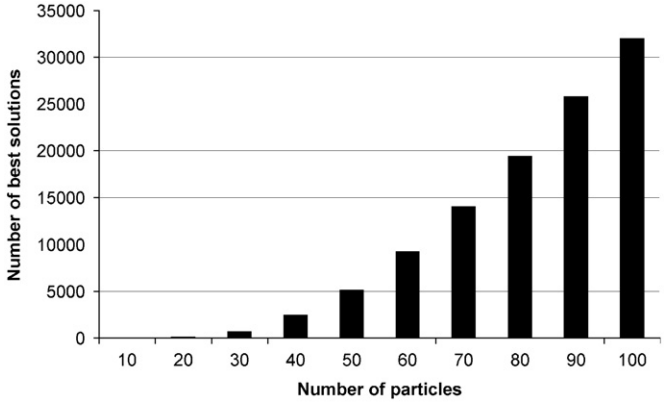


Fig. 9. Influence of the number of particles on the number of best solutions obtained.

shown in Fig. 4, Fig. 5 and Fig. 6 were produced using NP=40, crossover "Uniform" and mutation "Change Direction".

In our experiments for the HDPSO, we used a growing number of particles. This number was set to {10, 20, 30, 40, 50, 60, 70, 80, 90, 100}. We studied the impact on the average fitness (Fig. 7), execution time (Fig. 8) and the number of best solutions (Fig. 9) using all combination of parameters.

We observe that a small number of particles allows an initial faster convergence, but a worse final result, following an increased amount of suboptimal values. With a number of particles between 30 and 60, the algorithm reaches a good number of best solutions in a reasonable amount of time. With a higher number of particles, the algorithm can reach a better average fitness, however it is more time consuming (Fig. 8).

With the LS algorithm, the DPSO produces a better average fitness (Fig. 7). The time increases with the application of the LS algorithm however the overall performance is improved (Fig. 8). Without the LS method and even with 500 iterations, the algorithm achieved a worst average fitness (Fig. 7).

The best crossover operators (*F*2 and *F*3) are: "Uniform" and "Multiple". The best mutation operators are "Multiple" and "Change Direction" (*F*1). With these operators, the algorithm can reach a better average fitness (Fig. 10) and also produce a higher number of solutions (Fig. 11). A great advantage of the proposed "Multiple" operators is that they allow, through their execution, to select the crossover/mutation operators that are better adapted to the problems' resolution. With the multiple operators, independently of the problem or instance to be solved if the chosen crossover/mutation operators are the multiple, there is the guarantee that the best crossover/mutation operators will be selected (based on their contributions to the fitness values). We also observe that the combination of multiple mutation operators during the execution provides a positive impact on the fitness of the solutions.

We did not observe significant changes in the execution time using different crossover/mutation operators.

5.1.2. GA best results

In the tests carried out with the GA, was verified that "Tournament" is the selection method that obtains solutions with smaller fitness values and in a smaller execution time (Fig. 12). The best crossover methods used are "Uniform" and "Multiple" (like in HDPSO), with probability in the range [0.6–0.9] (Fig. 13). The best mutation strategy was the "Multiple", with probability in the range [0.5–0.7] (Fig. 13).

Like in other algorithms, for the GA we used a growing number of individuals in the initial population. This number was set to {100, 200, 300, 400, 500}. We studied the impact on the execution time (Fig. 14—Right) and the average fitness (Fig. 14—Left) using all combination of parameters. With a number of individuals between 200 and 300, the algorithm reaches a good average fitness in a reasonable amount of time. With a higher number of individuals, the algorithm can reach a better average fitness (there is more diversity), however it is more time consuming.

5.1.3. HDEM best results

The best results obtained with HDEM use crossover probability (CR) and factor (*F*) between [0.4–0.6] and [0.2–1], respectively (Fig. 15). The tests carried out have demonstrated that the strategies with best results are the binomial strategies (Fig. 16). The "Multiple" strategy obtains a higher percentage of best solutions (Fig. 17). A great advantage of the proposed "Multiple"

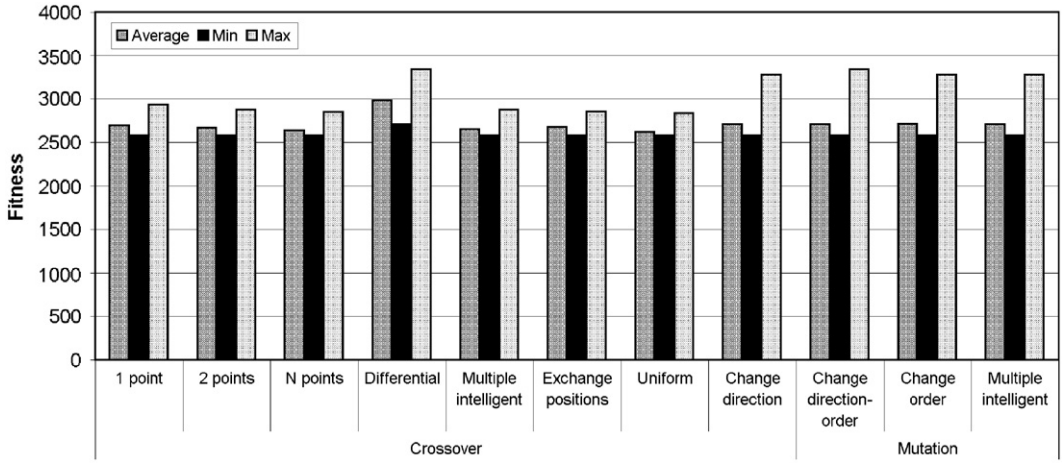


Fig. 10. Influence of the crossover/mutation operators on the fitness values using NP=40.

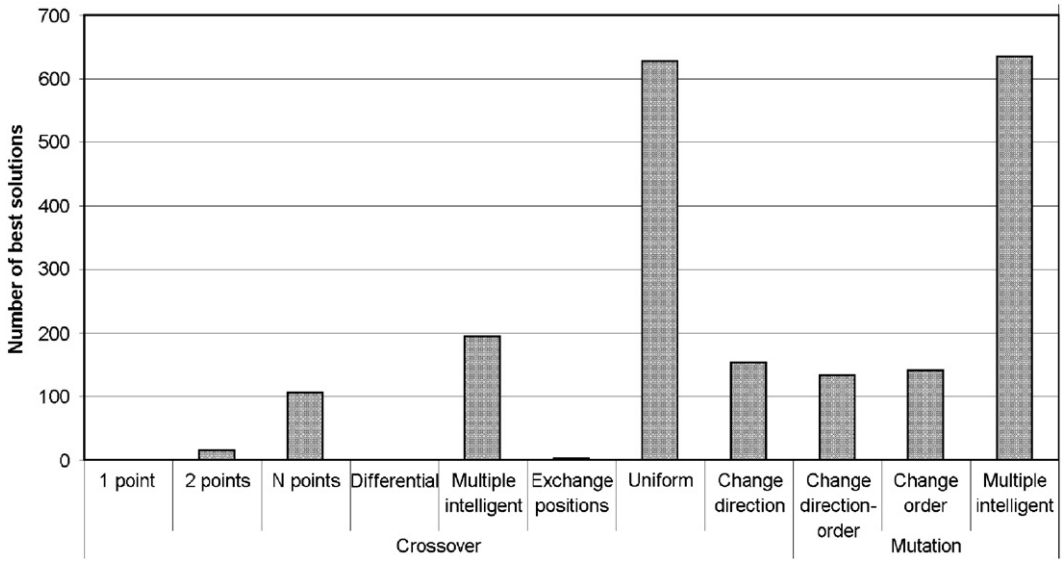


Fig. 11. Influence of the crossover/mutation operators on the number of best solutions obtained using NP=40.

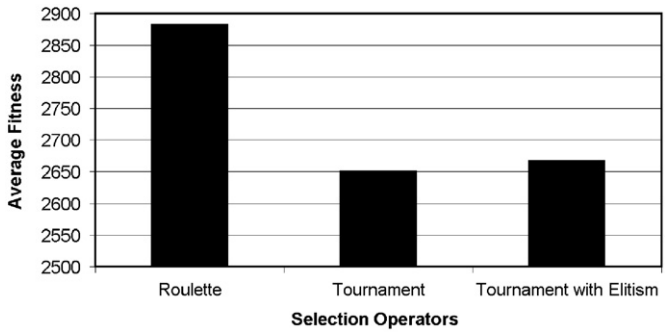


Fig. 12. Influence of the GA selection operators on the average fitness.

strategy is that it allows, through its execution, to select the strategies that are better adapted to the problem resolution.

When using the correct combination of parameters, the "Best1-Bin" and "Multiple" strategies obtain the best solution in almost all executions. We did not observe significant changes in executing time using different strategies.

For the HDEM, we used a growing number of individuals in the initial population. This number was set to {100, 200, 300, 400, 500}. We studied the impact on the number of best solutions obtained (Fig. 17) using all combination of parameters. We also studied the impact on the average fitness and execution time using the strategy "Best1Bin". With a number of individuals between 100 and 200, the algorithm reaches a good number of best solutions in a reasonable amount of time (Fig. 18). We also verify that with only 50 individuals in the population and with the correct combination of parameters, the algorithm reaches almost always the best solution.

Large types of experiments and considerations have been made to define other parameters. In general, experiments have demonstrated that the proposed parameter setting is very robust to the problems tested.

5.2. Comparison with other algorithms from literature

In this paper, we only compare our algorithms with the algorithms TS (Bernardino et al., 2009a), LS-PBPSO (Bernardino et al., 2009b), HDE (Bernardino et al., 2009a), ABC (Bernardino et al., 2010a), HACO (Bernardino et al., 2010b) and DDE (Bernardino et al., 2010c) because they: (1) used the same test instances; (2) adopted the same fitness function; (3) implemented the algorithms using

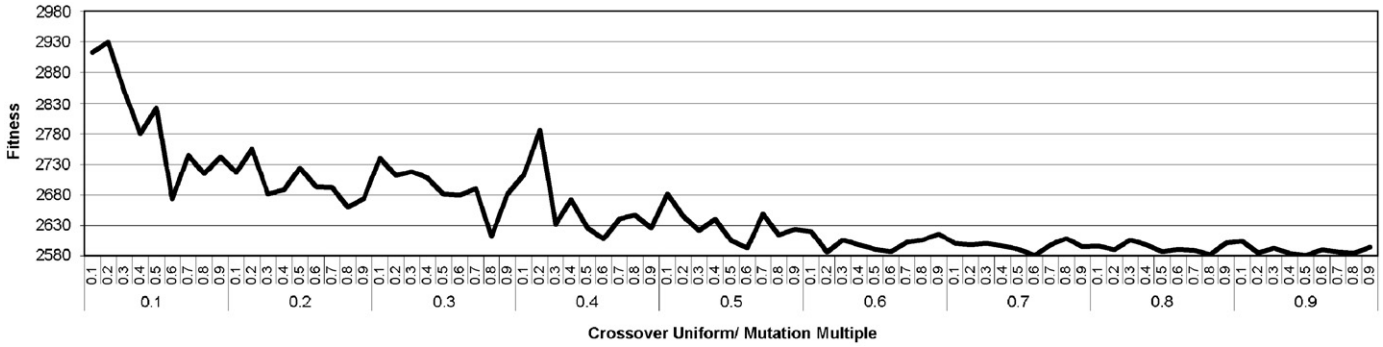


Fig. 13. Influence of the crossover/mutation probabilities on the fitness values.

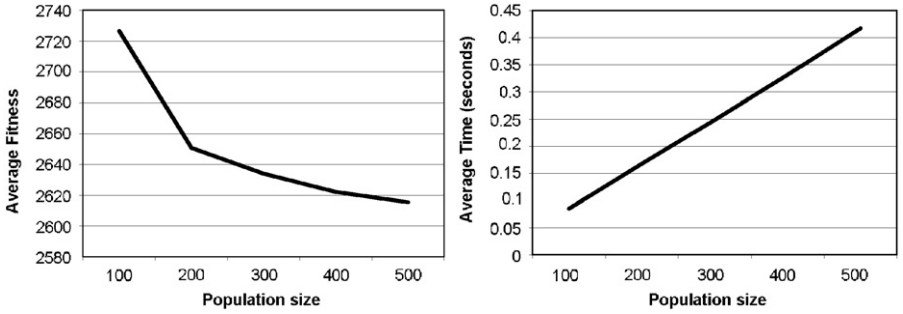


Fig. 14. Influence of the GA population size on the average fitness and on the average time.

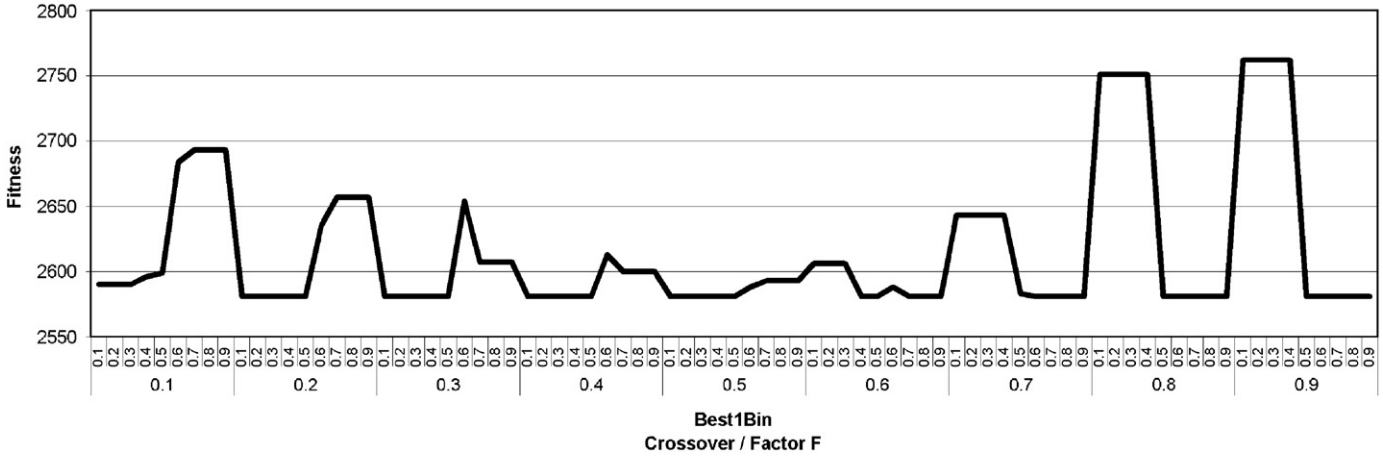


Fig. 15. Influence of the crossover/factor *F* probabilities on the fitness values.

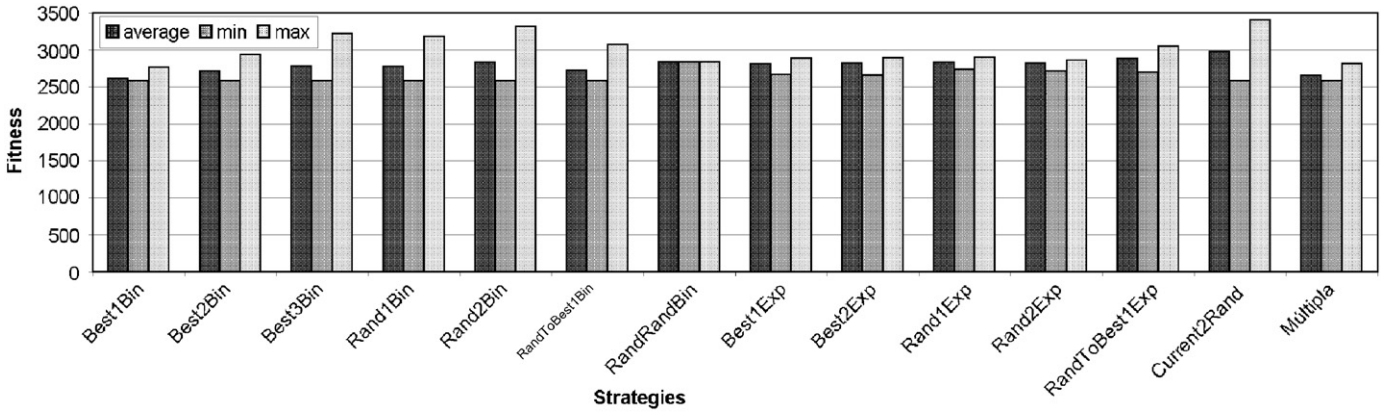
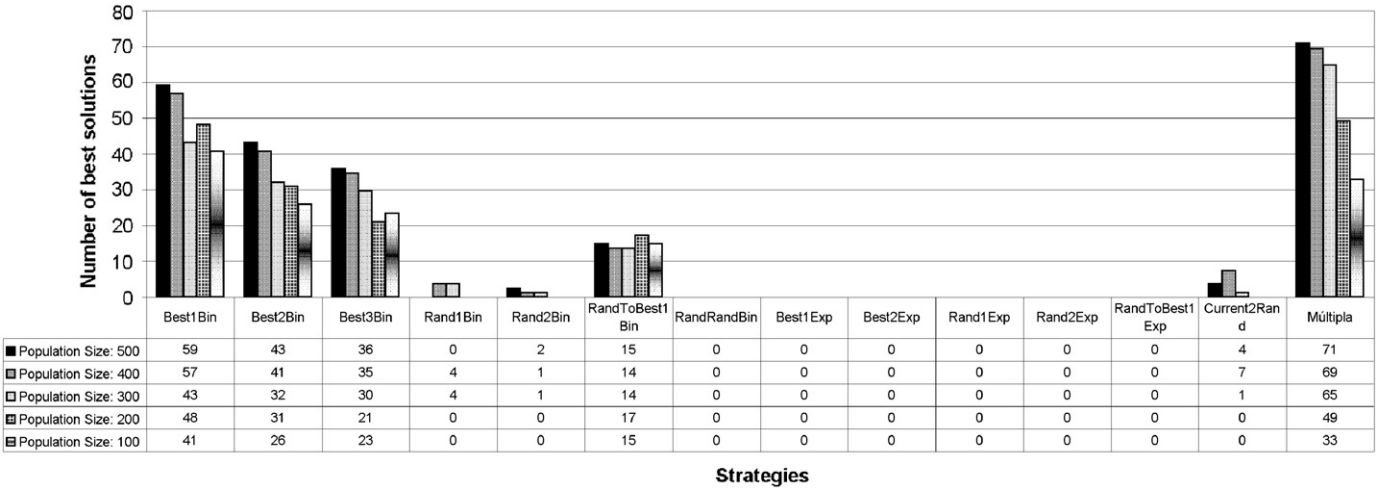


Fig. 16. Influence of the strategies on the fitness values.



	Best1Bin	Best2Bin	Best3Bin	Rand1Bin	Rand2Bin	RandToBest1 Bin	RandRandBin	Best1Exp	Best2Exp	Rand1Exp	Rand2Exp	RandToBest1 Exp	Current2Rand	Múltipla
Population Size: 500	59	43	36	0	2	15	0	0	0	0	0	0	4	71
Population Size: 400	57	41	35	4	1	14	0	0	0	0	0	0	7	69
Population Size: 300	43	32	30	4	1	14	0	0	0	0	0	0	1	65
Population Size: 200	48	31	21	0	0	17	0	0	0	0	0	0	0	49
Population Size: 100	41	26	23	0	0	15	0	0	0	0	0	0	0	33

Fig. 17. Influence of the number of individuals and the HDEM strategies on the percentage of best solutions.

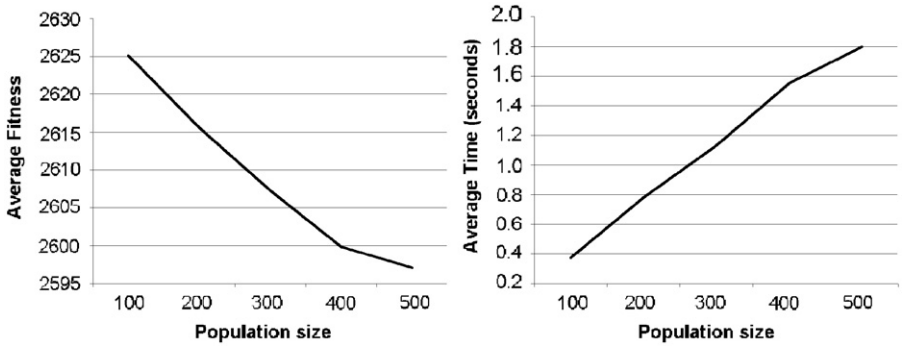


Fig. 18. Influence of the HDEM population size on the average fitness and on the average time.

the same language (C++), and; (4) adopted the same representation (binary). Other authors from literature have used the standard GA (Karunanithi and Carpenter, 1994) to solve the WRELP but the study of Bernardino et al. (2008) that used a GA with new operators, produce significant better results. Other authors from literature have used the ACO algorithm (Kim et al., 2008) to solve the WRELP, but it was not possible to compare our results with them, because they use different instances. Since Bernardino et al. (2010b) have proposed a Hybrid ACO algorithm with important improvements to solve the WRALP, we consider their results for comparison. Since our purpose is to obtain the best-known solution, it was not possible to compare with other heuristics from literature.

TS (Glover, 1986) is a metaheuristic algorithm, which belongs to the class of LS techniques. The basic concept of TS was described by Glover (1986). TS allows the search of solutions that decrease the objective function value only in those cases where these solutions are not forbidden (Glover and Laguna, 1997). Bernardino et al. (2009a) proposed a TS algorithm to solve WRELP and compared the results with those found with GA, DE and HDE.

PBPSO was proposed by Wang et al. (2008). In PBPSO, a novel updating strategy is adopted to update the swarm and search the global solution. The standard equations used to calculate the velocities and positions of the particles are all reserved for iterative evolution in PBPSO, and a different formula is used to determine new positions for the particles. Bernardino et al. (2009b) used this algorithm, combined with a LS method to solve the WRALP and compared it with other three variants of PSO. The LS-PBPSO was the algorithm that achieved best results.

ACO is a population-based optimisation method to solve hard combinatorial optimisation problems. The first ACO algorithm was presented by Dorigo et al. (1991, 1996), Dorigo (1991) and since then, many diverse variants of the basic principle have been reported in literature (Ant Colony Optimisation Website). It is based on the indirect communication of a colony of simple agents, called (artificial) ants, mediated by (artificial) pheromone trails. Gambardella et al. (1999) present a Hybrid Ant Colony System coupled with a LS (HAS_QAP), applied to the Quadratic Assignment Problem (QAP). HAS-QAP uses pheromone trail information to perform modifications on QAP solutions. The Hybrid ACO (HACO) algorithm proposed by Bernardino et al. (2010b) also uses pheromone trail information to perform modifications on WRALP solutions, unlike traditional ant systems that use pheromone trail information to construct complete solutions. The produced results were compared with GA and LS-PBPSO.

ABC is also a population-based optimisation method and it has been successfully applied to different optimisation problems. The ABC algorithm proposed by Karaboga (2005) to optimise numerical problems is one of the most recently introduced swarm-based algorithms. ABC simulates the intelligent behaviour of a bee colony (Artificial Bee Colony Algorithm Website; Karaboga and Akay, 2009). Bernardino et al. (2010a) have used this algorithm combined with a LS method to solve the WRALP. The authors compared their results with the ones obtained by the GA, TS and LS-PBPSO algorithms.

DDE algorithm was proposed by Pan et al. (2007) to solve the permutation flowshop scheduling problem. The DDE algorithm

first mutates a target population to produce the mutant population. Then the target population is recombined with the mutant population in order to generate a trial population. Finally, a selection operator is applied to both target and trial populations to determine who will survive for the next generation (Pan et al., 2007). Bernardino et al. (2010c) used a LS method embedded in the DDE algorithm to solve the WRALP and compared the results with GA, DE, TS and LS-PBPSO.

Suggestions from literature helped to guide our choice of parameter values for the TS, LS-PBPSO, HDE, ABC, HACO and DDE. We use the same values proposed by Bernardino et al. (2008, 2009a, b, 2010a, b, c) (see Table 3).

The nine algorithms were executed using a processor Intel Quad Core Q9450. The initial solutions of the nine algorithms were randomly created. For the instance *C*64, the SPA was used to create the initial populations.

In a first stage, we use different stop conditions for the algorithms. We use (1) a predefined number of evaluations, (2) a predefined number of seconds, and (3) a predefined number of iterations. We observe that EAs present a very poor performance (higher average fitnesses and standard deviations) if we compare them in terms of number of evaluations. This happens because the tested EAs need more population diversity. In each iteration, the EAs perform a higher number of evaluations in comparison with SI algorithms. We observe that SI algorithms can obtain very good results using small populations. The improvements using larger populations are not significant. Larger populations are more time consuming and do not provide significant better solutions using the SI algorithms. For that reason, we consider that the number of evaluations is not suitable to perform comparisons between EAs and SI algorithms.

We also observe the execution time per iteration. When using the same number of individuals in the population, we observe that SI algorithms are more time consuming, because they use probability vectors/matrices. Since SI algorithms use smaller populations, the differences between SI and EAs in terms of execution time are not relevant.

We establish the number of iterations based on preliminary observations on the convergence of the algorithms.

Table 4 presents the best obtained results. The first column represents the instance number (Instance), the second and third columns demonstrate the number of nodes (Nodes) and the number of pairs (Pairs), the fourth and fifth columns demonstrate the minimum fitness values obtained for WRALP and WRELP, and the sixth column demonstrates the number of iterations used to test each instance.

Table 5 presents the best WRALP results and Table 6 the best WRELP results obtained with the nine algorithms. The first column represents the instance number (Inst.), and the remaining columns demonstrate the results obtained (Time – Run Times in seconds, IT – Iterations) by the nine algorithms. The values presented have been computed based on 100 different executions for each test instance using the best combination of parameters found and different seeds. Tables 5 and 6 only consider the 30 best executions.

The nine algorithms reach feasible solutions for all test instances and all the algorithms reach the best solutions before the run times and number of iterations presented.

Table 7 presents the WRALP average fitness and the WRALP average time obtained with GA, HDEM, HDPSO, TS, LS-PBPSO, HDE, ABC, HACO and DDE using a limited number of iterations for the instances *C*41, *C*51 and *C*61 (harder instances). The first column represents the number of the instance (Instance), the second column demonstrates the number of iterations used to test each instance and the remaining columns show the results obtained (AF—Average Fitness, AT—Average Time in seconds, ST—Standard Deviation) by the nine algorithms. The results have been computed based on 100 different executions for each test instance using the best combination of parameters found and different seeds.

Table 3
Parameter values used for comparison.

Algorithm	Parameter name	Parameter values
GA	Number of individuals	200
	Crossover probability	[0.6, 0.9]
	Selection operator	"Tournament"
	Mutation probability	[0.5, 0.7]
	Crossover operator	"Uniform"
	Mutation operator	"Multiple"
HDEM	Number of individuals	50
	Crossover probability	[0.3, 0.5]
	Factor *F*	[0.4, 0.6]
	Strategy	"Multiple"
HDPSO	Number of particles	40
	Number of modifications	[3 … 10]
	Crossover operator (*F*2, *F*3)	"Uniform"
	Mutation operator (*F*1)	"Multiple"
	Crossover probability *C*1	[0.6, 0.7]
	Crossover probability *C*2	[0.6, 0.7]
	Mutation probability *W*	[0.4, 0.5]
TS	Number of elements in the tabu list	[4, 10]
LS-PBPSO	Number of individuals	40
	Constant *C*1	1.49
	Constant *C*2	1.49
	Inertia velocity *W*	[0.6, 0.8]
HDE	Number of individuals	50
	Crossover probability	[0.3, 0.5]
	Factor *F*	[0.5, 0.7]
	Strategy	"Best1Bin"
HACO	Number of ants	40
	Q	100
	Probability exploitation/exploration	[0.7, 0.8]
	Pheromone influence	[0.7, 0.8]
	Pheromone evaporation	[0.6, 0.8]
	Number of modifications	30
ABC	Number of Employed Bees	[10, 20]
	Number of Onlooker Bees	[25, 35]
	Number of modifications	[10, 15]
	LS method	"Exchange Max Arc"
	Scouts creation	"Randomly"
DDE	Number of individuals	50
	Number of perturbations	5
	Perturbation probability	[0.6, 0.8]
	Crossover probability	[0.1, 0.2]
	LS method	"Exchange Direction"

5.3. Discussion

In comparison, the HDEM algorithm produces a higher number of best solutions using the same number of iterations (Fig. 19). The HDPSO and ABC algorithms obtain a reasonable number of best solutions in a better running time (Fig. 19, Table 7). We verify that the HDPSO needs a higher number of iterations comparing with the ABC, however it produces solutions for some harder instances in smaller times (Tables 5 and 6). The TS is the slowest algorithm and it obtains a smaller number of best solutions in comparison with other algorithms (Fig. 19).

When using the SPA to create the initial solutions, the times and number of iterations decreases—instance *C*64. This instance is computationally harder than the *C*61, however the best solution is obtained faster. Based on preliminary observations, we consider to be more efficient to initially apply a SPA and, after, the metaheuristic to improve the solutions.

Table 4
Best obtained results.

Instance	Nodes	Pairs	Best fitness WRALP	Best fitness WRELP	Number iterations
C11	5	10	161	185	25
C12	5	8	116	137	10
C13	5	6	116	137	10
C21	10	45	525	583	50
C22	10	23	243	352	25
C23	10	12	141	199	10
C31	15	105	1574	1657	100
C32	15	50	941	941	50
C33	15	25	563	618	25
C41	20	190	2581	2745	300
C42	20	93	1482	1760	100
C43	20	40	612	683	50
C51	25	300	4265	4304	500
C52	25	150	2323	2488	400
C53	25	61	912	1015	250
C61	30	435	5762	5953	1500
C62	30	201	2696	2901	1000
C63	30	92	1453	4304	500
C64	30	435	27,779	29,245	500

Table 5
WRALP results—run times and number of iterations.

Inst.	GA		HDEM		HDPSO		TS		LS-PBPSO	
	Time (s)	IT	Time (s)	IT	Time (s)	IT	Time (s)	IT	Time (s)	IT
C11	< 0.001	2	< 0.001	2	< 0.001	2	< 0.001	5	< 0.001	2
C12	< 0.001	2	< 0.001	2	< 0.001	2	< 0.001	5	< 0.001	2
C13	< 0.001	1	< 0.001	1	< 0.001	1	< 0.001	1	< 0.001	1
C21	< 0.001	15	< 0.001	10	< 0.001	15	< 0.001	25	< 0.001	15
C22	< 0.001	5	< 0.001	3	< 0.001	5	< 0.001	5	< 0.001	3
C23	< 0.001	3	< 0.001	3	< 0.001	5	< 0.001	5	< 0.001	3
C31	0.1	30	0.1	10	0.1	20	0.1	90	0.1	20
C32	< 0.001	15	< 0.001	5	< 0.001	10	< 0.001	30	< 0.001	8
C33	< 0.001	5	< 0.001	3	< 0.001	5	< 0.001	20	< 0.001	5
C41	0.1	50	0.075	25	0.08	45	0.2	220	0.2	50
C42	0.075	40	0.03	10	0.04	25	0.1	85	0.075	20
C43	< 0.001	10	< 0.001	5	< 0.001	5	< 0.001	25	< 0.001	5
C51	0.75	80	0.5	25	0.5	100	1	260	0.75	80
C52	0.1	40	0.075	15	0.075	30	0.2	110	0.1	25
C53	0.01	25	0.0075	10	0.0075	20	0.05	40	0.01	15
C61	1.75	130	1.5	35	1.25	120	3.5	400	2	130
C62	0.2	60	0.15	20	0.15	50	0.8	230	0.4	50
C63	0.075	30	0.05	10	0.05	20	0.08	100	0.075	15
C64	0.3	30	0.1	3	0.1	5	1	250	0.5	40

Inst.	HDE		ABC		HACO		DDE	
	Time (s)	IT	Time (s)	IT	Time (s)	IT	Time (s)	IT
C11	< 0.001	2	< 0.001	2	< 0.001	2	< 0.001	2
C12	< 0.001	2	< 0.001	2	< 0.001	2	< 0.001	2
C13	< 0.001	1	< 0.001	1	< 0.001	1	< 0.001	1
C21	< 0.001	10	< 0.001	10	< 0.001	20	< 0.001	10
C22	< 0.001	3	< 0.001	3	< 0.001	3	< 0.001	3
C23	< 0.001	3	< 0.001	3	< 0.001	3	< 0.001	3
C31	0.1	15	0.1	15	0.1	30	0.1	10
C32	< 0.001	5	< 0.001	5	< 0.001	10	< 0.001	5
C33	< 0.001	5	< 0.001	3	< 0.001	5	< 0.001	3
C41	0.1	30	0.08	20	0.15	50	0.1	25
C42	0.05	10	0.03	10	0.06	25	0.05	8
C43	< 0.001	5	0.001	3	< 0.001	5	< 0.001	3
C51	0.75	30	0.5	50	0.6	100	0.5	30
C52	0.1	15	0.08	15	0.1	30	0.1	15
C53	0.01	10	0.0075	10	0.01	20	0.01	8
C61	1.75	40	1.5	80	1.75	150	1.5	50
C62	0.25	20	0.15	30	0.4	60	0.25	25
C63	0.075	10	0.05	10	0.075	20	0.06	10
C64	0.25	5	0.1	5	0.5	5	0.1	3

Table 6
WRELP results—run times and number of iterations.

Inst.	GA		HDEM		HDPSO		TS		LS-PBPSO	
	Time (s)	IT	Time (s)	IT	Time (s)	IT	Time (s)	IT	Time (s)	IT
C11	< 0.001	2	< 0.001	2	< 0.001	2	< 0.001	6	< 0.001	5
C12	< 0.001	1	< 0.001	1	< 0.001	1	< 0.001	1	< 0.001	1
C13	< 0.001	1	< 0.001	1	< 0.001	1	< 0.001	1	< 0.001	1
C21	< 0.001	15	< 0.001	15	< 0.001	20	< 0.001	60	< 0.001	20
C22	< 0.001	10	< 0.001	10	< 0.001	10	< 0.001	30	< 0.001	10
C23	< 0.001	5	< 0.001	5	< 0.001	5	< 0.001	15	< 0.001	5
C31	0.1	30	0.1	15	0.1	20	0. 2	120	0.1	30
C32	< 0.001	20	< 0.001	10	< 0.001	15	0.002	50	0.002	15
C33	< 0.001	10	< 0.001	5	< 0.001	5	< 0.001	30	< 0.001	5
C41	0.15	100	0.1	40	0.1	60	0.3	300	0.15	100
C42	0.075	40	0.05	15	0.05	25	0.1	100	0.06	30
C43	< 0.001	10	< 0.001	5	< 0.001	5	0.002	40	< 0.001	10
C51	1.5	150	1.25	100	1	150	2	500	1.5	150
C52	0.15	60	0.15	30	0.1	50	0.3	150	0.15	40
C53	0.01	25	0.01	10	0.01	20	0.05	50	0.01	15
C61	2.25	150	2	50	1.75	150	5	1000	2.5	200
C62	0.3	70	0.3	30	0.25	50	0.8	250	0.5	70
C63	0.075	30	0.075	10	0.05	20	0.08	100	0.075	15
C64	0.3	40	0.25	10	0.25	40	5	1000	3	300

	HDE		ABC		HACO		DDE	
	Time (s)	IT	Time (s)	IT	Time (s)	IT	Time (s)	IT
C11	< 0.001	2	< 0.001	2	< 0.001	4	< 0.001	2
C12	< 0.001	1	< 0.001	1	< 0.001	1	< 0.001	1
C13	< 0.001	1	< 0.001	1	< 0.001	1	< 0.001	1
C21	< 0.001	20	< 0.001	15	< 0.001	20	< 0.001	15
C22	< 0.001	10	< 0.001	10	< 0.001	10	< 0.001	10
C23	< 0.001	5	< 0.001	5	< 0.001	5	< 0.001	5
C31	0.1	20	0.1	25	0.1	30	0.1	15
C32	< 0.001	10	< 0.001	10	< 0.001	15	< 0.001	10
C33	< 0.001	10	< 0.001	10	< 0.001	10	< 0.001	5
C41	0.15	50	0.1	75	0.15	100	0.1	50
C42	0.075	20	0.06	15	0.075	25	0.05	15
C43	< 0.001	5	< 0.001	5	< 0.001	10	< 0.001	5
C51	1.5	130	1.25	100	1.5	250	1.25	80
C52	0.15	40	0.1	30	< 0.15	50	0.15	25
C53	0.01	15	0.01	15	< 0.01	20	0.01	15
C61	2.25	70	1.85	150	2	200	2	80
C62	0.4	40	0.25	30	0.3	60	0.25	30
C63	0.075	10	0.05	10	0.075	20	0.075	15
C64	0.3	20	0.25	30	0.3	40	0.25	25

Table 7
Results WRALP—average time/average fitness/standard deviation.

Inst	It	GA			HDEM			HDPSO			TS			LS-PBPSO		
		AF	AT	SD	AF	AT	SD	AF	AT	SD	AF	AT	SD	AF	AT	SD
C41	50	2587,62	0,17	3,46	2581,36	0,19	0,85	2581,82	0,16	1,42	2635,28	0,16	20,83	2594,36	0,26	7,70
C51	75	4273,18	0,43	2,97	4266,46	0,73	1,43	4266,81	0,58	2,05	4392,70	0,86	34,07	4291,52	0,86	16,85
C61	100	5784,62	1,34	10,05	5764,06	2,03	3,75	5777,56	1,55	9,43	5963,14	3,71	41,93	5837,58	3,10	23,19

Inst	It	HDE			ABC			HACO			DDE		
		AF	AT	SD	AF	AT	SD	AF	AT	SD	AF	AT	SD
C41	50	2584,31	0,27	1,15	2581,5	0,21	0,93	2591,23	0,16	7,73	2582,06	0,16	1,18
C51	75	4271,27	0,7	5,10	4266,52	0,77	1,6	4279,49	0,76	10,10	4268,96	0,53	5,71
C61	100	5783,18	1,87	7,45	5764,36	2,1	5,23	5793,68	2,23	14,17	5781,52	1,39	9,78

Some instances of the WRELP are computationally harder to solve comparing with the WRALP, nevertheless our algorithms prove to be very effective to solve both problems.

The HDEM is the algorithm that presents the best average fitness and the best standard deviation. However, HDPSO is faster and it also presents a good average fitness (Table 7).

6. Conclusions

In this paper we present three nature inspired algorithms with new operators and new modifications to improve the final results. We propose a GA with multiple operators, a HDEM algorithm and a HDPSO algorithm to solve two well-known loading problems:

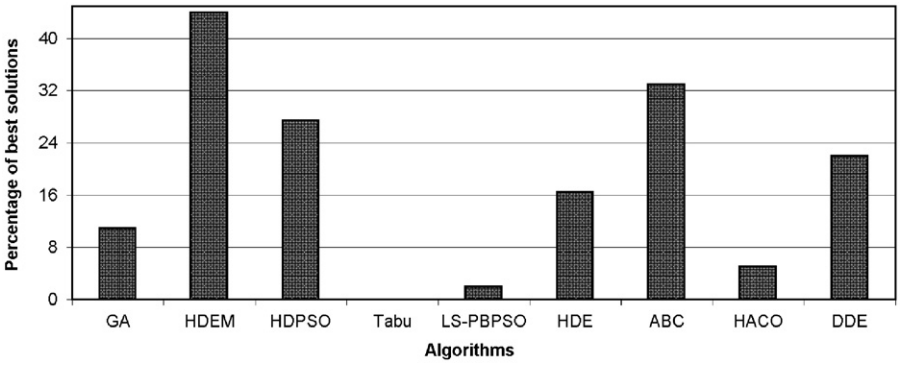


Fig. 19. Percentage of best solutions obtained by the nine algorithms—instance *C*41 (50 iterations).

the WRALP and the WRELP without demand splitting. The performance of our algorithms is compared with several algorithms: TS, LS-PBPSO, HDE, HACO, ABC and DDE used in literature to solve the same studied instances.

One of the main contributions of this paper is to show that nature inspired algorithms can be used in hybrid synthesis with other heuristics for solving the WRLP. In this paper we applied a separate LS method in the algorithms: DE and DPSO. The expanding neighbourhood search strategy is used in order to improve the solutions of each particle/solution in the swarm/population. We also proved that when using the SPA to create the initial solutions the best solution is obtained faster.

A second contribution is the application of multiple operators/strategy in the algorithms: GA and HDEM. A great advantage of the proposed "Multiple" strategy/operators is that through their execution, they allow to select the strategy or the crossover/mutation operators that are better adapted to the problems' resolution.

The three algorithms proposed were tested on a set of nineteen instances. The algorithms were thoroughly tested with different parameter values to establish the correct combination of parameters and obtain the best possible solution. These tests have proved the computational efficiency of the proposed algorithms.

Experimental results and comparisons performed with other algorithms, demonstrate that the proposed algorithms are effective and competitive approaches in composing fairly satisfactory results with respect to solution quality and execution time for the studied problems. HDEM provides a higher number of best solutions, a better average fitness and a better standard deviation using the same number of iterations. HDPSO provides solutions in smaller times for larger instances.

In literature the application of HDPSO for these two problems is nonexistent. For that reason this article demonstrates its enforceability in the resolution of these problems.

The continuation of this work will be the search and implementation of new methods to speed up the optimisation process.

Acknowledgements

This work has been partially supported by the Polytechnic Institute of Leiria (Portugal) and the MSTAR Project Reference: TIN2008-06491-C04-04/TIN (MICINN Spain).

References

Ant colony optimization website, 〈http://iridia.ulb.ac.be/dorigo/ACO/ACO.html〉.
Artificial bee colony algorithm website, 〈http://mf.erciyes.edu.tr/abc/〉.
Bernardino AM, Bernardino EM, Sánchez-Pérez JM, Vega-Rodríguez MA, Gómez-Pulido JA. A discrete differential evolution algorithm for solving the weighted ring arc loading problem. In: Proceedings the 23rd international conference on industrial, engineering and other applications of applied intelligent systems applications of evolutionary computation. Springer Berlin/Heidelberg; 2010a. p. 61–70.
Bernardino AM, Bernardino EM, Sánchez-Pérez JM, Vega-Rodríguez MA, Gómez-Pulido JAA. Hybrid ant colony optimization algorithm for solving the ring arc-loading problem. In: artificial intelligence: theories, models and applications, 6th hellenic conference on AI, SETN 2010. Springer Berlin/Heidelberg; 2010b. p. 49–59.
Bernardino AM, Bernardino EM, Sánchez-Pérez JM, Vega-Rodríguez MA, Gómez-Pulido JA. Efficient load balancing for a resilient packet ring using artificial bee colony. In: Proceedings of applications of evolutionary computation, EvoApplications 2010: EvoCOMNET, EvoENVIRONMENT, EvoFIN, EvoMUSART, and EvoTRANSLOG. Springer Berlin/Heidelberg; 2010c. p. 61–70.
Bernardino AM, Bernardino EM, Sánchez-Pérez JM, Vega-Rodríguez MA, Gómez-Pulido JA. Solving the non-split weighted ring arc-loading problem in a resilient packet ring using particle swarm optimization. In: Proceedings of international conference in evolutionary computation. Madeira; 2009b. p. 230–6.
Bernardino AM, Bernardino EM, Sánchez-Pérez JM, Vega-Rodríguez MA, Gómez-Pulido JA. Solving the ring loading problem using genetic algorithms with intelligent multiple operators. In: Proceedings of international symposium on distributed computing and artificial intelligence 2008 (DCAI 2008). Springer Berlin/Heidelberg; 2008. p. 235–44.
Bernardino AM, Bernardino EM, Sánchez-Pérez JM, Vega-Rodríguez MA, Gómez-Pulido JA. Solving the weighted ring edge-loading problem without demand splitting using a hybrid differential evolution algorithm. In: Proceedings of the 34th IEEE conference on local computer networks. IEEE Press; 2009a. p. 562–8.
Cho KS, Joo UG, Lee HS, Kim BT, Lee WD. Efficient load balancing algorithms for a resilient packet ring. ETRI Journal 2005;27(1):110–3.
Cosares S, Saniee I. An optimisation problem related to balancing loads on SONET rings. Telecommunication Systems 1994;3(2):165–81.
Davik F, Yilmaz M, Gjessing S, Uzun N. IEEE 802.17 resilient packet ring tutorial. IEEE Communications Magazine 2004;42(3):112–8.
Dell'Amico M, Labbé M, Maffioli F. Exact solution of the SONET ring loading problem. Operations Research Letters 1999;25(3):119–29.
Differential Evolution Website, 〈http://www.icsi.berkeley.edu/~storn/code.html〉.
Dorigo M, Maniezzo V, Colorni A. Positive feedback as a search strategy. Technical report 91-016, Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy; 1991.
Dorigo M, Maniezzo V, Colorni A. The ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics 1996;26:29–41.
Dorigo M. Ottimizzazione, apprendimento automatico, ed algoritmi basati su metafora naturale (optimisation, learning and natural algorithms). Doctoral dissertation, Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy; 1991.
Eberhart RC, Kennedy J. A new optimiser using particle swarm theory. In: Proceedings of the sixth international symposium on micro machine and human science, Nagoya, Japan; 1995, p. 39–43.
Eberhart RC, Shi Y. Particle swarm optimization: developments, applications and resources. In: Proceedings 2001 congress evolutionary computation. IEEE Press; 2001. p. 81–6.
Eiben A, Smith J. Introduction to evolutionary computing. Berlin: Springer; 2003.
Gambardella LM, Taillard ED, Dorigo M. Ant colonies for the quadratic assignment problem. Journal of the Operational Research Society 1999;50(2):167–76.
Glover F, Laguna M. Tabu search. Kluwer Academic Publishers; 1997.
Glover F. Future paths for integer programming and links to artificial intelligence. Computers and Operations Research 1986;13(5):533–49.
Goldberg D. Genetic algorithms in search. Optimization and machine learning. USA: Addison Wesley; 1989.
Goldschmidt O, Laugier A, Olinick EV. SONET/SDH ring assignment with capacity constraints. Discrete Applied Mathematics 2003;129(1):99–128.
Goralski WJ. SONET. McGraw-Hill Professional; 2002.
Guner AR, Sevkli MA. Discrete particle swarm optimisation algorithm for uncapacitated facility location problem. Journal of Artificial Evolution and Applications 2008;2008:1–9.

Holland JH. Adaptation in natural and artificial systems. The University of Michigan Press; 1975.
Karaboga D, Akay BA. Comparative study of artificial bee colony algorithm. Applied Mathematics and Computation 2009;214:108–32.
Karaboga D. An idea based on honey bee swarm for numerical optimization. Technical report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department; 2005.
Karunanithi N, Carpenter TA. Ring loading application of genetic algorithms. In: Proceedings of the ACM symposium on applied computing. ACM, New York, USA; 1994. p. 227–31.
Kennedy J, Eberhart RC, Shi Y. Swarm intelligence.. 1st ed. San Francisco, CA: Morgan Kaufmann; 2001.
Kennedy J, Eberhart RC Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks. IEEE Press; 1995. p.1942–48.
Kim S-S, Kim I-H, Mani V, Kim HJ. Ant colony optimization for SONET ring loading problem. International Journal of Innovative Computing, Information and Control 2008;4(7):1617–26.
Kubat P, Smith JM. Balancing traffic flows in resilient packet rings. In: Girard André, editor. Performance evaluation and planning methods for the next generation internet. Springer; 2005. p. 125–40.
Lee CY, Chang SG. Balancing loads on SONET rings with integer demand splitting. Computers and Operations Research 1997;24(3):221–9.
Memetic Algorithms Website, 〈http://www.ing.unlp.edu.ar/cetad/mos/memetic_home.html〉.
Moscato P. On evolution, search, optimization, genetic algorithms and martial arts: toward memetic algorithms. Caltech concurrent computation program, C3P Report 826; 1989.
Myung Y-S, Kim H-G, Tcha D-W. Optimal load balancing on SONET bidirectional rings. Operations Research 1997;45(1):148–52.
Myung Y-S, Kim H-G. On the ring loading problem with demand splitting. Operations Research Letters 2004;32(2):167–73.
Pan Q-K, Tasgetiren MF, Liang Y-C. A discrete differential evolution algorithm for the permutation flowshop scheduling problem. In: Proceedings of the ninth annual conference on genetic and evolutionary computation. ACM New York, NY, USA; 2007. p. 126–33.
Pan Q-K, Tasgetiren MF, Liang Y–C. A discrete particle swarm optimisation algorithm for the no-wait flowshop scheduling problem. Computers and Operations Research 2008;35(9):2807–39.
Particle Swarm Optimisation Website, 〈http://www.particleswarm.info/〉.
Price K, Storn R, Lampinen J. Differential evolution—a practical approach to global optimization. Berlin: Springer; 2005.
RPR Alliance: a summary and overview of the IEEE 802.17 resilient packet ring standard; 2004.
Schrijver A, Seymour P, Winkler P. The ring loading problem. SIAM Journal of Discrete Mathematics 1998;11:1–14.
Storn R, Price K. Differential evolution— a simple and efficient adaptive scheme for global optimisation over continuous spaces. Technical report TR-95-012, ICSI; 1995.
Storn R, Price K. Differential evolution—a simple and efficient heuristic for global optimisation over continuous spaces. Journal of Global Optimisation, Kluwer Academic Publishers, Hingham 1997; 11:p. 341–59.
Tasgetiren MF, Suganthan PN, Pan Q-K. A discrete particle swarm optimisation algorithm for the generalised traveling salesman problem. In: Proceedings of the ninth annual conference on genetic and evolutionary computation. ACM New York, NY, USA; 2007. p. 158–67.
van Hoesel SPM. Optimization in telecommunication networks. Statistica Neerlandica 2005;59(2):180–205.
Wang BF. Linear time algorithms for the ring loading problem with demand splitting. Journal of Algorithms 2005;54(1):45–57.
Wang L, Wang X, Fu J, Zhen LA. Novel probability binary particle swarm optimization algorithm and its application. Journal of Software 2008;3(9):28–35.
Yuan J, Zhou S. Polynomial time solvability of the weighted ring arc-loading problem with integer splitting. Journal of Interconnection Networks 2004;5(2):193–200.
Yuan P, Gambiroza V, Knightly E. The IEEE 802.17 media access protocol for high-speed metropolitan-area resilient packet rings. IEEE Network 2004;18(3):8–15.