Differential Evolution Variants and MILP for the Pipeline Network Schedule Optimization Problem

Jonas Krause, Edson Luiz Sieczka Jr., Heitor Silvério Lopes
Federal University of Technology Paraná – UTFPR
Curitiba, Paraná, Brazil
jonaskrause@hotmail.com, sieczka@gmail.com, hslopes@utfpr.edu.br

***Abstract*—This paper presents two variants of the Differential Evolution (DE) algorithm, with binary and continuous encoding, for the pipeline network schedule problem. A binary mathematical model is proposed to represent the flow of oil products in a 48 hours horizon period. Although computationally expensive, a Mixed Integer Linear Programming (MILP) approach was also used to obtain optimal solutions so as to compare results with the other methods. In this paper, we introduced a benchmark of scheduling problems for testing the algorithms with a fixed network topology, but with different number of products and demands by final clients. MILP results determined optimal solutions for six of the proposed benchmarks, but it required far more computational effort than the DE-variants. Even though it is a real-parameter algorithm, the DE presented itself as a good heuristic alternative for the discrete problem approached here. The overall comparison of results between the proposed DE-variants and MILP supports the efficiency, robustness and convergence speed of DE algorithm.**

I. Introduction

One of the major problems of the petroleum industry is the distribution of oil products by polyducts. Transportation of crude oil to refineries and refined products to depots through pipelines comprise a distribution network. The schedule of each of the 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 operations of these networks are highly complex to satisfy all the constraints related to production, demand, storage and transportation time. A fast and efficient decision making system is required to maximize these networks and reduce the transportation costs. Heuristic methods, such as Genetic Algorithm [1], [2] and Differential Evolution [3], [4], have been used in schedule problems to provide satisfactory and fast solutions with reasonable computation efforts.

A pipeline network includes a number of refineries, depots and distribution centers. The ducts connecting each one of these nodes compose the network structure. These ducts usually transport different products along time and can be bidirectional. Products are transported in batches, i.e., 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 on each node.

The optimization of the network consists in delivering products to different demand points with minimum operational costs [5], [6], and recent studies have been focused on the time delivery [7]. In this case, the objective is to deliver each product as fast as possible and the operational costs are represented as constraints. The proposed mathematical model follows uses a time window method and an objective function is minimized. Demand and costs are handled as constraints. Therefore, this model may be applied to different network structures with several products.

Mathematical programming and heuristic methods have been used to determine the best solution for such sort of scheduling problems. Depending on the network structure, the number of possible solutions is intractable with mathematical programming methods, since the processing time becomes unacceptable. In these cases, heuristic methods, such as Differential Evolution, can be successfully used as alternative method to provide faster solutions, but without guaranteeing optimality.

In this work, two variants of the DE are compared with MILP for specific pipeline network schedule optimization problems. The DE-variants results support the efficiency of this real-parameter algorithm in continuous search spaces. The binary encoded DE also presented satisfactory results requiring less time processing. MILP results determined optimal solutions for six of the proposed benchmarks, but it required more computational effort for the remaining ones. Even though it is a real-parameter algorithm, the DE presented itself as a good heuristic alternative for discrete problems.

II. Problem Description

The 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 [2], is presented in Fig. 1, and it has been the focus of other recent studies [1], [8].

The network 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-verse. During the schedule planning these ducts are not allowed to be used 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 I presents these time units ($P_{1,\ldots,10}$) corresponding to Fig. 1.

978-1-4673-8418-6/15/$31.00 ©2015 IEEE

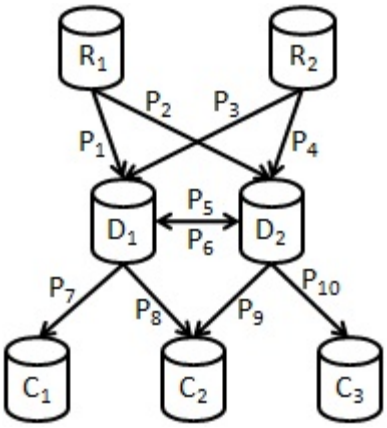

Fig. 1. Example of a distribution network [2].

TABLE I. UNITS OF TIME NEEDED FOR ONE BATCH TRAVERSE EACH DUCT [8].

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

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

A. Mathematical Model Formulation

The proposed mathematical model uses X binary variables to represent the presence or not (1 or 0) of each product, on each duct during a specific time period. Equation 1 represents these variables.

$$X_{n,p,k} = \begin{cases} 1 \\ 0 \end{cases} \qquad \forall n, p, k \tag{1}$$

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 k represents the time period and depends on the time units needed to traverse each polyduct. Equation 2 shows how k represents each time period.

$$k = P_p \times t \qquad k < H \tag{2}$$

The objective function Z, shown in Equation 3, represents the minimization of the sum of the X binary variables.

$$Z = min\left(\sum_{n=1}^{N}\sum_{p=1}^{P}\sum_{k=0}^{H-1}(P_p \times (k+1)) \times X_{n,p,k}\right) \tag{3}$$

Each binary variable X is weighted by each polyduct time units 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 duct in each period of time.
- **Bidirectional duct**: To ensure that ducts P_5 and P_6 are not used at the same time.
- **Batches order**: To ensure that the products arriving on distribution centers (C) were stored in depots (D) and provided by refineries (R).
- **Supply and delivery demands**: To ensure that the correct demand is delivered and the origin of all products is one of the refineries.

These constraints, defined as linear equations, restrict the search space of the problem. DE-variants and MILP are applied using these binary mathematical model to determine the best schedule for this pipeline network.

III. DIFFERENTIAL EVOLUTION (DE)

The DE algorithm was introduced by [9] 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 has different strategies. The most used and successful strategy is the DE/rand/1/bin [10]. 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. 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 by Equation 4:

$$v(g,i,j) = x(g,r_3,j) + F \times [x(g,r_1,j) - x(g,r_2,j)] \tag{4}$$

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. This 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 [11]. Both proposed methods in this paper use the classical DE/rand/1/bin strategy.

A. Binary Differential Evolution (BDE)

This DE variant was proposed by [12] 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 [13]. 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. Algorithm 1 presents the pseudocode of the BDE algorithm.

Algorithm 1 Binary Differential Evolution

```
Parameters: range, NP, PM, 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) < PM}
        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
```

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. PM 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.

B. Discretized Differential Evolution (DDE)

This DE-variant was proposed by [14] 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 [15].

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 5.

$$X_i = \begin{cases} 1, \text{if } \frac{2}{1+\exp(-2.F_i)} - 1 > 0, \\ 0, \text{otherwise} \end{cases} \tag{5}$$

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. This feature may be the key to efficiently apply continuous algorithms to discrete problems.

Algorithm 2 presents the pseudocode of the DDE algorithm.

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
```

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.

IV. Mixed Integer Linear Programming (MILP)

The Mathematical Programming consists in formulating a real-world model, variables and procedures using mathematical symbols to represent their relations. These models included three main elements [16]: decision variables and 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.

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 [17]. MILP involves 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.

V. Computational Experiments and Results

For the MILP implementations, the LPSolve 5.5.2.0 was used. This is a free solver for linear programming based on SIMPLEX and Branch-and-Bound methods. The DE-variants were based on DE 3.6 [1] implemented in ANSI C language. Tests used a cluster with 40 processing cores 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 simulated scenarios vary according to the number of products and the demand of each final client. Table II presents the proposed benchmarks.

TABLE II. Proposed benchmarks

Instance	N	R_1	R_2	C_1	C_2	C_3	Optimal solution
J01	2	1	3	12	12	12	658
J02	2	1	3	12	24	12	1100
J03	2	1	3	12	24	18	1450
J04	3	1,2	3	8	8	8	803
J05	3	1,2	3	12	12	12	1654
J06	3	1,2	3	12	18	12	2288
J07*	4	1,2	3,4	8	8	8	1245
J08*	4	1,2	3,4	8	16	8	2024
J09*	4	1,2	3,4	12	18	10	3206

Consider $N = \{2, 3, 4\}$ the number of products transported, $R_1 = \{1, 2\}$ and $R_2 = \{3, 4\}$ 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 used a time window of 7 days to complete processing and find the optimal solution. Benchmarks J01, J02 and J03 used a time window of 5 hours to complete processing, and benchmarks J04, J05 and J06 required 6 days. Benchmarks J07, J08 and J09 did not complete the processing and presented only sub-optimal solutions.

Fig. 2 presents the optimal graphic solution for benchmark J03 (products 1 and 4) and the sub-optimal solution achieved for benchmark J08 (products 1, 2, 3 and 4).

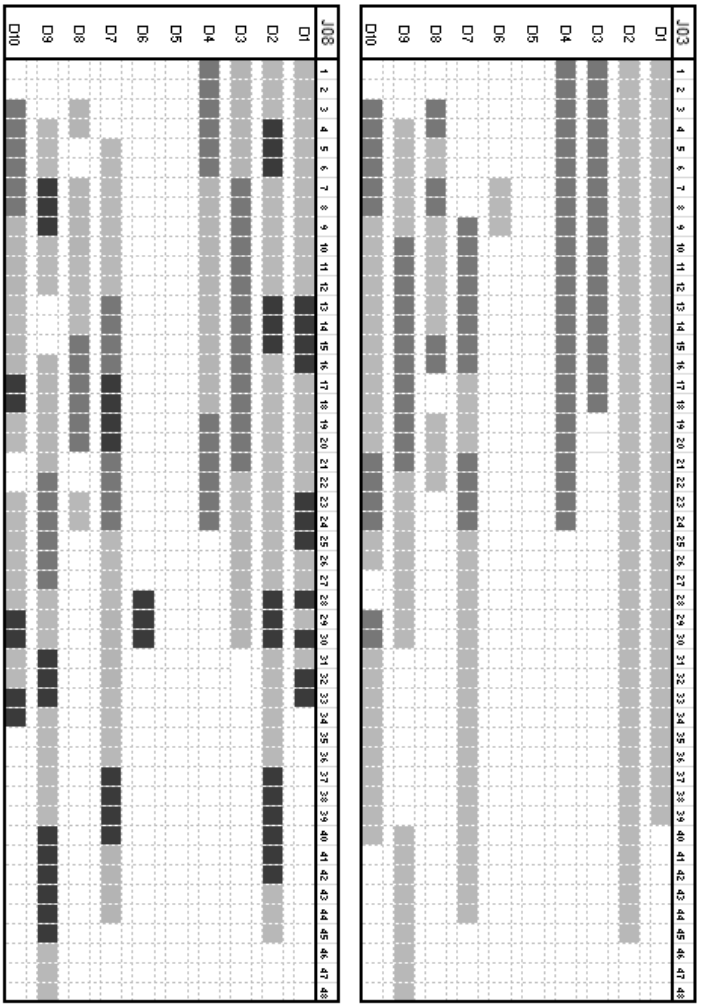

Fig. 2. Graphic solutions for benchmarks J03 (optimal) and J08 (sub-optimal)

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 [18]. Mutation (MP), perturbation (PP) and crossover (CR) parameters were tested with a $\pm$ 5% variation. The base value for these parameters were presented in [12] and [14]. Table III shows the parameters of the with which the best solutions were found.

After 100 runs, each result of DE-variants achieved a feasible solution. Table IV shows all results and compares them with the optimal and sub-optimal solutions. It presents the best fitness solution achieved by each DE-variant (Best), the average and standard deviation of the 100 runs (Avg $\pm$ SD)

[1] http://http.icsi.berkeley.edu/~ storn/

TABLE III. BDE AND DDE PARAMETERS

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

and the achieved percentage (%) of the optimal or sub-optimal found by MILP.

TABLE IV. RESULTS OBTAINED BY BDE AND DDE ALGORITHMS.

	BDE			DDE		
Bench	Best	Avg ± SD	%	Best	Avg ± SD	%
J01	**658**	664.40 ± 11.04	**0.00%**	**658**	720.20 ± 36.59	**0.00%**
J02	**1100**	1128.33 ± 14.98	**0.00%**	**1100**	1198.55 ± 46.74	**0.00%**
J03	**1450**	1458.55 ± 24.58	**0.00%**	**1450**	1541.90 ± 57.43	**0.00%**
J04	857	866.30 ± 15.03	6.72%	**803**	867.62 ± 46.32	**0.00%**
J05	1747	1752.68 ± 17.89	5.62%	**1654**	1739.40 ± 51.56	**0.00%**
J06	2446	2460.84 ± 24.59	6.91%	2358	2479.30 ± 76.73	3.06%
J07	1345	1370.03 ± 33.49	8.03%	1303	1396.79 ± 59.88	4.66%
J08	2177	2213.92 ± 37.98	7.56%	2136	2331.53 ± 74.49	5.53%
J09	3517	3564.02 ± 73.66	9.70%	3438	3625.86 ± 98.89	7.24%

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, for J01, J02 and J03, was less than 1 hour. The DE-variants required different time windows to process the other benchmarks: BDE used 5 hours for J04, J05 and J06, and 12 hours for J07, J08 and J09. DDE required 6 hours for J04, J05 and J06, and 16 hours for J07, J08 and J09. The average time for both DE-variants are considerably lower than the time required for MILP processing.

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 and J08.

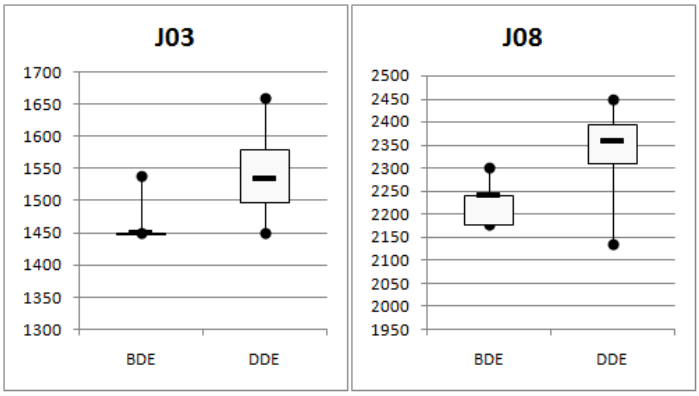

Fig. 3. Box plot of BDE and DDE results for J03 and J08 benchmarks.

Benchmarks J01, J02, 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, but with similar results. For J06 and J07, the statistical test showed a significant difference (98% and 99% respectively) between the results.

VI. CONCLUSIONS

This article compared three methods for binary optimization of a real-world problem. The binary model allows 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.

Optimization of oil distribution networks is essential to reduce the cost linked to the transportation and to the correct delivery of each product to end customers. To illustrate the applicability of the proposed model, the simplified network described in Section II is solved by MILP and two Differential Evolution variants, BDE and DDE. These meta-heuristics were recently proposed in the literature and, basically, the main difference between them is how solutions 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. Real world problems may require the addition of new restrictions and changes in the mathematical model. Notwithstanding, the proposed benchmarks can be very useful for testing the performance of exact of 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 and sub-optimal solutions. Only for benchmarks J01, J02, J03, J04, J05 and J06 the optimal values were found. For the other benchmarks (J07 ,J08 and J09) sub-optimal values were achieved. Overall results found by the DE-variants can be considered very good. EDB achieved the optimal values for three instances and EDD for five ones. As expected, the performance of the meta-heuristic methods degraded as the complexity of the search space increased. However, considering the processing time needed by DE-variants when compared with MILP, they are 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 as an effective method for global optimization. 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 is one of the main motivations for this work. The new modeling presented here aims at adapting this real world problem to binary spaces. With a large number of variables and several constraints, heuristic algorithms require a good balance between its 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 [19].

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

As future work, the optimal solutions of benchmarks J07, J08 and J09 can be achieved with more processing effort. The mathematical model presented here also allows the creation of new benchmarks with different time horizons, product types and delivered quantities. The variation of the number of refineries, storage tanks and final customers suggests further case studies with different network topologies. Other heuristic algorithms such as Genetic Algorithm, Particle Swarm Optimization, Artificial Bee Colony and hybrid methods can be implemented using the same proposed model. The usage of DDE to solve other discrete problems is also a future goal. Other versions of this method, with different discretization process (see [15]), can be applied to a vast range of problems.

ACKNOWLEDGMENT

The authors would like to thank Dr. Rafael S. Parpinelli (UDESC) for his contributions and the National Council for the Improvement of Higher Education (CAPES) for the scholarship to J. Krause. This work was also partially supported by a research grant from the CNPq to H.S. Lopes.

REFERENCES

[1] T. C. N. de Souza, E. F. G. Goldbarg, and M. C. Goldbarg, "Transgenetic algorithm for the biobjective oil derivatives distribution problem," in *IEEE Congress on Evolutionary Computation*. Piscataway, USA: IEEE Press, 2010, pp. 1–8.

[2] J. M. C. Garcia, J. L. R. Martin, A. H. Gonzales, and P. F. Blanco, "Hybrid heuristic and mathematical programming in oil pipelines networks," in *Proc. Congress on Evolutionary Computation*, vol. 2. Piscataway, NJ, USA: IEEE Press, 2004, pp. 1479–1486.

[3] G. Onwubolu and D. Davendra, "Scheduling flow shops using differential evolution algorithm," *European Journal of Operational Research*, vol. 171, no. 2, pp. 674–692, 2006.

[4] "A comparative study of differential evolution and genetic algorithms for optimizing the design of water distribution systems," *Journal of Zhejiang University - SCIENCE A*, vol. 13, no. 9, 2012.

[5] L. Magatão, L. V. R. Arruda, and F. Neves Jr., "A mixed integer programming approach for scheduling commodities in a pipeline," *Computers & Chemical Engineering*, vol. 28, no. 1-2, pp. 171–185, 2004.

[6] D. C. Cafaro and J. Cerdá, "Optimal scheduling of multiproduct pipeline systems using a non-discrete MILP formulation," *Computers & Chemical Engineering*, vol. 28, no. 10, pp. 2053–2068, 2004.

[7] L. Yongtu, L. Ming, and Z. Ni, "A study on optimizing delivering scheduling for a multiproduct pipeline," *Computers & Chemical Engineering*, vol. 44, no. 9, pp. 127–140, 2012.

[8] H. Westphal, F. Neves Jr., and L. V. R. de Arruda, "Algoritmo micro-genético aplicado ao scheduling de uma rede de distribuição de derivados de petróleo," in *Computação Evolucionária em Problemas de Engenharia*, H. S. Lopes and R. H. C. Takahashi, Eds. Curitiba (PR): Omnipax, 2011, pp. 331–354, (in portuguese).

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

[10] P. G. C. B. V. Babu and J. H. S. Mubeen, "Multiobjective differential evolution (MODE) for optimization of adiabatic styrene reactor," *Chemical Engineering Science*, vol. 60, no. 17, pp. 4822–4837, 2005.

[11] J. Adeyemo, F. Bux, and F. Otieno, "Differential evolution algorithm for crop planning: Single and multi-objective optimization model," *International Journal of the Physical Sciences*, vol. 5, no. 10, pp. 1592–1599, 2010.

[12] J. Krause and H. S. Lopes, "Proposta de um algoritmo inspirado em evolução diferencial aplicado ao problema multidimensional da mochila," in *Anais do Encontro Nacional de Inteligência Artificial*. Curitiba, PR: SBC, oct 2012, (in portuguese).

[13] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA, USA: Addison-Wesley, 1989.

[14] J. Krause, R. S. Parpinelli, and H. S. Lopes, "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*. Piscataway, NJ, USA: IEEE Press, 2013, pp. 381–387.

[15] J. Krause, J. A. Cordeiro, R. S. Parpinelli, and H. S. Lopes, "A survey of swarm algorithms applied to discrete optimization problems," in *Swarm Intelligence and Bio-Inspired Computation: Theory and Applications*, X.-S. Yang, Z. Cui, R. Xiao, A. H. Gandomi, and M. Karamanoglu, Eds. Amsterdam, The Netherlands: Elsevier Science, 2013, pp. 169–192.

[16] M. Bazaraa, J. J., and H. Sherali, *Linear Programming and Network Flows*. New York, USA: J. Wiley & Sons, 1990.

[17] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Mineola, NY, USA: Dover Books, 1998.

[18] K. Price, R. Storn, and J. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, ser. Natural Computing. Heidelberg, Germany: Springer, 2005.

[19] M. H. Maruo, H. S. Lopes, and M. R. B. S. Delgado, "Self-adapting evolutionary parameters: encoding aspects for combinatorial optimization problems," in *Proc.* 5^{th} *European Conference on Evolutionary Computation in Combinatorial Optimization*, ser. Lecture Notes in Computer Science, G. R. Raidl and J. Gottlieb, Eds. Heidelberg, Germany: Springer-Verlag, 2005, vol. 3448, pp. 154 – 165.