Evolutionary Computation Methods for the Schedule Optimization of Pipeline Networks

Jonas Krause

Department of Information and Computer Sciences,
University of Hawai'i at Manoa,
Honolulu, HI, USA
E-mail: krausej@hawaii.edu

Heitor Silvério Lopes *

Bioinformatics & Computational Intelligence Laboratory,
Federal University of Technology Paraná – UTFPR,
Curitiba, PR, Brazil
E-mail: hslopes@utfpr.edu.br
*Corresponding author

Abstract: Two evolutionary computation methods are presented in this paper, both variants of the Differential Evolution (DE) algorithm. Their main difference is the encoding process (binary and continuous) and both methods were successfully applied to the pipeline network schedule problem. A binary mathematical model is proposed to represent the flow of oil products in a 48 hours horizon period. In this paper, we introduce new benchmarks of the pipeline scheduling problem for testing the proposed evolutionary algorithms on a specific network topology, but with different products and demands. Although computationally expensive, a Mixed Integer Linear Programming (MILP) approach is used to obtain optimal solutions so as to compare results with the evolutionary methods. MILP results achieved optimal solutions for nine out of the fifteen benchmarks proposed, but it requires far more computational effort than the DE-variants. Even though it is a real-parameter algorithm, the DE can be consider as a good heuristic, which is an alternative for the discrete problem studied. The overall comparison of results between the proposed DE-variants and MILP supports the efficiency, robustness and convergence speed of DE algorithm suggesting its usefulness to real-world problems of limited complexity.

Keywords: Evolutionary Computation; Differential Evolution; MILP; Pipeline Network.

Reference to this paper should be made as follows: Krause, J. and Lopes, H. S. (2016) 'Evolutionary Computation Methods for the Schedule Optimization of Pipeline Networks', *International Journal of Innovative Computing and Applications*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Jonas Krause is graduated in Mathematics and has a MSc degree in Computer Engineering from the Federal University of Technology Paraná, Curitiba, Brazil, and currently he is a PhD student at the University of Hawaii at Manoa, USA. Heitor S. Lopes has a BSc, MSc and PhD in Electrical Engineering, and currently he is Titular Professor at Federal University of Technology Paraná, Curitiba, Brazil.

1 Introduction

One of the major problems of the oil industry is the distribution of petroleum products by polyducts. Transportation of crude oil to refineries and refined products to depots through pipelines comprise a distribution network. A precise schedule of each refined products (gasoline, kerosene, diesel, gas fuel, etc.) into the polyducts is crucial. Decision tools based on operations research can be used to determine the best sequence of batches to optimize the distribution. The operation of these networks seeks to satisfy all the constraints related to production, demand, storage and transportation time, leading to a high complexity schedule problem. A fast and efficient decision making system is required to maximize these networks and reduce the transportation costs. Bioinspired methods, such as Genetic Algorithms (Garcia et al., 2004) and Differential Evolution (Onwubolu and Davendra, 2006; Dong et al., 2012; Krause et al., 2015), have already been used in schedule problems to provide satisfactory and fast solutions with reasonable computation efforts.

Those heuristic methods seek sub-optimal or, eventually, optimal solutions in the search space of feasible solutions, managing computational operators of

local and global searches. Such algorithms are known as meta-heuristics and can be defined as an iterative search procedure inspired by biological mechanisms. The line of research that deals with these algorithms is known as Evolutionary Computation (EC) and covers the study of evolutionary strategies, genetic programming, evolutionary programming, genetic algorithms and other population algorithms such as differential evolution. The main advantage of EC is the possibility of finding good solutions to complex problems. Once the objectives and constraints are mathematically defined, EC algorithms use generic and flexible procedures to search for good quality solutions even without the guarantee of optimality. These procedures can be adapted to a wide range of problems, providing robustness and flexibility to the selected method. Therefore, the EC can be understood as a set of techniques that can be applied to solve complex problems for which other known techniques are impractical or difficult to apply.

The optimization problems are the ones that most benefited from the use of EC algorithms. Many engineering problems, for instance, may be modeled as optimization problems (Michalewicz and Fogel, 2004; Goldbarg et al., 2010). Such problems are often highly complex and cannot be solved in polynomial time. The application of meta-heuristics for solving these optimization problems presents a new paradigm, because it does not guarantee the optimal solution but achieves good feasible solutions with reasonable computational effort. An optimization problem widespread in the literature is the scheduling one and it can be modeled as a combinatorial problem (Andrade et al., 2015). Thus, this work focuses on optimizing the scheduling of oil products distribution network problem.

A pipeline network includes refineries, depots and distribution centers. The ducts connecting each one of these nodes compose the network structure. These ducts can be unidirectional or bidirectional, and they usually transport different types of products and so are called polyducts. Products are transported in batches, such a product is loaded into the pipeline pushing the previous product batched. These batches can be fractionated and frequently fill the entire duct. In these cases, the complexity of the network increases due the number of possible operations to be done in each node.

The optimization of the network consists in delivering products to different demand points with minimum operational costs (Magatão et al., 2004; Cafaro and Cerdá, 2004), and recent studies have been focused on the time delivery (Yongtu et al., 2012). Therefore, the objective is to deliver each product as fast as possible and the operational costs are represented as constraints. The proposed mathematical model uses a time window method and an objective function to be minimized. Demands and costs are handled as restrictions. The generality of the mathematical model allows it to be applied to different network structures with different products.

As mentioned before, mathematical programming and heuristic methods have been used to determine the best solution for such kind of problems. Depending on the network structure, the number of possible solutions is huge in such a way that browsing the search space is intractable with mathematical programming methods, since the processing time becomes unacceptable. In these cases, heuristic methods such as Differential Evolution can be an alternative to provide faster sub-optimal solutions. In this work, two variants of the DE are compared with MILP for specific pipeline network schedule optimization problems. This work is organized as follows: Section 2 describes the problem formally, including its complete mathematical formulation; Section 3 presents the two variants of Differential Evolution used to solve the benchmark problems; Section 4 presents some aspects of Linear Programming with focus on Mixed-Integer Linear Programming; Section 5 shows the proposed benchmarks and the solutions obtained by all methods; finally, Section 6 presents our final considerations about the methods and results and, also, points future research directions.

2 Problem Description

The discrete modeling of real-world problems using MILP is frequently found in the recent literature. However, the computational time to solve discrete models can be very high. According to Schrage (2000), the average processing time of linear programming is directly related to the number of variables and to the square of the number of constraints described in the mathematical model. This is one of the main motivations for using heuristic methods (such as those used here) for finding sub-optimal solutions. Evolutionary Computation methods can be an interesting alternative for solving such sort of optimization problems, since they have a lower computational cost (compared with MILP) for complex problems with many variables. Even if those methods do not guarantee the optimality of solutions, they are very useful to find reasonable good solutions for real-world problems in a relatively short processing time.

The pipeline schedule problem consists in distributing N products from R refineries through D depots to C distribution centers. Polyducts connect each one of these nodes and they are represented by the letter P. A simplified version of the network structure, introduced by Garcia et al. (2004), is presented in Fig. 1, and it has been the focus of other recent studies as well (Krause et al., 2015; de Souza et al., 2010).

The network model used in this work has nine polyducts represented by ten arrows (ducts). The bidirectional duct connecting the two depots is represented by two arrows, P_5 representing the flow from D_1 to D_2 and P_6 vice versa. During the schedule planning, these ducts are not allowed to be pumped at the same time, otherwise it would have a collision. The

same flow rate is considered for all products. To reduce product fragmentation and setup costs, each batch has to fill the entire duct up to be fully transported. Storage tanks on each node receive all types of products with aggregate tankage. The length of each duct is given by the number of time units needed for a batch to traverse it. Table 1 presents these time units ($P_{1,\ldots,10}$) corresponding to Fig. 1.

Products may be refined in different places. To reflect this feature, the model considers that R_1 refines products N_1, N_2 and N_3, and R_2 refines N_4, N_5 and N_6.

Generally, products are refined and transported to storage centers, and then to a final destination (end-customer, storage tanks ports or other companies). Real-world polyducts are, for the most part, bidirectional, usually connecting the storage tanks. In these polyducts the various products can flow from a storage tank to another in any direction, enabling the exchange of products between the them. Some networks use one-way pipelines, with a unique direction from refineries to storage tanks and tanks to the final destinations.

The use of a specific polyduct can be constrained due to specific pumping requirements or, eventually, maintenance. In this case, the whole pipeline network is affected and, as a consequence, the demand of products by the customers may not be satisfied. In such situations a re-scheduling of product batches to other ducts is immediately required. To match those constraints, the network management must compute, as soon as possible, new paths for each product. However, the real-world pipeline network cannot wait hours (or days) until the system computes the optimal solution, thus making evident the need of alternative methods. Given a fixed planning horizon of 48 hours, this scheduling problem consists of determining which product may be batched from the refineries, through the depots, and finally to the distribution centers, such that a given demand is satisfied.

2.1 Mathematical Model Formulation

The proposed mathematical model uses binary variables ($X_{n,p,k} = \{0,1\} \quad \forall n,p,k$) to represent the presence or not (1 or 0) of each product on each polyduct during a specific time period. Considering H the number of hours of the time horizon and N the number of products, the following indexes can be established: $n = \{1,\ldots,N\}$, $p = \{1,\ldots,P\}$ and $t = \{0,\ldots,H-1\}$. Index t represents each time unit of the horizon H. The index k depends on the length of each polyduct and the time units needed to traverse it. Equation 1 shows how k represents each time period.

$$k = P_p \times t, \qquad k < H \tag{1}$$

The objective function Z, shown in Equation 2, represents the minimization of the sum of the X binary variables.

$$Z = min\left(\sum_{n=1}^{N}\sum_{p=1}^{P}\sum_{k=0}^{H-1}(P_p \times (k+1)) \times X_{n,p,k}\right) \tag{2}$$

Each binary variable X is weighted by each polyduct time unit P_p and their position k on the schedule plan. Index k is increased by one to take into consideration the initial position.

The model constraints seek to represent the linear restrictions of a real network. The constraints are created to ensure the uniqueness of each product in each pipeline, no collision on the bidirectional polyduct, the batches order and supply/delivery demands, as follows:

- **Products uniqueness**: To ensure that only one product is transported by each polyduct in each period of time. Equation 3 represents the restrictions of products uniqueness in all p polyducts at all k periods of time.

$$\sum_{n=1}^{N} X_{n,p,k} \leq 1, \quad \forall p,k \tag{3}$$

- **Bidirectional duct**: To ensure that polyducts P_5 and P_6 are not used at the same time. Equation 4 shows the mathematical model for these restrictions.

$$\sum_{n=1}^{N}(X_{n,5,k} + X_{n,6,k}) \leq 1, \quad \forall k \tag{4}$$

- **Batches order**: These constraints ensure that the products arriving to distribution centers (C) were previously stored in depots (D) and earlier provided by refineries (R). For instance, polyduct P_7 that supplies client 1 depends on polyducts P_1 and P_5 to receive products 1, 2 and 3, and depends on polyducts P_3 and P_5 for products 4, 5 and 6. Similarly, polyducts P_8, P_9 and P_{10} have some dependence on previous polyducts in the network. Also, polyducts P_5 and P_6 have to respect the order of batches, since they depend on the products pumped through ducts P_1, P_2, P_3 and P_4. Therefore, Equation 5 presents the order constraints of the batches for polyducts P_7 and P_8, which receive products 1, 2 and 3 (n=$\{1, 2, 3\}$) from polyducts P_1 and P_6. Similarly, Equation 6 presents the order constraints of the batches for products 4, 5 and 6 (n=$\{4, 5, 6\}$) from polyducts P_3 and P_6.

$$\sum_{k=0}^{H-1}(P_7 \times X_{n,7,k+P_7} + P_8 \times X_{n,8,k+P_8}) \leq \sum_{k=0}^{H-1}(P_1 \times X_{n,1,k} + P_6 \times X_{n,6,k}), \quad \forall n \tag{5}$$

$$\sum_{k=0}^{H-1}(P_7 \times X_{n,7,k+P_7} + W_8 \times X_{n,8,k+W_8}) \leq \sum_{k=0}^{H-1}(P_3 \times X_{n,3,k} + P_6 \times X_{n,6,k}), \quad \forall n \quad (6)$$

Equation 7 presents the same order constraints for polyducts P_9 and P_{10}, which receive products 1, 2, and 3 (p={1, 2, 3}) from polyducts P_2 and P_5.

$$\sum_{k=0}^{H-1}(P_9 \times X_{n,9,k+P_7} + P_{10} \times X_{n,10,k+P_8}) \leq \sum_{k=0}^{H-1}(P_2 \times X_{n,2,k} + P_5 \times X_{n,5,k}), \quad \forall n \quad (7)$$

Similarly, Equation 8 do the same for products 4, 5 and 6 (p={4, 5, 6}) from polyducts P_4 and P_5.

$$\sum_{k=0}^{H-1}(P_9 \times X_{n,9,k+P_7} + P_{10} \times X_{n,10,k+P_8}) \leq \sum_{k=0}^{H-1}(P_4 \times X_{n,4,k} + P_5 \times X_{n,5,k}), \quad \forall n \quad (8)$$

The order of polyducts P_5 and P_6 is presented in Equations 9 and 10, respectively.

$$\sum_{k=0}^{H-1}(P_5 \times X_{n,5,k+P_5}) \leq \sum_{k=0}^{H-1}(P_1 \times (X_{1,1,k} + X_{2,1,k} + X_{3,1,k}) + P_3 \times (X_{4,3,k} + X_{5,3,k} + X_{6,3,k})), \quad \forall n \quad (9)$$

$$\sum_{k=0}^{H-1}(P_6 \times X_{n,6,k+P_5}) \leq \sum_{k=0}^{H-1}(P_2 \times (X_{1,2,k} + X_{2,2,k} + X_{3,2,k}) + P_4 \times (X_{4,4,k} + X_{5,4,k} + X_{6,4,k})), \quad \forall n \quad (10)$$

- **Supply and delivery demands**: These constraints ensure that the correct demand is delivered and the origin of all products is one of the refineries. The proposed model also allows that the total pumping from refineries is larger than the overall amount delivered. The extra products remain stored in the intermediary tanks. Therefore further demand can be satisfied faster in the next period of time. Equation 11 defines that the total amount of products supplied by the refinery is larger than or equal to the amount requested ($Q_{1,2,3}$). Equation 12 defines that the total amount of products delivered to final customers is equal to the amount requested.

$$\sum_{n=1}^{N}\sum_{k=0}^{H-1}(P_1 \times X_{n,1,k} + P_2 \times X_{n,2,k} + P_3 \times X_{n,3,k} + P_4 \times X_{n,4,k}) \geq Q_{1,2,3} \quad (11)$$

$$\sum_{n=1}^{N}\sum_{k=0}^{H-1}(P_7 \times X_{n,7,k} + P_8 \times X_{n,8,k} + P_9 \times X_{n,9,k} + P_{10} \times X_{n,10,k}) = Q_{1,2,3} \quad (12)$$

For customers 1 and 3, who are supplied by only one polyduct, Equations 13 and 14 represent the delivery constraints such that $Q_{1,2,3}$ is the amount of each product to be supplied to each customer.

$$\sum_{k=0}^{H-1}(P_7 \times X_{n,7,k}) = Q_1, \quad \forall n \quad (13)$$

$$\sum_{k=0}^{H-1}(P_{10} \times X_{n,10,k}) = Q_3, \quad \forall n \quad (14)$$

Customer C_2 is supplied, simultaneously, by polyducts P_8 and P_9. Hence, the amount Q_2 requested by this customer is the sum of the products carried by these polyducts, and Equation 15 represents such constraint.

$$\sum_{k=0}^{H-1}(P_8 \times X_{n,8,k} + P_9 \times X_{n,9,k}) = Q_2, \forall n \quad (15)$$

To represent the production constraints, the proposed model assumes that refinery R_1 produces only products 1, 2 and 3, and that refinery R_2 produces only products 4, 5 and 6. Consequently, variables $X_{1,3,k}$, $X_{1,4,k}$, $X_{2,3,k}$, $X_{2,4,k}$, $X_{3,3,k}$, $X_{3,4,k}$, $X_{4,1,k}$, $X_{4,2,k}$, $X_{5,1,k}$, $X_{5,2,k}$, $X_{6,1,k}$, and $X_{6,2,k}$ can be excluded from the model studied in this paper. Notwithstanding, the model allows other more complex production schemes.

The above-defined constraints, defined as linear equations, restrict the search space of the problem. The more restricted the search space, the more difficult is for a given method to traverse it and find good (or even, optimal) solutions. DE-variants and MILP are applied using this binary mathematical model to determine the best schedule for this pipeline network.

3 Differential Evolution (DE)

Evolutionary Computation (EC) methods propose alternative solutions to the purely mathematical methods. These methods have been frequently applied

to operations research problems, an interdisciplinary area of knowledge dedicated to the development of mathematical models and algorithms for solving real complex problems. However, the search for the optimal solution to real world problems (with thousands of variables and constraints) requires a huge computational effort. Therefore, a balance between accuracy of solutions and time to obtain them is necessary for most of those complex problems. This is where evolutionary algorithms can be more useful. Many EC methods and variants have been proposed along the last decades. Amongst them, Differential Evolution (DE) is one of the most widely used in recent years (Price et al., 2005).

The DE algorithm was first introduced by Storn and Price (1995) and devised for optimization in continuous spaces. It arose as a simple and efficient real-parameter algorithm for global optimization. DE is a stochastic population-based algorithm and, so, it evolves a population of possible solutions (individuals) for the optimization problem. Such individuals, encoded as real-valued vectors, undergo the action of selection, mutation and crossover operations, so that new individuals are constantly generated and improved. DE is similar to other Evolutionary Computation methods, but the strategies used here aim at mutating the individuals after vector operations. Different strategies were devised to modify individuals of DE, although the most successful and used one is the DE/rand/1/bin (Price et al., 2005). This strategy consists in randomly selecting an individual to be mutated, one difference vectors to perturb the selected individual and a binomial crossover. The mutation and crossover processes are applied in a trial population, each individual fitness is calculated and the new population is evaluated. The number of individuals of the population is maintained constant along iterations thanks to the selection procedure that substitute worse by better-fitted individuals. Along a large number of iterations, the population size is maintained but its overall quality is improved or, at least, does not deteriorate. Using continuous variables, DE searches the best quality individual evolved in a predetermined number of generations. The formulation of the DE/rand/1/bin strategy is presented in Equation 16:

$$v_{i,g} = x_{r1} + F \times [x_{r2} - x_{r3}] \tag{16}$$

This equation represents the trial vector v receiving the random vector x plus a random difference variation. The DE/**rand**/1/bin represents the random individual x to be mutated. Parameter F is the weighting factor used to control the amplification of the differential variation. Biologically, the term mutation means a sudden change in the genetic characteristics of a chromosome. In the context of DE, mutation can also be seen as a change or disorder in the subject using a random element. This algorithm selects a random vector using the Mutation Parameter (MP) to be the basis of the new individual called target vector. The vector obtained after the mutation process is known as a donor vector and finally after the donor recombination process or Crossover (CR) this new vector is called a trial vector (Price et al., 2005).

The DE/rand/1/bin strategy may vary depending on the individual selected. Elitism can be used to select the individual with the highest fitness value for the mutation process, and this strategy is known as DE/**best**/1/bin. The best individual can also be added on the differential variation, in this case the strategy is called DE/**rand-to-best**/1/bin. Other strategies consist in using two differential variations, represented by DE/rand/**2**/bin, DE/best/**2**/bin and DE/rand-to-best/**2**/bin. The last element is the crossover process and it can be set to DE/rand/1/**bin** and DE/rand/1/**exp**, representing the binomial and exponential crossovers, respectively. The exponential crossover process can also be selected in all described DE strategies, creating a list of ten possible strategies that can be used on DE algorithm (Adeyemo et al., 2010). Both proposed methods in this paper use the classical DE/rand/1/bin strategy.

3.1 Binary Differential Evolution (BDE)

This DE variant was presented in Krause et al. (2013) and it is adapted for binary spaces only. The adaptation of the original DE starts in the initialization of the population, such that random binary values are used to create individuals instead of random continuous values. The original mutation process of DE is replaced by a random bit inversion. This adaptation of the DE mutation process is inspired on the mutation process of the Genetic Algorithm (Goldberg, 1989).

Algorithm 1 presents the pseudocode of the BDE algorithm.

Algorithm 1: Binary Differential Evolution

```
Parameters: range, NP, MP, PR
Initial Population x_i (i = 1, ..., range)
Evaluate fitness f(x_i) of each individual
WHILE {not done} DO
  FOR {i = 1 TO NP}
    IF {rndreal(0,1) < PR OR j = j_rand}
      IF {rndreal(0,1) < MP}
        InvertBit (y_j)
      END IF
        Crossover (y_j)
    END IF
  END FOR
  Calculate fitness f(y)
  IF {f(y) > f(x_i)}
    x_i <== y
  END IF
  Evaluate x*
END WHILE
```

A new parameter (Perturbation Rate) is inserted to establish how many individuals of the population will undergo the mutation and crossover processes. This new

parameter also ensures that at least one individual will be mutated. The DE crossover process is kept as the original since all individuals will continue to be binary after it. Adapted for binary problems, this meta-heuristic is a general algorithmic structure that can be applied to several optimization problems.

The BDE algorithm starts by setting parameters. The first parameter is the individual dimension or *range*. NP is the number of individuals of the population. MP and PR are the mutation and perturbation rates, respectively. A random population $\vec{x_i}$ is created and the fitness of each individual, $f(\vec{x_i})$, is calculated. Each individual is randomly selected by the perturbation rate and it is submitted to the mutation and crossover processes. The new individual $\vec{y}$ has its fitness calculated $f(\vec{y})$. If the new fitness of the trial individual is better (higher for maximization and lower for minimization) than the previous individual fitness $f(\vec{x_i})$, the trial solution $\vec{y}$ will be part of the new population.

3.2 Discretized Differential Evolution (DDE)

This DE variant was was also presented in Krause et al. (2013) and it evolves the possible solutions in a continuous search space and, later, discretize them to the binary space after the mutation and crossover processes. The individual to be evolved by the DDE is a n dimensional vector and each dimension is populated with a random float number between -1 and 1. With a normalized entry for each individual of the initial population, the DDE proceeds with the DE/rand/1/bin strategy of mutation and crossover through the generations. The new population is a group of vectors of continuous variables and these individuals are discretized to get their fitness evaluated. This discretization method uses a sigmoid function that allows each dimension to evolve gradually and individually (Krause et al., 2015, 2013).

Let F be the float number in each i dimension, the solution vector is discretized using the result of the sigmoid function of F. If this result is greater than zero the i-th dimension is set to 1, otherwise, it is set to 0. This discretization process is represented by Equation 17.

$$X_i = \begin{cases} 1, \text{if } \frac{2}{1+\exp(-2.F_i)} - 1 > 0, \\ 0, \text{otherwise} \end{cases} \tag{17}$$

Using this strategy, the evolved dimensions do not jump from 0 to 1 in the binary. They evolve gradually around zero using weighted values to search for the best continuous combination. Although rather simple, this feature is the key to efficiently apply continuous algorithms to discrete problems.

Algorithm 2 presents the pseudocode of the DDE algorithm.

The DDE algorithm is initialized with the parameters NP, CR, F and *range*. The NP states for the total number of individuals in the population. The CR and F are the crossover and mutation rates, respectively. Parameter *range* is the dimension of each individual. An initial random population is created with NP individuals and their initial fitness is calculated. Through the number of generations previously set, a trial population is created using the mutation and crossover processes. This new population is discretized by the sigmoid function which assigns values 1 or 0, depending whether the continuous value for each dimension of the individual. The fitness of the trial and discretized population is calculated and if the trial individual fitness is greater than the previous one, the new individual is included to the new population.

This strategy is widely used when adapting continuous devised algorithms to binary problems. It can also be used to convert continuous values into integer and, consequently, apply the DE to discrete problems. Hence, this version of the DE may be called discretized and can be adapted to other combinatorial problems.

4 Mathematical Programming

The Mathematical Programming consists in formulating a real-world model, variables and procedures using mathematical symbols to represent their relations. Mathematical programming is frequently used in decision-making processes in large engineering systems. This technique allows the definition of inter-relationships between variables that would be difficult to find intuitively.

These mathematical models include three main elements (Bazaraa et al., 1990): decision variables and

Algorithm 2: Discretized Differential Evolution

```
Function f(x) = DE (range, NP, CR, F)
x ⇐ random (range, NP)
fit_x ⇐ f(x)
WHILE {not done} DO
  FOR {i = 1 to NP}
    v_{i,G+1} ⇐ mutation (x_{i,G}, F)
    u_{i,G+1} ⇐ crossover (x_{i,G}, v_{i,G+1}, CR)
  END FOR
  IF {sigmoid (u_{i,G+1}) > 0}
    u_{i,G+1} ⇐ 1
  ELSE
    u_{i,G+1} ⇐ 0
  END IF
  fit_u ⇐ f(u)
  FOR {i = 1 to NP}
    IF {fit_u(i) > fit_x(i)}
      x_{i,G+1} ⇐ u_{i,G+1}
    ELSE
      x_{i,G+1} ⇐ x_{i,G}
    END IF
  END FOR
END WHILE
```

parameters; restrictions; and the objective function. The decision variables are the unknown values to be determined by each method, while parameters refer to the fixed input data. The constraints are a set of equations or inequalities that limit the possible values of variables. Finally, the objective function is a mathematical function that evaluates a solution and expresses the intention to maximize or minimize the model output.

To solve this models, Dantzig (1963) devised a method based on repetition cycles. This iterative algorithm is called SIMPLEX and it has been used since then to find the optimal solution of linear programming models. SIMPLEX is regarded as one of the most significant advances in the mathematics of the twentieth century.

This method consists of a set of criteria for choice of basic solutions to improve the model performance. For that, the problem must submit an initial basic solution. Subsequent basic solutions are calculated with the exchange of base variables for not base ones, generating new solutions. The criteria for selection of vectors and therefore the variables that enter and leave for the formation of the new base constituting the center of the SIMPLEX method.

Mixed Integer Linear Programming (MILP) involves mathematical models that can provide continuous, integer and/or binary variables, all related by linear constraints. Binary variables usually represent decisions to be taken that mean yes or no, on or off, true or false, and allow the programmer to specify logical conditions of the model. Integer variables are used to represent quantities considered indivisible, as the number of vehicles or number of people.

5 Computational Experiments and Results

For the MILP implementations, the GLPSol 4.57 was used. This is a free solver of the GNU Linear Programming Kit (GLPK) for linear programming based on the SIMPLEX method. The DE-variants were based on DE 3.6 [1] implemented in ANSI C language. All experiments done in this work used a cluster with 40 processing cores each one with Intel(R) Core(TM) i7 (3.5GHz), 8GB RAM and Ubuntu Server 12.04 operational system.

The objective of testing the proposed mathematical model and the DE-variants led to the creation of benchmarks with different levels of complexity. The scenarios simulated vary according to the number of products and the demand of each final client and, so, the complexity of the problem. Table 3 presents the proposed benchmarks.

Consider $N = \{2, 3, 4, 5, 6\}$ the number of products transported, $R_1 = \{N_1, N_2, N_3\}$ and $R_2 = \{N_4, N_5, N_6\}$ the products refined in each refinery and C_1, C_2 and C_3 the demand of each product in each distribution center. MILP experiments were set to maximum time-window of 14 days to complete processing and find the optimal solution. Benchmarks J01, J02 and J03 used a time window of 5 hours to complete processing, benchmarks J04, J05 and J06 required 5 days and benchmarks J07, J08, J09 completed after 12 days. Benchmarks J10, J11, J12, J13, J14 and J15 did not complete the processing and presented only sub-optimal solutions. With the increasing number of refined products (N), the number of variables of the model increases proportionally. Consequently, the search space increases exponentially along with the necessary processing time.

The solutions can be represented in Gantt graphics, where each product (N) is bumped into a polyduct (P) for each time frame (t). As example, the optimal solutions for benchmarks J03 and J09 are presented on Figure 2 together with a sub-optimal solution for benchmark J15. In these solutions we can see the "holes" in the plot, representing that the polyduct is not being used at that time. These gaps are network truncations and they represent the waiting for the pumping of the product by the refineries/storage tanks.

In all solutions, the polyducts P_1, P_2, P_3 e P_4 were continuously used to deliver the demand of products as soon as possible. The polyducts P_5 e P_6, representing the bidirectional pipeline, were mostly used by benchmarks with five and six products. These polyducts help in diverting the products from storage tanks and it is needed to carry larger amounts at the same time horizon. Polyducts P_7, P_8, P_9 e P_{10} deliver the products to end customers and depend on the availability of each product in storage tanks. Consequently, these polyducts have time periods not being used because they must wait for the subsequent polyducts finished their batches.

The experiments using the proposed evolutionary algorithms (BDE and DDE) also presented some optimal and sub-otimal solutions. Each method use a different adaptation to work in binary spaces, resulting in different possible solutions to the problem. As every heuristic method, BDE and DDE start their individuals with random values and seek the evolution of the population. Therefore, the results are influenced by the number of individuals of the population, the number of generations and their control parameters.

The running parameters used in the DE-variants, such as number of individuals in the population and number of iterations, were mostly set according to the literature (Price et al., 2005). Mutation (MP), perturbation (PP) and crossover (CR) parameters were tested with a $\pm$ 5% variation. The basic values for these parameters were those used in Krause et al. (2015, 2013). Table 2 shows the parameters of the with which the best solutions were found.

After 100 runs, each individual of each DE-variants achieved a feasible solution. Tables 4 and 5 present all results (BDE and DDE respectively) and compares them with the optimal and sub-optimal solutions. Each table shows the best fitness solution achieved by each DE-variant (Best), the average and standard deviation of the

100 runs (Avg ± SD) and the percentage (%) achieved of the optimal or sub-optimal found by MILP.

Results in bold (J01, J02 and J03 for BDE and J01, J02, J03, J04 and J05 to DDE) indicate that the DE-variants achieved the optimal solution for these benchmarks. The processing time for BDE and DDE on J01, J02 and J03 was less than 2 hours. The DE-variants required different time windows to process the other benchmarks: BDE used 3 hours for J04, J05 and J06, 4 hours for J07, J08 and J09, 5 hours for the J10, J11 and J12 and 7 hours for the other three benchmarks (J13, J14, J15). The continuous encoding of DDE requires more processing time when compared with BDE. For benchmarks J04, J05 and J06, DDE took 4 hours to complete processing. Around 5 hours for J07, J08 and J09 and 8 hours for benchmarks J10, J11 and J12. For the last three benchmarks, DDE required almost 12 hours processing. The average time for both DE-variants are considerably lower than the time required for MILP processing.

These results also present the behavior of each evolutionary method. In all proposed benchmarks, DDE found better solutions when compared to BDE. Despite that, the average and standard deviation of the DDE solutions are greater than the average and standard deviation of BDE. Suggesting that DDE has a good global search. This fact is probably associated with the original process of DE mutation, this characteristic was maintained throughout the iterations of the DDE. However, both methods presented viable solutions to the proposed combinatorial problem.

Statistical hypothesis tests were executed to compare the results obtained by each method. The statistical analysis of the average results was done with box plots. Significant differences in most benchmarks tested were found. Fig. 3 presents the box plots from benchmarks J03, J09 and J15.

Benchmarks J01, J02, J03, J08, J09, J10, J11 , J13 and J15 presented a similar behavior. As presented in Fig. 3, intervals between the first and third quartiles do not overlap themselves, indicating that results are statistically significant by both DE-variants. Benchmarks J04, J05, J06, J07 and J12 presented an overlap on the quartiles ranges of their box plots. This fact indicates that the difference between these results may not be significant. For these cases, statistical normality and non-parametric tests were used to determine the significance of the results. The Shapiro-Wilk test was performed with 95% of confidence to determine whether the data followed a normal distribution or not. All the five tested benchmarks reject the test hypothesis since data deviate from the normal distribution. Consequently, the non-parametric Wilcoxon Signed-Rank test was used to compare these results. For J04, J05 and J12, there was no significant difference, meaning that both evolutionary methods found similar solutions. For J06 and J07, the statistical test showed a significant difference (98% and 99% respectively) between the results.

6 Conclusions

This work compared methods for binary optimization of a real-world scheduling problem. Optimization of oil distribution networks is essential to reduce the costs associated to the transportation and to the correct delivery of each product to end customers. The proposed binary mathematical model is flexible enough to allow the study of different network topologies, with different structures, products and time horizons. However, it leads to a complex combinatorial problem and requires a large computational effort to solve it, even for moderate instances of the problem. To illustrate the applicability of this model, the simplified network described in Section 2 is solved by MILP and two Differential Evolution variants, BDE and DDE. These two meta-heuristics were recently proposed in the literature and, basically, the main difference between them is how the individuals of the population are encoded for binary problems.

The proposed benchmarks present a set of rules and restrictions that simulate a real-world situation. A more detailed representation of real situations brings the possibility to create new case studies. The difficulty of representing all the constraints of the problem leads to simplifications in the benchmarks constrains. Real-world problems may require additional restrictions and changes in the mathematical model. Notwithstanding, is a fair approximation of a real problem and the proposed benchmarks can be very useful for testing the performance of exact and heuristic optimization algorithms.

Experiments sought to test the proposed binary model and reached the main goal of finding feasible solutions to the problem. MILP was successfully applied in this model and provide optimal solutions for benchmarks J01, J02, J03, J04, J05, J06, J07, J08 and J09. Sub-optimal solutions were obtained for benchmarks J10, J11, J12, J13, J14 and J15, after a predefined processing time (14 days).

Overall results found by the DE-variants can be considered very good. BDE achieved the optimal values for three instances and DDE for five of them. As expected, the performance of the meta-heuristics degraded as the size of the search space increased. However, considering the processing time needed by DE-variants when compared with MILP, they presented themselves as good alternative methods with low computational effort.

Results of the BDE and DDE also show the behavior of each algorithm. Average and standard deviation results of DDE suggest this algorithm is consistent and an effective method for global optimization. It is known that the regular Differential Evolution algorithm performs well for many problems due to its crossover and mutation processes, which use vector differences for the evolution of solutions. These processes, as well as continuous encoding were maintained in the DDE algorithm. These features are responsible for the diversity of the achieved results. When DDE is compared

to the BDE, it presents different (better and worse) solutions to the proposed problem. Numerical results obtained by DDE suggest that algorithms designed for continuous spaces can be efficiently applied to some discrete problems. The use of the sigmoid function in the discretization process allows the application of the DDE algorithm to other binary and integer combinatorial problems.

The high complexity of the scheduling optimization problem was one of the main motivations for this work. The new modeling presented here aims to adapt a real-world continuous problem to binary spaces. With a large number of variables and several constraints, the meta-heuristic algorithms showed a good balance between global and local search. This equilibrium is essential to achieve good feasible solutions with less computational effort. Although the application of DE-variants here can be considered successful, future work will focus on self-adaptation of parameters, that was proved to lead to better results than fixed-parameters' algorithms (Maruo et al., 2005).

Another important contribution of this work is the set of benchmark instances for the simplified pipeline network model. These benchmarks may be useful for other researchers to test optimization methods. The possibility of creating new and more complex benchmarks also provides a wide range of case studies.

ACKNOWLEDGMENTS

This work was partially supported by grants from the National Counsel of Technological and Scientific Development (CNPq) to J. Krause and H.S. Lopes.

References

Adeyemo, J., Bux, F., and Otieno, F. (2010). Differential evolution algorithm for crop planning: single and multi-objective optimization model. *International Journal of the Physical Sciences*, 5(10):1592–1599.

Andrade, M. R. Q., Ochi, L. S., and Martins, S. L. (2015). Heuristics for the periodic mobile piston pump unit routing problem. *Int. J. Nat. Comput. Res.*, 5(1):1–25.

Bazaraa, M., J., J., and Sherali, H. (1990). *Linear Programming and Network Flows*. J. Wiley & Sons, New York, USA.

Cafaro, D. C. and Cerdá, J. (2004). Optimal scheduling of multiproduct pipeline systems using a non-discrete milp formulation. *Computers & Chemical Engineering*, 28(10):2053–2068.

Dantzig, G. (1963). *Linear Programming and Extensions*. Princeton University Press, Princeton, USA.

de Souza, T. C. N., Goldbarg, E. F. G., and Goldbarg, M. C. (2010). Transgenetic algorithm for the biobjective oil derivatives distribution problem. In *IEEE Congress on Evolutionary Computation*, pages 1–8, Piscataway, USA. IEEE Press.

Dong, X.-l., Liu, S.-q., Tao, T., Li, S.-p., and Xin, K.-l. (2012). A comparative study of differential evolution and genetic algorithms for optimizing the design of water distribution systems. *Journal of Zhejiang University – Science A*, 13(9):674–686.

Garcia, J. M. C., Martin, J. L. R., Gonzales, A. H., and Blanco, P. F. (2004). Hybrid heuristic and mathematical programming in oil pipelines networks. In *Proc. Congress on Evolutionary Computation*, volume 2, pages 1479–1486, Piscataway, NJ, USA. IEEE Press.

Goldbarg, M. C., Goldbarg, E. F. G., and Duarte, H. D. M. (2010). Transgenetic algorithm for the periodic mobile piston pump unit routing problem with continuous oil replenishment. *Int. J. Innov. Comput. Appl.*, 2(4):203–214.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Boston, MA, USA.

Krause, J., Parpinelli, R. S., and Lopes, H. S. (2013). A comparison of differential evolution algorithm with binary and continuous encoding for the MKP. In *Proc. of BRICS Congress on Computational Intelligence and* 11^{th} *Brazilian Congress on Computational Intelligence*, pages 381–387, Piscataway, NJ, USA. IEEE Press.

Krause, J., Sieczka Jr., E. L., and Lopes, H. S. (2015). Differential evolution variants and MILP for the pipeline network schedule optimization problem. In *Proc. of* 2^{nd} *Latin-American Congress on Computational Intelligence*, pages 1–6, Piscataway, NJ, USA. IEEE Press.

Magatão, L., Arruda, L. V. R., and Neves Jr., F. (2004). A mixed integer programming approach for scheduling commodities in a pipeline. *Computers & Chemical Engineering*, 28(1-2):171–185.

Maruo, M. H., Lopes, H. S., and Delgado, M. R. B. S. (2005). Self-adapting evolutionary parameters: encoding aspects for combinatorial optimization problems. In Raidl, G. R. and Gottlieb, J., editors, *Proc.* 5^{th} *European Conference on Evolutionary Computation in Combinatorial Optimization*, volume 3448 of *Lecture Notes in Computer Science*, pages 154 – 165. Springer-Verlag, Heidelberg, Germany.

Michalewicz, Z. and Fogel, D. B. (2004). *How to Solve It: Modern Heuristics*. Springer, Berlin, Germany, 2nd edition.

Onwubolu, G. and Davendra, D. (2006). Scheduling flow shops using differential evolution algorithm. *European Journal of Operational Research*, 171(2):674–692.

Price, K., Storn, R., and Lampinen, J. (2005). *Differential Evolution: a practical approach to global optimization*. Springer-Verlag, Heidelberg, Germany.

Schrage, L. (2000). *Optimization Modeling with LINGO*. Lindo Publishing, Chicago, USA.

Storn, R. and Price, K. (1995). Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, International Computer Science Institute, Berkeley University, Berkeley, CA, USA.

Yongtu, L., Ming, L., and Ni, Z. (2012). A study on optimizing delivering scheduling for a multiproduct pipeline. *Computers & Chemical Engineering*, 44(9):127–140.

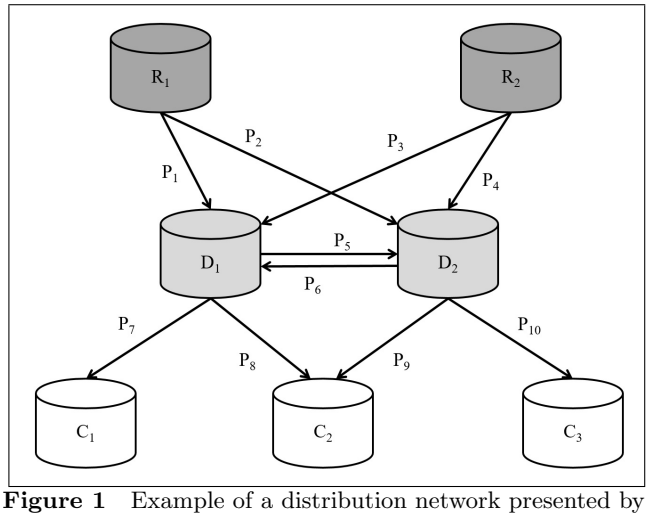

Figure 1 Example of a distribution network presented by Garcia et al. (2004).

Table 1 Units of time needed for one batch to traverse each duct.

Ducts	Time
P_1	1
P_4, P_8, P_{10}	2
P_2, P_3, P_5, P_6, P_9	3
P_7	4

Table 2 BDE and DDE parameters

Parameter	BDE	DDE
Population	300	300
Generations	20,000	20,000
MP	15%	10%
PR	50%	–
CR	–	80%

Table 3 Proposed benchmarks. Solutions with "*" are sub-optimal.

Bench	N	R_1	R_2	C_1	C_2	C_3	Solution
J01	2	1	4	12	12	12	658
J02	2	1	4	12	24	12	1100
J03	2	1	4	12	24	18	1450
J04	3	1,2	4	8	8	8	803
J05	3	1,2	4	12	12	12	1654
J06	3	1,2	4	12	18	12	2288
J07	4	1,2	4,5	8	8	8	1207
J08	4	1,2	4,5	8	16	8	1968
J09	4	1,2	4,5	12	18	10	3170
J10	5	1,2,3	4,5	8	8	6	1617 *
J11	5	1,2,3	4,5	8	12	6	2179 *
J12	5	1,2,3	4,5	8	12	8	2860 *
J13	6	1,2,3	4,5,6	4	6	4	1037 *
J14	6	1,2,3	4,5,6	4	8	4	1260 *
J15	6	1,2,3	4,5,6	4	12	6	2393 *

Table 4 Results obtained by **BDE** algorithm.

Bench	Best	Avg ± SD	%
J01	**658**	664.40 ± 11.04	**0.00%**
J02	**1100**	1128.33 ± 14.98	**0.00%**
J03	**1450**	1458.55 ± 24.58	**0.00%**
J04	857	866.30 ± 15.03	6.72%
J05	1747	1752.68 ± 17.89	5.62%
J06	2446	2460.84 ± 24.59	6.91%
J07	1345	1370.03 ± 33.49	11.43%
J08	2177	2213.92 ± 37.98	10.61%
J09	3517	3564.02 ± 73.66	10.94%
J10	1770	1819.23 ± 30.77	9.46%
J11	2401	2506.54 ± 59.06	10.18%
J12	3344	3481.07 ± 80.27	16.92%
J13	1210	1244.26 ± 48.68	16.68%
J14	1493	1688.22 ± 61.30	18.49%
J15	3153	3240.42 ± 94.29	31.75%

Table 5 Results obtained by **DDE** algorithm.

Bench	Best	Avg ± SD	%
J01	**658**	720.20 ± 36.59	**0.00%**
J02	**1100**	1198.55 ± 46.74	**0.00%**
J03	**1450**	1541.90 ± 57.43	**0.00%**
J04	**803**	867.62 ± 46.32	**0.00%**
J05	**1654**	1739.40 ± 51.56	**0.00%**
J06	2358	2479.30 ± 76.73	3.06%
J07	1303	1396.79 ± 59.88	7.95%
J08	2136	2331.53 ± 74.49	8.53%
J09	3438	3625.86 ± 98.89	8.45%
J10	1715	1928.06 ± 82.30	6.06%
J11	2392	2673.06 ± 98.31	9.77%
J12	3307	3485.51 ± 107.32	15.62%
J13	1201	1292.96 ± 94.40	15.81%
J14	1479	1734.32 ± 101.79	17.38%
J15	3000	3372.58 ± 138.35	25.36%

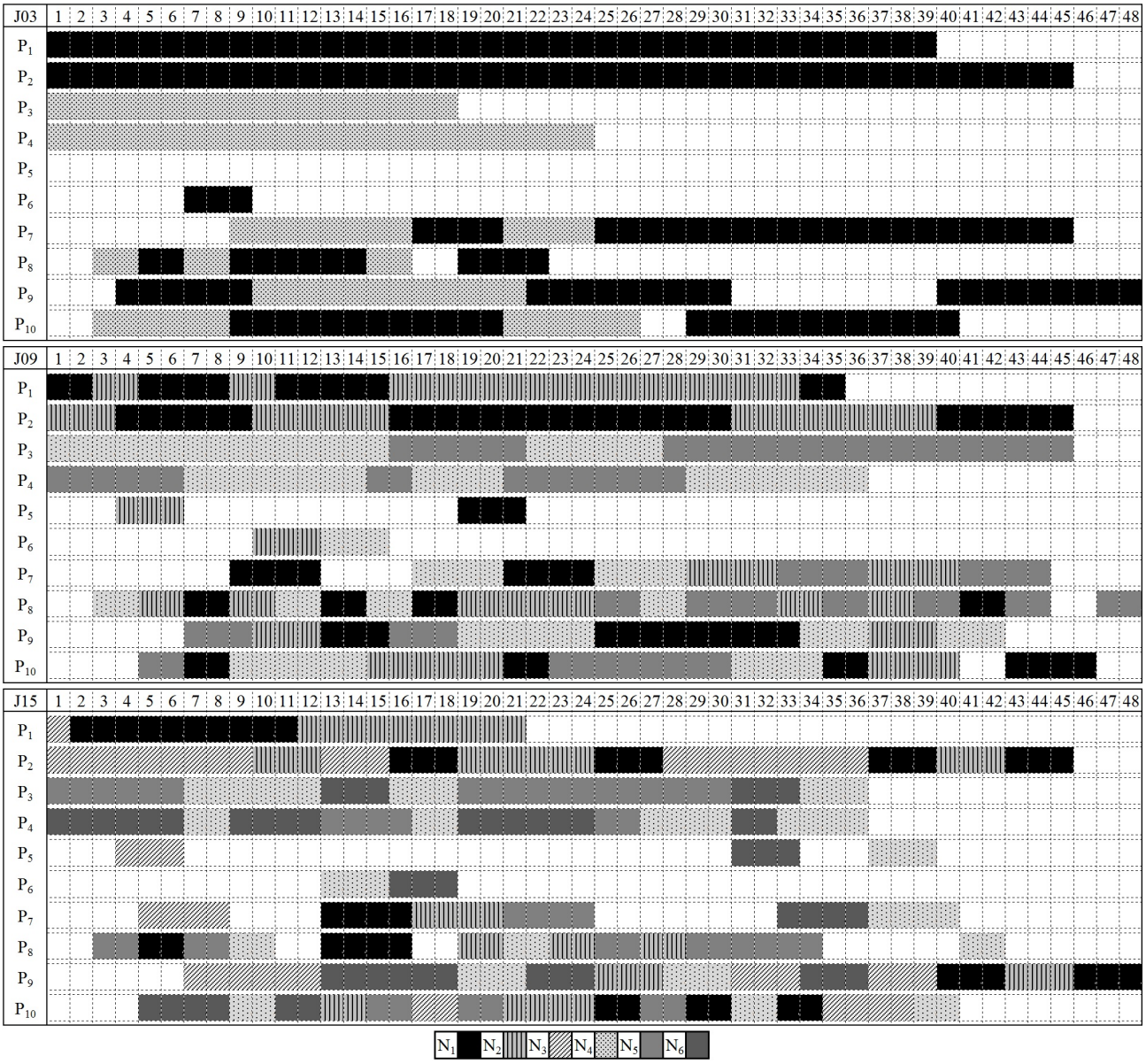

Figure 2 Graphic solutions for benchmarks J03, J09 (optimal) and J15 (sub-optimal)

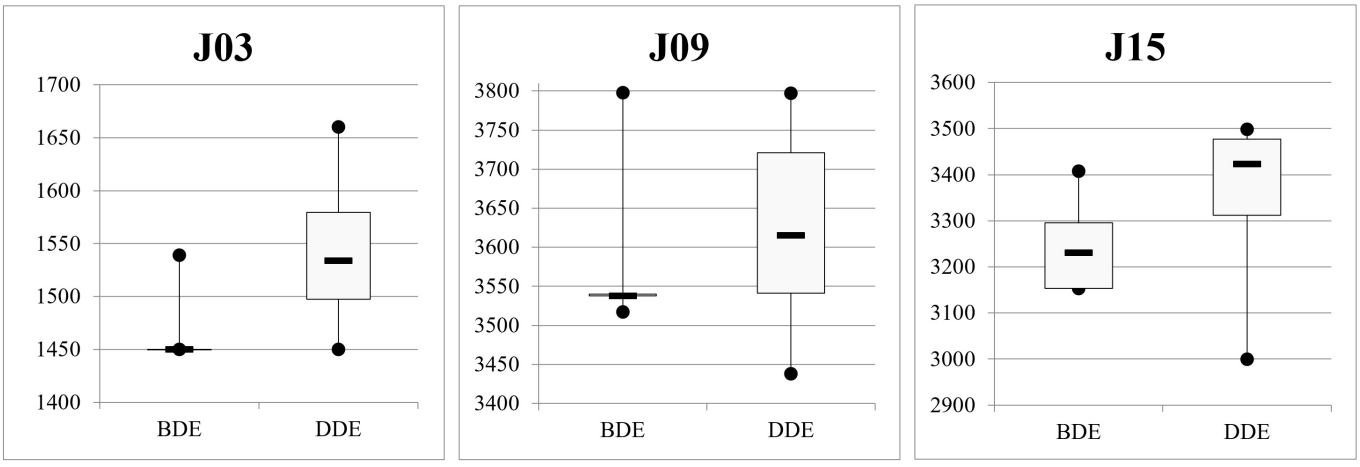

Figure 3 Boxplot of BDE and DDE results for J03, J09 and J15 benchmarks.