2009 Ninth International Conference on Intelligent Systems Design and Applications

The Bi-objective Problem of Distribution of Oil Products by Pipeline Networks Approached by a Particle Swarm Optimization Algorithm

Thatiana C. N. de Souza
UFRN, Brazil
thatinsouza@gmail.com

Elizabeth F. G. Goldbarg
UFRN, Brazil
beth@dimap.ufrn.br

Marco C. Goldbarg
UFRN, Brazil
gold@dimap.ufrn.br

Abstract

The distribution of petroleum products through pipeline networks is an important problem that arises in production planning of refineries. It consists in determining what will be done in each production stage given a time horizon, concerning the distribution of products from source nodes to demand nodes, passing through intermediate nodes. Constraints concerning storage limits, delivering time, sources availability, among others, have to be satisfied. This problem can be viewed as a bi-objective problem that aims at minimizing time and the successive transmission of different products in the same pipe. In this paper, a discrete Particle Swarm Optimization algorithm is applied to this problem. The results obtained with the proposed approach are compared with the results obtained by two Genetic Algorithms proposed previously for the problem.

1. Introduction

Usually, the main purpose of production planning is to coordinate operations such that profitability is maximized or costs are minimized. The production in refineries is a dynamic process and often new scenarios can impact the programming of production activities. A number of factors have to be considered in the production planning of refineries such as changes in the demand of products, product specifications, delivery dates, quality and quantity of raw materials, availability and performance of the processing units, among others. Integrated strategies to improve planning in order to program the production activities need to be considered. In this context, a distribution network is composed of oil refineries, terminals and parks for storage that are interconnected by polyducts. A polyduct is a pipeline that transports different types of oil products. If the polyduct is long enough then different types of products can occupy different parts of it during transportation. In order to address the problem under a discrete optimization approach, some researchers consider that products are transported through the polyducts in units of volume called packages or batches.

The consecutive transmission of two different products may cause contamination of both fluids. In some cases, the contaminated fluids have to return to the refinery to be reprocessed. It increases the production costs and causes delay in the delivery. The consecutive transmission of two different products is called fragmentation.

The problem consists in scheduling the sending of packages from source nodes to terminals satisfying constraints related to production, demand, time and storage capacity. Some researchers consider the problem with the following two objectives: to minimize the time needed to for transporting the set of packages through the network and to minimize fragmentation. This approach is also adopted in this paper that introduces a Particle Swarm Optimization algorithm for the investigated problem.

Particle Swarm Optimization (PSO) is a technique that has shown good results in industrial applications. PSO algorithms were first introduced in the context of continuous optimization problems and research in this area has significantly grown in the last few years with a number of successful applications concerning single and multi-objective optimization problems [2], [7]. Motivated by the success of PSO algorithms, researchers that deal with combinatorial optimization problems have investigated ways to adapt the original proposal to the discrete case. Although, a number of applications of PSO to discrete problems do exist, these algorithms are not as effective as they are when applied to continuous problems, once they are easily trapped into local optima. To overcome the efficiency problems, the researchers have been focused on new PSO versions and hybridization with other techniques. The algorithm presented in this paper is based on a PSO version for discrete problems presented also for a biobjective optimization problem [6]. The results

978-0-7695-3872-3/09 $26.00 © 2009 IEEE
DOI 10.1109/ISDA.2009.185

IEEE computer society

obtained with the proposed approach are compared with the results obtained by two Genetic Algorithms proposed previously for the problem.

The paper is organized as follows. The model of the distribution network is addressed in Section 2. The proposed algorithm is described in Section 3. A computational experiment is reported in Section 4. Finally, some concluding remarks are presented in Section 5.

2. Model of the Distribution Network

This paper adopts the model presented by De la Cruz et al. [3], where a simplified version of an actual network is presented. The network is composed of nodes that are divided in three categories: sources (refineries), terminals (clients) and intermediary nodes that represent storage tanks. An example is shown in Fig. 1, where refineries are represented by nodes N_1 and N_2, N_3 and N_4 represent intermediary nodes and the points of destination or terminals are represented by nodes N_5, N_6 and N_7. The network has 9 polyducts that are represented by 10 arrows. The directions of the arrows show the flow direction. Arrows D5 and D8 refer to the same polyduct that allows flow in two directions.

Different products are delivered in this network. The model considers that the number of storage tanks at each node corresponds to the number of products that intermediary or terminal nodes are allowed to receive. For example, if four different types of products can be received by a node, then there are four different tanks, one for each product, at this node. Some simplifications of a real network are adopted in this paper. It is assumed that all polyducts have the same diameter and characteristics. The same flow rate is considered for all products that occupy the same volume in the polyducts.

The products are delivered as discrete packages. A package represents a minimum volume that is transported during a unit time. The number of packages is utilized to refer to the tanks capacity and the demand. Numbers in links joining two nodes give the normalized distance in terms of units of time needed by a given packet to cover the whole pipe. Table 1 shows the units of time that the package takes to cross each connection shown in Fig. 1. For instance, the connection D7 in the polyduct linking nodes N_3 and N_6 means that one packet spends two periods of unit time to go from node N_3 to node N_6, or that the polyduct may contain two packets.

Given a planning horizon, divided in time units, a solution for this problem consists in defining which product is being sent by each source or intermediary node at each instant. Under the point of view of the connections, a solution can be represented by the product being sent at each time instant in each polyduct. The following constraints have to be satisfied. A minimal transmission of products is required in order to avoid paralyzing the production in the source nodes or exceed the storage tanks capacity. The demands of each terminal node have to be satisfied (and not exceeded). Bidirectional connections are not allowed to send flow in two different directions at the same time.

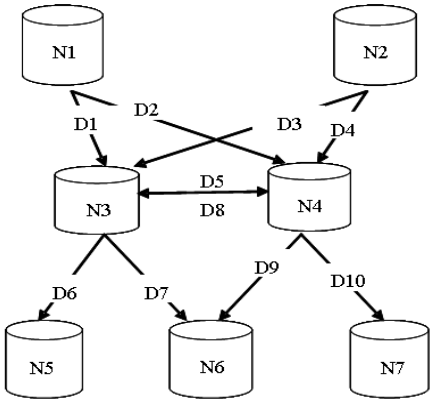

Figure 1. Example of a distribution network

Table 1. Units of time necessary for one package to traverse each connection

Connection	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}
Distance	1	3	3	2	3	4	2	3	3	2

The model is subjected to the following constraints:

1. The minimum production at each source node must be met. A minimum number of packages have to be sent to prevent the stoppage of production at the source nodes.
2. The demand of each destination node must be met.
3. There should be no collisions of packages in bidirectional connections.
4. The maximal capacity of each tank cannot be violated.
5. The packages must be delivered at the destination nodes at due times.

Let us consider the network of Fig. 1 and the transmission of four different products {1,2,3,4}. Node N_1 is the source of products 1 and 2, and node N_2 is the source of products 3 and 4. The remaining nodes can receive the four products. Table 2 shows one solution for the problem presented in Fig. 1 with a planning horizon of 10 time units. A 0 in line *i* and column *j* indicates that no product is being sent in connection D_i at time *j*.

Table 2. One solution for network of fig. 1

Polyduct	Time									
	1	2	3	4	5	6	7	8	9	10
D1	1	0	2	1	2	0	2	1	2	2
D2	0	1	2	1	1	1	2	2	1	1
D3	0	0	3	3	3	0	4	4	3	3
D4	3	0	4	0	3	4	0	3	3	0
D5	2	3	0	0	2	0	3	0	0	0
D6	3	3	4	4	2	2	1	2	1	2
D7	3	1	1	1	3	3	1	1	2	2
D8	0	0	0	0	0	0	0	0	0	3
D9	2	2	4	3	3	4	0	3	3	0
D10	3	3	4	0	2	1	1	2	4	3

Evolutionary algorithms were proposed for this problem by De la Cruz et al [3], [4] and Westphal [10]. De la Cruz et al. [3] consider four objectives to be optimized: two for time and two for fragmentation. They apply a multiobjective genetic algorithm to a network with 12 nodes and 21 connections, one of them is bidirectional. De la Cruz et al. [4] develop a multiobjective genetic algorithm, a Mathematical Programming algorithm and a hybrid algorithm where the solutions of the latter algorithm are used in the genetic algorithm. They apply their algorithms to one network with 7 nodes and 7 connections, one of them bidirectional. Westphal [10] presents a multiobjective genetic algorithm with elitism and niches.

The results of the multiobjective genetic algorithms proposed by De la Cruz et al. [3] and Westphal [10] are compared with the results of the approach proposed in this paper. The algorithm of De la Cruz et al. [3] by us and the other genetic algorithm was kindly made available by its author.

3. PSO for the Oil Products Distribution Problem

In the classical mono-objective PSO algorithms, each particle has a position and a velocity, knows its own position and the value associated with it, knows the best position it has ever achieved and the value associated with it and knows its neighbors, their best positions and their values. In many applications the neighbors can be replaced by one representative neighbor that is chosen in accordance with some specified criterion. The particles move composing three options of movement [9]: to follow its own way, to go back to its best previous position and to go towards its best neighbor's position. In this paper a social neighborhood approach is adopted. Social neighborhoods are based upon "relationships" that can be defined at the very beginning of the algorithm.

Usually, the particle's position is identified with one problem solution. The velocity defines how this position is modified.

Since in this paper, a discrete approach is adopted to deal with the oil derivatives distribution problem, a discrete PSO algorithm is developed for the problem. The proposed algorithm considers search strategies to modify the position of the particles. Therefore, each search strategy is considered as a movement alternative. The first option, that is the particle follows its own way, is implemented by means of a local search procedure [1]. The other two movement options concern a trajectory between the current position of a given particle and another position that can be the previous best position the considered particle has ever achieved or the position of the particle's best neighbor. In these cases the search strategy used in this paper is the Path-relinking [5].

In the proposed algorithm, one movement alternative is randomly assigned to each particle, according to a given probability. The particle moves accordingly and the position is modified. The pseudo-code of the proposed algorithm is given in figure 2.

```
/* Define initial probabilities x + y + z = 1 (100%) */
pr1 = x; pr2 = y; pr3 = z
iter = 0
for i =1 to #particles
    generate_particle(p[i])
Repeat
   timeRepair(); demandRepair()
   capacityRepair(); collisionRepair()
   for i = 1 to #particles
        update_arc(Local_arc_p[i], p[i])
        update_arc(Global_arc, p[i])
   for i = 1 to #particles
        change_position(p[i],pr1, pr2, pr3)
   pr1 = pr1 * 0.95
   pr2 = pr2 * 1.01
   pr3 = pr3 * 100% - (pr1 + pr2)
   iter = iter + 1
Until (iter > #max_iter)
Return(Global_arc)
```

Figure 2. Pseudo-code of PSO algorithm

At the beginning, initial probabilities $pr1$, $pr2$ and $pr3$ are assigned to each possible movement alternative corresponding respectively to the likelihood the particle follows its own way, goes toward the best previous position and goes toward its best neighbor. The probabilities are updated and the algorithm proceeds to the following iteration. Initially, a high value is set to $pr1$, and low values are assigned to $pr2$ and $pr3$. The goal is to allow that individualistic moves occur more frequently in the first iterations. During the execution this situation is being modified and, at the final

iterations, *pr*3 has the highest value in order to intensify the search in good regions of the search space.

Once a bi-objective problem is being investigated, the concepts of "*best position ever achieved*" and "*best neighbor's position*" are reviewed. Instead of a single solution for each of these concepts the algorithm deals with limited archives of non-dominated solutions. A set of, at most, 10 solutions is assigned to each particle *i*, called *Local_arc_p*[*i*]. That archive stores non-dominated solutions that correspond to positions previously achieved by particle *i*. A set of non-dominated solutions found by the algorithm is stored in *Global_arc*, a limited archive with, at most, 20 solutions. This archive replaces the concept of the best neighbor. At the beginning, the elements of *Global_arc* are the non-dominated solutions generated in the initial population. The management of archive *Global_arc* is based on the proposal presented by Knowles [8]. New non-dominated solutions are always added to the archive until it reaches the maximal size. A bidimensional grid is used to manage this archive. The grid has fixed size. If a new non-dominated solution is generated and the archive is full, then one solution is randomly chosen from the most populous region of the grid to be replaced by the new solution. The solutions that leave the global archive are stored in another archive, *Garb_arc*, that is not scanned until the end of the execution of the algorithm. The output of the algorithm is a set of non-dominated solutions among the solutions of *Global_arc* and *Garb_arc*.

Each solution, considered as a particle position, is encoded in a vector as illustrated in table 3 where a solution for a problem with four products, ten connections and a planning horizon of 10 units of time is sketched. The products are numbered from 1 to 4, a 0 indicates that no product is being sent by the corresponding connection at the corresponding time unit. The bidirectional connections are unfolded in two connections, each representing one flow direction. In this example, it is possible to observe that product 1 is being sent in connection D1 at time 1 and by connection D2 at time 10, product 2 is being sent by connections D5 and D9 at time 1 and in connections D1, D6 and D7 at time 10, and so on.

Table 3. Example of vector solutions

Time	Moment 1	...	Moment 10
Connection	1 2 3 4 5 6 7 8 9 10	...	1 2 3 4 5 6 7 8 9 10
Product	1 0 0 3 2 3 3 0 2 3	...	2 1 3 0 0 2 2 3 0 3

The initial population was generated randomly, choosing for each moment, a product to be sent in each connection, within the range of products possible for each connection [3]. Then, the defragmentation heuristic by Westphal [10] is used. This heuristic aims to group successive submissions of the same product that are in the same connection.

The procedure *change_position*( ) receives 4 input parameters: the particle's position and the values of *pr*1, *pr*2 and *pr*3. According to these probability values, one movement alternative is chosen for the particle. The corresponding search strategy is, then, applied to the particle that modifies its position.

In the local search, a solution π is generated from a starting solution π' by changing the value of one position of π. It is done by changing the product being sent in a given connection and moment. All possible products are tested for all connections and instants. The search is interrupted if a solution better than the starting solution is generated. A solution π is better than π' if π is non-dominated regarding the solutions in *Global_arc*.

In order to take a particle from one position to another, the technique of path-relinking is utilized. The position of the particle is considered as the starting solution. In the case the particle goes toward its best neighbor, a solution of *Global_arc* is randomly chosen, with uniform probability, as the position of the best neighbor. In the case the particle goes toward its best previous position, a local archive of non-dominated solutions is maintained for each particle. A solution of this local archive is chosen at random, with uniform probability, to play the role of the best previous position of the particle. The local archive stores non-dominated solutions that were generated during modifications of the position of the correspondent particle. Local archives have at most 10 solutions. If a new non-dominated solution is generated and the local archive has already reached its maximal size, then a solution chosen at random in the local archive is replaced by the new solution. Non-dominated solutions generated during the operation of path-relinking regarding the local and the global archive update these repositories in accordance with the strategies previously explained. Given the starting, π, and final position, π', then the algorithm replaces iteratively, the product being sent in connection i, $i = 1,\ldots,$ *nconnec*, in instant j, $j = 1,\ldots,$ *ninst*, by the product in connection i in instant j in solution π', where *nconnec* and *ninst* are the number of connections and instants.

The algorithm makes use of four repairing functions. The first one, *timeRepair*( ), verifies if the limits for arrival time of the packages are being violated. The packages that violate the limit of maximum arrival time are removed. The second repairing function, *demandRepair*( ), verifies for each time instant if the

demand of each product has been satisfied. If the demanded number of packages of a given product has been exceeded, then the exceeding packages are removed. The third repairing function, *capacityRepair*(), deals with the tanks capacity. The procedure verifies if the minimum capacity of origin nodes and the maximum capacity of destination nodes are being violated. If the minimum capacity of the tanks is violated, the package is removed from the connection and added to the tank of the origin. If the maximum in the destination node is violated the packages that violate this constraint are removed. The last repairing function, *collisionRepair*( ), checks the occurrence of collisions in bidirectional connections. If two packages sent at time *t1* and *t2*, *t1* < *t2*, collide in the same connection, then the package sent on time t2 is removed. If *t1* = t2, then one package is selected randomly, with uniform probability, and removed.

The stopping criterion of the algorithm presented in figure 2 is a maximum number of iterations, #*max_iter*.

4. Computational Experiments

The proposed algorithm was implemented in C and executed on Pentium IV, 1GB of RAM, under Linux, gcc compiler. The algorithm was applied to a set of 6 instances. The instances were obtained from the work of De la Cruz et al. [3].

The results obtained with the proposed algorithm are compared with the ones obtained by the genetic algorithms presented by De la Cruz et al. [3] and Westphal [10]. The former algorithm was implemented by the authors and the latter was made available by its author. For each instance, 20 independent executions of each algorithm were performed. Two networks are the bases for the 6 instances. The first network consists of 7 nodes, 8 unidirectional connections and one bidirectional connection. The second network has 12 nodes, 20 unidirectional connections and one bidirectional connection. For each of these networks, different initial conditions are established. The names of the instances are ind_*XX*_*YY*, where *XX* defines the number of nodes in the network and *YY* refers to the number of units of time in the planning horizon. Instances ind_07_15 and ind_12_15 are introduced in the paper by De la Cruz et al. [3].

For all instances, the minimum capacity of tanks and the minimum time are equal to zero. The number of products considered in all cases is 4. The initial conditions of instances ind_07_50, ind_07_150, ind_12_50 and ind_12_150 are presented in Table 4, where the identification 1, 2, 3 and 4 are used, respectively, for each instance. In table 4, M represents the maximum storage capacity of the tanks correspondents to the nodes, and D represents the demanded amount of each product by each node.

Table 4. Initial networks configuration

	Node	Prod A		Prod B		Prod C		Prod D	
		M	D	M	D	M	D	M	D
1	1	400	0	400	0	400	0	400	0
	2	400	0	400	0	400	0	400	0
	3	300	0	300	0	300	0	300	0
	4	300	0	300	0	300	0	300	0
	5	400	3	400	3	400	3	400	3
	6	400	3	400	3	400	3	400	3
	7	400	3	400	3	400	3	400	3
2	1	500	0	500	0	500	0	500	0
	2	500	0	500	0	500	0	500	0
	3	300	0	300	0	300	0	300	0
	4	300	0	300	0	300	0	300	0
	5	300	10	300	10	300	10	300	10
	6	500	11	500	12	500	10	500	9
	7	500	14	500	18	500	20	500	11
3	1	400	0	400	0	400	0	400	0
	2	400	0	400	0	400	0	400	0
	3	400	0	400	0	400	0	400	0
	4	300	0	300	0	300	0	300	0
	5	300	0	300	0	300	0	300	0
	6	300	0	300	0	300	0	300	0
	7	300	0	300	0	300	0	300	0
	8	300	0	300	0	300	0	300	0
	9	400	20	400	21	400	21	400	20
	10	400	19	400	18	400	16	400	15
	11	400	21	400	22	400	22	400	21
	12	400	18	400	19	400	20	400	22
4	1	500	0	500	0	500	0	500	0
	2	500	0	500	0	500	0	500	0
	3	500	0	500	0	500	0	500	0
	4	300	0	300	0	300	0	300	0
	5	300	0	300	0	300	0	300	0
	6	300	0	300	0	300	0	300	0
	7	300	0	300	0	300	0	300	0
	8	300	0	300	0	300	0	300	0
	9	500	21	500	20	500	20	500	21
	10	500	20	500	18	500	19	500	16
	11	500	22	500	22	500	21	500	20
	12	500	15	500	16	500	18	500	19

The objectives are the minimization of the total fragmentation and the minimization of the total time. The total fragmentation is the sum of the fragmentations in all network connections.

It is important to note that the algorithm of De la Cruz et al. [3] uses a multi-objective approach where one objective is fixed and the other is optimized. It optimizes the time through an Integer Linear Programming approach; afterwards, the fragmentation is optimized through of the genetic algorithm. In the algorithm Westphal [10] uses the weighting method which consists in applying a weight to each criterion in order to aggregate them in a single objective function. Only one solution for each test case in each independent run is generated by these algorithms.

Table 5 shows the results of the computational experiment, where *#ND_Sol* shows the average size of *Global_arc*, *#D_Sol_C* and *#D_Sol_W* show the average number of solutions produced by the PSO algorithm that strictly dominate the solutions generated by the algorithms presented by De la Cruz et al. [3] and Westphal [10], respectively.

Table 5. Computational results

Instance	#ND_Sol	#D_Sol_C	#D_Sol_W
Ind_07_15	2	1	2
Ind_07_400	20	3	6
Ind_07_500	21	3	8
Ind_12_15	23	8	18
Ind_12_400	22	6	17
Ind_12_500	22	6	13

No solution produced by the other algorithms strictly dominates any solution generated by the proposed algorithm. The table 5 shows that the proposed approach is capable of producing a variety of solutions for each case, which makes the decision-maker can choose among them the most convenient. De la Cruz et al. [4] presents a solution for the Ind_07_15 instance that dominates one solution generated by the proposed approach and is non-dominated regarding the other solution.

5. Conclusion

This paper presented an algorithm based on Particle Swarm Optimization that was applied to the problem of distributing oil derivatives through a network of polyducts. The results generated with the application of the proposed technique were compared to the results of two genetic algorithms presented previously for the same problem. Although such algorithms have been based on concepts of optimization with multiple objectives, they generate only one solution to each instance of the problem. The algorithm reported in this paper generates a set of non-dominated solutions that gives the decision-maker a better overview of available solutions.

The proposed algorithm produces several solutions that strictly dominate the solutions generated by algorithms used for comparison.

Acknowledgments

This research was partially supported by the National Agency of Petroleum through the PRH-22 program, and by the CNPq. The authors wish to thank Henrique Westphal who kindly has sent the code of his genetic algorithm.

References

[1] E. Aarts, J. K. Lenstra, Local Search in Combinatorial Optimization. John Wiley & Sons, Chichester, England, 1997.

[2] C.A.C. Coello, G.T. Pulido, M.S. Lechuga, "Handling multiple objectives with particle swarm optimization", *IEEE Transactions on Evolutionary Computation* 8(3), 2004, pp. 256-279.

[3] J.M. De la Cruz, B. Andrés-Toro, A. Herrán-González, E. Besada-Portas, and P. Fernández-Blanco, "Multiobjective optimization of the transport in oil pipeline networks", *IEEE International Conference on Emerging Technologies and Factory Automation* 1, 2003, pp. 566-573.

[4] J.M. De la Cruz, A. Herrán-González, J. L. Risco-Martín, and B. Andrés-Toro, "Hybrid heuristic and mathematical programming in oil pipelines networks: Use of immigrants", *Journal of Zhejiang University SCIENCE*, 6A(1), 2005, pp. 9-19.

[5] F. Glover, M. Laguna, R. Martí, "Fundamentals of scatter search and path relinking", *Control and Cybernetics* 29(3), 2000, pp. 653-684.

[6] E.F.G. Goldbarg, G.R. Souza, M.C. Goldbarg, "Particle swarm optimization for the bi-objective degree-constrained minimum spanning tree", *Proceedings of the 2006 Congress on Evolutionary Computation*, 1, 2006, pp. 420-427.

[7] J. Kennedy, R. Eberhart, "Particle swarm optimization", *IEEE International Conference on Neural Networks* 4, 1995, pp. 1942-1948.

[8] J. D. Knowles, "Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization", PhD Thesis. Department of Computer Science, University of Reading, Reading, UK, 2002.

[9] G.C. Onwubolu, and M. Clerc, "Optimal path for automated drilling operations by a new heuristic approach using particle swarm optimization", *International Journal of Production Research* 42(3), 2004, pp. 473-491.

[10] H. Westphal, Algoritmo Genético Aplicado a Otimização Multiobjetivo em Redes de Distribuição de Petróleos e Derivados, M. Sc. Dissertation, Universidade Tecnológica Federal do Paraná, 2006.