A GRASP FOR REAL LIFE INVENTORY ROUTING PROBLEM: APPLICATION TO BULK GAS DISTRIBUTION

Hugues Dubedout, Pierre Dejax, Nicoleta Neagu, Thomas Yeung

To cite this version:

Hugues Dubedout, Pierre Dejax, Nicoleta Neagu, Thomas Yeung. A GRASP FOR REAL LIFE INVENTORY ROUTING PROBLEM: APPLICATION TO BULK GAS DISTRIBUTION. 9th International Conference on Modeling, Optimization & SIMulation, Jun 2012, Bordeaux, France. 2012. <hal-00728664>

HAL Id: hal-00728664

https://hal.archives-ouvertes.fr/hal-00728664

Submitted on 30 Aug 2012

9th International Conference of Modeling, Optimization and Simulation - MOSIM'12
June 06-08, 2012 – Bordeaux - France
"Performance, interoperability and safety for sustainable development"

A GRASP FOR REAL LIFE INVENTORY ROUTING PROBLEM: APPLICATION TO BULK GAS DISTRIBUTION

Hugues DUBEDOUT[1 2], Pierre DEJAX[2], Nicoleta NEAGU[1], Thomas YEUNG[2]

[1]Air Liquide/ Centre de recherche Claude et Delorme
Jouy-en-Josas - France
hugues.dubedout@airliquide.com,
nicoleta.neagu@airliquide.com

[2]Ecole des Mines de Nantes / IrCCyN
Nantes - France
Thomas.Yeung@mines-nantes.fr
Pierre.Dejax@mines-nantes.fr

ABSTRACT:

In this paper we propose two versions of a GRASP algorithm for solving a real life Inventory Routing Problem (IRP) arising in bulk gas distribution and we show their benefits by comparison to an efficient existing specialized heuristic. The first approach uses a deterministic greedy heuristic and generates parallel solutions using an existing local search, and the second one is based on a randomized implementation of a greedy algorithm in order to find multiple initial solutions.
We perform a computational study in which the two procedures are applied on 16 real-life test cases. The outcome of the tests is promising as the proposed methods show improvement relatively to a state of the art local search algorithm. The tests show that our procedures yield a 5% average improvement of the objective function, whereas using the existing local search alone leads only to a 1.6% improvement with similar computation time and in comparable running conditions.

KEYWORDS: *Real-life inventory routing problems, local search, grasp, parallelization***,** *bulk gas distribution*

1 INTRODUCTION

The competition in local and global markets is pushing companies to look for reduction in their logistic costs as they represent an important part of the final cost of goods. To that aim, more centralized supply chain management systems are needed and thus, a recent approach in seeking logistic cost reductions is to consider the integration of transportation and inventory decisions.

In this paper we address the so called "Inventory Routing Problem" (IRP), see Bertazzi et al. (2008). This problem can be placed within the general framework of planning and optimizing the supply chain (supply chain), see Dejax, (2001) and Kok and Graves, (2003). The IRPs are among the most challenging extensions of vehicle routing problems and they present high interest as they occur often in real-life settings. The **IRPs problems** are twofold: **inventory management** at customer site, and **vehicle routing**, The IRP problems differ from vehicle routing problem by the fact that they coordinate the vehicles routing with the decision on customers' inventories, so called Vendor Management Inventory (VMI).

We consider the IRP problem in the context of the bulk gas distribution application. The production of liquid gas is realized in plants and the products are distributed from the stocks of these plants by vehicles which travel from "bases" and carry the products to customers. The customer delivery should be planned over several days to avoid breaking customer's stock and is based on a system of inventory management and customer forecasting models assumed to be reliable. The distribution is made either based on forecasts or on customer orders and obeys several constraints, including geographical and temporal. It takes place over a planning horizon (e.g., two weeks). It is based on the quality of demand forecasts and the availability of product inventory in plants.

For solving this IRP problem, a heuristic algorithm has been proposed by Benoist et al. (2011). It combines a greedy construction algorithm followed by a local search and is described in the next section. These procedures are be called "the original heuristic", the "greedy algorithm" and "the local search" respectively all throughout this paper.

In order to improve the solutions quality of this "local search" for solving the rich IRP problem, we propose two different designs of a GRASP framework for IRP problems in the context of a real-life setting of bulk gas distribution. We imbed the "local search" heuristic within the GRASP frameworks. The two GRASP methods are tested on 16 real-life test cases and compared to the results provided by the initial "local search". Extensive testing validates the efficiency of our procedure.

The paper is organized as follows: Section 2 describes the specificities of the IRP studied in this paper; Section 3 gives a state of the art on both the IRP and GRASP meta-heuristic. Section 4 describes the two GRASP implementations we propose. Section 5 describes the methodology used for testing as well as the test cases used. Section 6 presents the results and gives some insight on the influence of computation time. Lastly, Section 7 concludes this paper.

2 PROBLEM SETTING

In this section, we describe the real life IRP problem as it occurs in bulk gas distribution. Gases are produced at the vendor's plants and are consumed at customer sites. Both plants and customers store the product in tanks. Reliable forecast of production at plants is known over a short-term horizon.
The supply management provided by the vendor at the customer site can be of two types:

- The first one, called the "Vendor Managed Inventory", corresponds to customers for which the supplier decides the delivery schedule. For most of these customers, a consumption forecast is available over a short-term horizon. The inventory of each customer must be replenished by tank trucks so as to never fall under its safety level.

- The second one, called "Order-based resupply", corresponds to customers who send orders to the vendor, specifying the desired quantity and the time window in which the delivery must be done.

The constraints that consist of satisfying orders (no missed orders) and maintaining inventory levels above safety levels (no stock outs) are defined as soft, since the existence of an admissible solution is not ensured in real-life conditions. Three hierarchical objectives are used. The most important one is to minimize the number of customer orders that were not met. The second is to minimize the time spend by customers with an inventory below their safety level. Lastly the cost per unit of product delivered is minimized.
In order to solve this problem, a local search solution was designed. This local search focuses on testing as much 'moves' as possible during the allowed computation time. We refer the interested reader to Benoist *et al.* (2011) for more detail on the heuristic. However, as all local search heuristics, it can get stuck within local optima. When this happens, even increasing significantly the number of iterations performed does not allow finding good improvement to the solution. In order to keep improving the solution, one has to find another way to explore the solution space. One approach is to generate various solutions constructed with different methods and then select the best one.
Therefore, in this paper we explore for new approaches based on parallel computing as well as a GRASP heuristic in order to find better solutions.

3 STATE OF THE ART

This section presents the state of the art on the Inventory Routing Problem and GRASP. Firstly, it focuses on the deterministic version of the Inventory Routing Problem, reviewing previous solving method, heuristic or exact. The second part of the state of the art focuses on previous applications of the GRASP metaheuristic.

3.1 The inventory routing problem

In this section, we give a description of the work already done in the Operations Research community on the Inventory Routing Problem (IRP).

3.1.1 Heuristics

The first inventory routing model appeared in the 1980s. These papers are, for the most part, inspired by application of the vehicle routing where inventory has to be considered. In order to overcome the high complexity of IRP models, most papers only take into consideration short term planning. For example Golden et Al. 1984, focuses on maintaining a "good" level of inventory at the customer on a single day while minimizing the costs. Their heuristic computes the urgency of each customer in order to decide which customers need to be delivered. Customers are then iteratively added into shifts until the time limits for all shifts are reached. In this model, each day is treated as a different entity.
Chien et Al. (1989) also use a single day model, but consider a multiple-day time horizon. After the decisions are made for the first day, the result is passed to the next day in order to compute the profit for the next days. A MIP is used for both the inventory and routing decision and a Lagrangean dual ascent method is used to solve the program.
In 2000, Bertazzi, Speranza and Ulkovich propose a model based on previous work of Speranza and Ulkovich. They consider a supplier and several customers with given production rate and demand. The specificity of their model is that the shipment from the supplier to any customer can only occur within a given set of frequencies. E.g. a customer would be delivered every 5 days. Even with a single customer, this problem is NP-Hard, and thus, a local search based heuristic is used to find good solutions.
Bertazzi et al (2005) focus on the order up to level policy. They consider a single vehicle model, with inventory costs, both at the supplier and the customer site. The inventory cost is computed at each time step, and is equal to the quantity of product stored times an inventory cost, which is a parameter for each customer. They use a two-step heuristic, the first step being a greedy algorithm to create a feasible solution. After that, they improve the solution by removing a pair of customers and then reinserting them in the current solution. If a better solution is created, then they repeat the whole improvement phase, if not, then they continue until all pairs

of customers have been removed and reinserted. This algorithm is a local search with no random parameters.

Bertazzi et al. (2005) go further and propose solution for different delivery policies, and also compare the Vendor Managed Inventory (VMI) policy with the Retailer Managed (RMI), in which the retailers place orders to be delivered. The specificity of their model is that the decision variables include the production to be made for each time period. The production cost includes a fixed set up cost, and a variable cost that is charged for each unit produced. Two different VMI policies are studied. The first one is the Order-up-to level policy where the quantity shipped to the retailer fills the tank capacity, the second one is the fill-fill-dump policy, in which the order-up-to level quantity is shipped to all but the last retailer of each route, and the quantity delivered to the last customer is the minimum between the order-up-to level policy and the remaining capacity of the vehicle. Bertazzi et al. decomposed the problem into two sub problems: the distribution sub problem and then the production sub problem. Two approaches are suggested. In the first one, the distribution sub problem is solved first, assuming that all retailers are served every day. After the distribution is solved, the production problem is solved again. The other approach consists of solving first the distribution sub problem and then the production sub problem. Both approach show similar results.

The RMI policy is simulated with the following rule: each customer that will have a run out at time t+1 is visited at time t. The tests were run on a set of instances with 50 customers and a time horizon of 30 days. The results clearly indicate that the VMI based heuristic yields much better results than the RMI policy, with an average 60% total cost decrease.

Savelsbergh and Song (2008) study an industrial real life problem. A randomized greedy heuristic is used to solve the IRP with continuous moves, whereas the volume to be delivered is being computed via linear programming. They also propose a discrete time mathematical model, solved with a Branch and Cut algorithm. The results are presented on instances with a 5 days time horizon with 2 plants and 50 customers and 1 hour time step. The branch and cut algorithm manages to find the optimal solution with 30 min average computation time. However, with bigger time steps (2 hours for the first 2 days, then 4 hours for the next 3) , computation time can be reduced to less than 2 minutes with a solution quality decrease of 3%. On larger instances (3 plants,3 vehicles, 100 customers), the computation time increases to an average of 2 days, thus making the use of the exact model unusable in practice. However, the proposed heuristic manages to find solutions with less than 5% optimality gap in less than 5 minutes.

Abdelmaguid et al. (2009) propose a mixed integer programming formulation for a single depot, multi vehicle, backlogging model. However, even with less than 15 customers and 2 vehicles, they did not manage to obtain the optimal solution within a one hour time limit, thus motivating a heuristic approach. As with many heuristic approaches, the approach proposed by Abdelmaguid et al. consists of a constructive phase in which a solution is built, and an improvement phase in which the solution is further improved. The constructive heuristic is an Estimated Transportation Cost Heuristic, in which all decisions are taken based on an estimation of the transportation costs. Then, a local search is performed in order to improve the solution found. The local search focuses on modifying the quantities delivered and on delivery exchanges. For large instances, the heuristic gives better results than the CPLEX solver. However, precise computation time for the heuristic is not given.

Boudia and Prins (2007) propose a memetic algorithm with population management to solve an integrated production-distribution problem. They apply their algorithm to instances with up to 200 customers and 20 periods, and compare the results obtained to the improved GRASP presented in Boudia et al. (2006). They show that better results are obtained at the cost of a reasonable computation time increase.

Due to the extensive research done on the Inventory Routing Problem, it would be impossible to have an exhaustive literature survey here. We refer the interested reader to the survey made by Andersson et al (2010) for additional references on the Inventory Routing Problem.

3.1.2 Exact Methods

Due to the complexity of the inventory routing problem, very few effective exact methods have been developed. Some papers gives a standard MIP formulation (e.g. Abdelmaguid and Dessouky (2009)) and solve it within a given time limit in order to obtain lower and upper bounds for the cost of the optimal solution. These bounds are then used as benchmark for heuristic. Most of these MIP formulations are computationally intractable for large instances. However, several recent papers have developed good algorithms for finding the optimal solution of the inventory routing problems.

The first one is a paper of Archetti et al. (2009). They consider a model with a single vehicle and deterministic demand, an order-up-to-level policy and do not allow backlogging at the customers. A branch-and-cut algorithm is used. The second paper is from Solyali and Süral(2011). They improved the results of Archetti et al. (2009) by proposing a strong MIP formulation within a branch and cut algorithm. While both use a two-index vehicle flow formulation for the routing decision, Archetti et al.(2009) decided to use standard inventory balance constraints where Solyali and Süral(2011) used a shortest path formulation which seems to yield better results.

Another MIP formulation was proposed by Solyali, Cordeau and Laporte (2010) for a single vehicle, deterministic demand model with backlogging penalty. Using a thigh formulation for inventory decisions and a two-index flow formulation for the routing decisions, they propose a branch and cut algorithm that yields better results than Abdelmaguid and Dessouky's (2009) MIP formulation.

Oppen et al. (2010) propose a column generation method to solve a rich Inventory Routing Problem in the field of meat industry. They present results on multiple instances, with 20 to 27 orders. While solutions are found for problems with less than 25 customers, the computation time ranges from several minutes to few hours depending on the characteristics of the instance.
Due to the intrinsic complexity of the inventory routing problem, most solutions developed for the deterministic version are heuristic approaches. Such heuristics include lagrangian relaxation, local search, and decompositions into sub problems.

3.2 Greedy Randomized Adaptative Search Procedure

The Greedy Randomized Adaptative Search Procedure (GRASP) is a multi-start meta-heuristic that was first described by Feo and Resende (1989, 1995). Each iteration of the procedure consists of two phases: a construction phase and an optimization phase. During the construction phase, a feasible solution is iteratively constructed, using a randomized greedy algorithm (see Section 5.2). Then, during the optimization phase, this feasible solution is improved, often by the use of a local search procedure.
The GRASP methodology has been used in many different problems with great success. Resende and Ribeiro (2003) as well as Festa and Resende (2001) present application in fields such as Routing, Assignment problems, Scheduling and telecommunication. An example of application of the GRASP methodology to an Inventory Routing Problem can be found in Grellier et al (2004).
Parallelization approaches are very appropriate for GRASP methodology, as explained in Cung et al (2001). Because each iteration can be run in a different thread with no need of interacting with each other, the gain in time for using parallelism is close to linear on the number of processor used.
Boudia et al. (2006) use the GRASP for solving a combined production-distribution problem. The GRASP methodology they proposed is improved using either a reactive mechanism or path relinking. The reactive mechanism optimizes the value of one parameter after each iteration, trying to obtain the value that gives the best results on average. The path relinking is used as a post optimization process, where the path between several elite solutions found by the GRASP is explored, thus potentially finding better solution. Results show that both the reactive GRASP and the path relinking method provide better results than a simple GRASP.

Our goal in this paper is to improve the performance of a local search by upgrading it to a meta heuristic and by using parallelism. GRASP seemed the appropriate meta-heuristic for solving our problem as it allows to easily imbed an existing heuristic, and is can be easily parallelized.

4 GRASP DESIGN AND IMPLEMENTATION METHODOLOGY

In this section, we describe the design of the GRASP, and its implementation. We propose two different GRASP approaches which are based on the integration of the specialized heuristic proposed by Benoist et al (2011).

The GRASP algorithm, as described by Feo and Resende (1995), consists of multiple iterations of two successive phases: a construction phase, in which an initial solution is constructed, using a randomized greedy algorithm, and an improvement phase, during which a local search algorithm is used to further optimize the solution previously constructed.

As a first approach, we do not include the multi-start component of the GRASP meta-heuristic. Instead, the construction phase is only run once before any iteration. Then during each iteration, the local search optimization is performed from start using a different random seed. Thus, the final solution found by each iteration is different from each other.
The second approach includes the multi start component, as it is usually done within a GRASP meta-heuristic. In the following sections, we are going to describe the algorithms used for each approach.

4.1 Single start GRASP

In this implementation, the construction phase is only run once, and the feasible solution found is used as the starting solution for each optimization iteration.

4.1.1 Construction phase

In the construction implemented, we integrated the greedy algorithm originally use in the heuristic of presented in Figure 1 shows the main methodology described by to Benoist *et al.* (2011):

```
Procedure Greedy
1 $Solution \leftarrow \phi$
2 List all demands and orders
3 while $Solution$ is not completed do
4     Select the demand d with the earliest deadline;
5     Create the cheapest delivery to satisfy d;
6     Update the Solution to include this delivery;
7     Update the list of demands and orders;
8 end;
9 return Solution;
End Greedy
```

Figure 1 : Deterministic Greedy Algorithm.

This algorithm starts with an empty solution, and lists all the demands and order to satisfy. Then, the demand with the earliest deadline is selected, and the incremental costs of all possible insertions into the current solution (insertion within an existing shift, or creation of a new shift) are evaluated. The best insertion is then selected, and both the solution and the list of demands are updated.

4.1.2 Improvement phases

As described in Cung et al (2001), the GRASP metaheuristic is easy to parallelize as all iterations are independent from each other; i.e. they do not depend of the result of the previous iteration. Therefore, each improvement phase can be executed in a separate thread. The algorithm used for the improvement phase within each thread is the local search component of the original heuristic presented in Benoist et al. (2011). A large set of moves are used: The insertion, deletion and ejection moves apply to a customer within a shift. Swap and Move movements are defined both within routes, and between different routes. A mirror move is also present.

Figure 2 presents the pseudo code used for the parallelizing local search. At first, an array for storing all the solutions is created. Then, all the local searches are launched into separate threads, with different random seed. Once all the threads have finished, the best solution found is returned.

```
Procedure Single_Start_GRASP (NbIterations)
1 Read Input();
2 Solution_init $\leftarrow$Greedy(Input)
3 Create array Results of size NbIterations
4  for $k = 1 :: NbIterations$ do
5     Launch new thread;
6     Set seed = k;
7     Results[k] $\leftarrow$Local_Search(
      seed, Solution_init);
8     end thread
0  end;
10 Wait for all threads to end;
11 Solution $\leftarrow$Best_Solution (Results)
12 Return Solution;
End Single_Start_GRASP.
```

Figure 2 : Parallel local search pseudo code.

4.2 Multi start GRASP

In this section, we describe the implementation of the GRASP algorithm, including the multi-start component. It uses a randomized greedy algorithm in order to provide multiple initial solutions for a local search heuristic. The best solution found by the local search is kept as the result.

4.2.1 Construction Phase

In order to generate multiple start solutions, Resende and Ribeiro, (2002) suggested the use of a randomized greedy algorithm. Figure 3 presents the pseudo code of the generic Greedy_Randomized_Construction as they suggest it. It starts with an empty solution. The incremental cost of each candidate element (e.g., insertion place within route planning) is evaluated, and a *restricted candidate list* (RCL) is created with the candidate having the smallest incremental cost. This list can be limited either by the number of element (i.e., the k better candidates are selected for the RCL) or by a threshold value (i.e. all candidates whose incremental cost is smaller than *Max_Value* are selected).

Once the RCL is constructed, the candidate element to be added to the solution is randomly selected. The solution is updated to include this element. This constitutes the randomized part of the procedure. The list of candidate elements is then updated and the incremental cost of each element is re-evaluated. This constitutes the adaptive part of the procedure. A new RCL is then created, and the procedure continues until the solution is completed.

```
Procedure
Greedy_Randomized_Construction (Seed)
1 $Solution \leftarrow \phi$
2 Evaluate the incremental costs of the candidate elements;
3 while $Solution$ is not completed do
4    Build the restricted candidate list (RCL)
5  Select and element s from the RCL at random;
6   $Solution \leftarrow Solution \cup \{s\}$;
7  Reevaluate the incremental cost
8 end;
9 Return Solution;
End GRASP
```

Figure 3 : Pseudo code of the generic randomized greedy procedure

Note that the procedure described in Figure 3 is already very close to the greedy algorithm described in Section 4.1.1. One approach for changing it from a deterministic procedure to a randomized procedure is to not always select the cheapest delivery. Instead, the delivery to be included in the solution is selected from the k cheapest possible deliveries. In order to do this a restricted candidate list of k elements is built during the evaluation of the cost of all possible deliveries.

```
Procedure Randomized_Greedy (k, seed)
1 $Solution \leftarrow \phi$
2 List all demands and orders;
3 while $Solution$ is not completed do
```

```
4 Select the demand d with the earliest deadline;
5     Create the RCL with the k cheapest deliveries that satisfy d;
6     Randomly select one delivery from the RCL;
7 Update the Solution to include this delivery;
8 Update the list of demands and orders;
9 end;
10 Return Solution;
End Randomized Greedy
```

Figure 4 : Pseudo code of the randomized greedy procedure.

4.2.2 Improvement phase

Unlike the single start GRASP described in Section 4.1.2 the multi-start GRASP executes both the construction phase and the improvement phase during each iteration of the algorithm. This creates different starting solution for the improvement phase, and thus can explore a larger range of the solution space. One drawback of this method is that the improvement phase must compensate for potentially worse starting solution.
Figure 5 describes the overall multi-start GRASP procedure. For each of the *NBIteration* iterations, a random initial solution is created, using the randomized greedy procedure. Then, this initial solution is optimized using the local search procedure. Each iteration being run in a separate thread. If the solution found by the local search is better than the current best solution, the new found solution is kept as the new best solution.

```
Procedure Multi_Start_GRASP (NbIterations)
1 Read Input ();
2 Create array Results of size NbIterations
3  for $k = 1 :: NbIterations$ do
4      Launch new thread;
5      Set seed = k;
6      Solution_init ← Randomized_Greedy (seed, Input)
7      Results[k] ← Local_Search (seed, Solution_init);
8      end thread
0  end;
10 Wait for all threads to end;
11 Solution ← Best_Solution (Results)
12 return Solution;
End Multi_Start_GRASP.
```

Figure 5: Pseudo Code for the GRASP meta-heuristic.

5 TESTING & RESULTS

In this section, we describe the methodology used to test and compare the different approaches aforementioned in Section 4.
The algorithm is implemented in C# and tested on a 16-core; 8GB RAM computer running windows server. Testing is performed on 16 different instances adapted from real life data. 3 different heuristics are compared. The first one is based on the basic deterministic greedy and local search heuristic. The second one is the single start GRASP. Lastly, the third heuristic is the multi-start GRASP methodology as described in Section 4.2.

5.1 Testing methodology

The parameter with the most influence on the quality of the result is the computation time. Before our study, a single run of the original heuristic algorithm performed 4 millions local search iterations, which took an average of 264 seconds to perform. Section 6 provides an example of how the local search sometimes keeps steadily improving the solution until up to 7 minutes of computation time before getting stuck in local optima. This also means that for both implementations the time needed for a single GRASP iteration would be over 4 minutes.

In order to show a fair comparison of all the three algorithms, we decided after preliminary testing to use the following parameters for each algorithm:

- 16 Million Iterations for the Local search.
- 20 Iterations for the two GRASP implementations. Each optimization phase of the GRASP performs 4 Million local search iterations.

Please note that the computation time needed for each algorithm is similar, the 16M iterations local search requiring an average of 1111 seconds to find a solution, and the 20 threads GRASP needs an average of respectively 1290 and 1295 seconds.

The original results heuristic, obtained using 4 Million local search iterations, are used as benchmark for comparison of the other 3 methods.

5.2 Test instances

16 different test cases were used for the evaluation of the three methods. All instances cover a 15 day horizon. Note that the computation time needed does not only depends on the size of the test case (i.e., the number of customers to be delivered), but also on its composition and the difficulty to satisfy the constraints such as the compatibility of the resources, the ratio between the quantity produced and the demand of the customers.

- **C area test cases**

These test cases contain 4 sources and 165 customers. The resources used to deliver products consist of 23 drivers, 6 trailers and 10 tractors. Data from 6 different time periods were used, resulting in 6 different instances.

- **B area test cases**

These test cases consist of 6 sources and 75 customers. The resources available for the deliveries are 35 drivers, 20 tractors and 4 trailers. Once again, data from 5 different time periods were used resulting in 5 different instances

- **B_L performance test case**

This test is based a 15 day horizon real life test case. It consists of 6 different sources and 175 customers. The resources available for the deliveries are 35 drivers, 20 tractors and 12 trailers.

- **A1 performance test cases**

This is a randomly generated instance that was initially created for performance testing of the local search solver. It consists of 2 sources and 83 customers, delivered by 20 drivers, 20 trailers and 10 tractors.

- **A2 performance test cases**

This is a randomly generated instance that was initially created for performance testing of local search solver. It consists of 1 source and 73 customers, delivered by 10 drivers, 10 trailers and 10 tractors.

- **A3 performance test cases**

This is a randomly generated instance that was initially created for performance testing of the local search solver. It consists of 4 sources and 149 customers, delivered by 20 drivers, 20 trailers and 20 tractors.

- **A4 performance test cases**

This is a randomly generated instance that was initially created for performance testing of the local search solver. It consists of 5 sources and 250 customers, delivered by 30 drivers, 30 trailers and 30 tractors.

6 RESULTS OBTAINED

6.1 Overall results

Table 1 shows the comparison between 4 million local search iterations for the original heuristic and 16 Million local search iterations. The two first lines show the value of the objective function and the computation time for the 4Million iteration local search for each instance. The last three lines show the value of the objective function for the 16 million iteration heuristic, the improvement compared to the 4 millions iterations local search and the tested heuristic. We see that even with 4 times the number of iterations and computation time, the results are only improved by 1.66%.

Table 2 presents the results obtained using the single start GRASP methodology. At the cost of a slightly longer computation time, we see that the average improvement is significantly better than the improvement obtained by simply increasing the number of iterations of the local search. The single start GRASP obtains a 5.44% average improvement of the solution compared to the heuristic.

Table 3 presents the results obtained using the Multi start GRASP methodology. The size of the restricted candidate list for the randomized greedy procedure was set to 3. This allows finding good initial solutions while still ensuring a high diversity amongst them. Tests were made with a restricted candidate list of size 5, but this led to an important deterioration of the initial solution, which the local search was not able to overcome. With these parameters, a 5.07% improvement of the objective function is obtained.

As shown by these results the solutions obtained using the both GRASP methodologies are similar in average. They show a significant improvement over the result of the heuristic alone. This proves that both methods succeed in exploring the solution space.

6.2 Result analysis

The results presented above can be completed with the following remarks.

The original heuristic manages to obtained good results for the C_5 and C_6 test cases. This indicates that it has not reached local optima. Whereas, in the test cases B_2 and B_1, the original heuristic has reached a local optima, and increasing the number of iteration does not lead to better solutions.

Even in the case where the 16M local search had good results (such as C_5 and C_6), the two implementations of the GRASP still manage to find better solutions. As expected, the best results for the GRASP are obtained on instance where the local search got stuck early in local optima. We also note the existence of instance where the 16M local search obtained better result that the single start GRASP.

The multi-start GRASP methodology seems to be better single start for all the C instances, with an average improvement of 6.5% in the logistic ratio, whereas the single start GRASP has an average improvement of 4.5%.
In two different instance (A3 and B_LAR_3), the multi start GRASP methodology only finds solutions worse than the original solution. This is due to the fact that the local search cannot compensate for the deterioration of the original solution.

We see that on average, the single start heuristic seems to be yielding the best results.
In the following section, we will detail the performance of the single start compared to the original heuristic in term of computation time.

Table 1 : Heuristic (Greedy + Local search) NbIteration increase

Iterations	instances	C_1	C_2	C_3	C_4	C_5	C_6	A1	A2	
4M	Value	0.030923	0.033268	0.030798	0.025052	0.030588	0.027956	0.022459	0.248066	
	Time(s)	307	321	345	303	329	286	367	460	
16M	Value	0.030618	0.032694	0.030291	0.024807	0.028854	0.026580	0.022359	0.247969	
	Time(s)	1289.2	1349.7	1450.9	1271.6	1381.6	1202.3	1544.4	1931.6	
	Impr(%)	0.99	1.73	1.65	0.98	5.67	4.92	0.44	0.04	
	instances	A3	A4	B 1	B 2	B 3	B 4	B 5	B LIN	Average
4M	Value	0.276152	0.258490	0.065691	0.048508	0.068380	0.073294	0.076952	0.025241	
	Time(s)	406	386	222	149	143	129	135	365	290.4
16M	Value	0.269808	0.249544	0.065691	0.048508	0.067690	0.071757	0.076952	0.024901	
	Time(s)	1706.1	1622.5	933.9	623.7	600.6	541.2	568.7	1534.5	1222.1
	Impr(%)	2.3	3.46	0	0	1.01	2.1	0	1.35	1.66

Table 2 : Local Search vs. Single Start GRASP

Grasp Iterations	Instances	C_1	C_2	C_3	C_4	C_5	C_6	A1	A2	
	Value	0.030923	0.033268	0.030798	0.025052	0.030588	0.027956	0.022459	0.248066	
1	Time(s)	307	321	345	303	329	286	367	460	
	Value	0.030124	0.032256	0.029771	0.023638	0.028383	0.026412	0.022274	0.222454	
	impr (%)	2,59	3,04	3,33	5,64	7,21	5,52	0,82	10,32	
20	Time(s)	1239	1295	1378	1123	1168	1197	1790	1859	
	Instances	A3	A4	B 1	B 2	B 3	B 4	B 5	B LIN	Average
	Value	0.276152	0.258490	0.065691	0.048508	0.068380	0.073294	0.076952	0.025241	
1	Time(s)	406	386	222	149	143	129	135	365	290.4
	Value	0.273250	0.253299	0.056911	0.042754	0.065557	0.066613	0.072846	0.024800	
	impr (%)	1,05	2,01	13,37	11,86	4,13	9,12	5,34	1,75	5,44
20	Time(s)	1817	1668	1759	698	715	680	731	1527	1290

Table 3 : Local Search vs. Multi Start GRASP

GRASP Iterations	Instances	C_1	C_2	C_3	C_4	C_5	C_6	A1	A2	
1	Value	0.030923	0.033268	0.030798	0.025052	0.030588	0.027956	0.022459	0.248066	
20	Value	0.029499	0.031435	0.028860	0.023705	0.028221	0.025425	0.022255	0.230067	
	impr (%)	4,60	5,51	6,29	5,38	7,74	9,05	0,91	7,26	
	Time(s)	1243	1299	1382	1127	1172	1201	1794	1867	
	Instances	A3	A4	B 1	B 2	B 3	B 4	B 5	B LIN	Average
1	Value	0.276152	0.258490	0.065691	0.048508	0.068380	0.073294	0.076952	0.025241	
20	Value	0.281377	0.257066	0.056911	0.042754	0.071016	0.067851	0.072846	0.024837	
	impr (%)	-1,89	0,55	13,37	11,86	-3,85	7,43	5,34	1,60	5,07
	Time(s)	1825	1673	1765	702	719	685	735	1531	1295

6.3 Computation time sensitivity

In this section, we focus on two very different instances (C_4 and B1) and analyze how the original heuristic and the single start GRASP compares with different values of computation time.

6.3.1 C_4 Test Case.

Figure 6 presents the global cost over time found by the Local Search, as well as the global cost over time found by the parallel Solution Generation Method. Please note that as it takes 5 minutes to generate a single solution, the solution generation method does not give any results before the 5 minute mark. Also, as the average solution generation time is close one minute (It takes 1290 seconds to generate 20 solutions), the number of solutions generated is a good estimate of the computation time.

In this case, we see that the local search converges in about 5 minutes, and then struggles a lot to keep improving the solution. This is a good indicator that the local search is stuck within local optima. On the other hand, by generating solutions in parallel, we manage to find a better solution within those 5 minutes, and also keep finding better solutions over time.

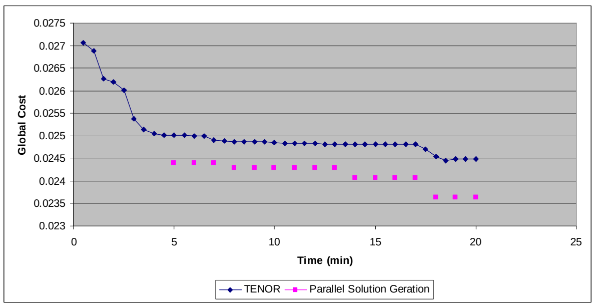

Figure 6 : Computation Time Influence: C_4.

Table 4 presents the improvement of the parallel local search versus the basic local search over time. Please note that it sometimes decreases as the local search manages to improve the solution and the parallel generation does not find a better solution. However, it mostly improves over time.

Table 4 : Improvement over time

Time (min)	Improvement(%)
5	2.47
6	2.39
7	2.04
8	2.36
9	2.33
10	2.23
11	2.21
12	2.16
13	2.15
14	3.02
15	2.97
16	2.96
17	2.95
18	3.69
19	3.45
20	3.43

6.3.2 B1 test case

Figure 7 also presents the global cost over time found by the Local Search, as well as the global cost over time found by the parallel solution generation method, but this time on the B Instance. The B instance converges very quickly into local optima, as there is almost no improvement to the solution after the first 30 seconds of computation time. However, we can see that the solution found is still far from optimal as the parallel solution generation method is able to find a much better solution in the same time.

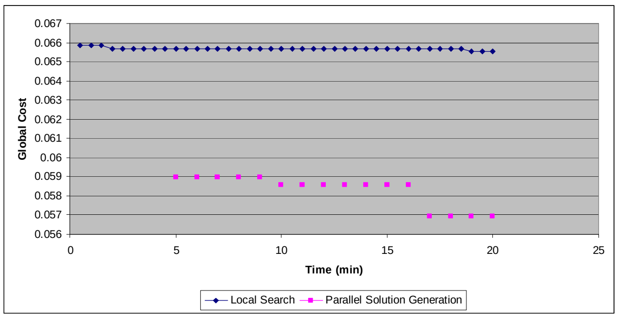

Figure 7 : Time Influence over time: B _1

Table 10 shows the average improvement of the parallel local search over the basic local search. We show here that the improvement goes from 10% in 5 minutes to 13% over a computation time of 20 minutes.

Table 5: Average Improvement over time

Time (min)	Improvement (%)
5	10.24
6	10.24
7	10.24
8	10.24
9	10.24
10	10.80
11	10.80
12	10.80
13	10.80
14	10.80
15	10.80
16	10.80
17	13.36
18	13.36
19	13.18
20	13.18

We see here that even for shorter computation time, the GRASP methodologies can yield better result than the original heuristic. In the cases where the original heuristic is stuck early in local optima, it also keeps improving over time, while the original heuristic doesn't.

7 CONCLUSIONS AND FUTURE WORK

In this paper we proposed two versions of the GRASP (a single start and a multi-start version) for the rich real life Inventory Routing Problem, as it occurs in bulk gas distribution. Our procedures imbed an existing specialized heuristic for this problem. We also used parallelization to reduce the time needed to perform all GRASP iterations. We conducted extensive texting on 16 data set representative of real life data set. We show that within reasonable computation time (less than 25 minutes) we manage to reduce the objective function value by 5.44% on average. This increase is much more significant than just letting the current heuristic run for about 20 minutes, which only gives an average reduction of 1.6% in objec-

tive function value. Achieving high performance in a limited amount of time is crucial in an industrial operational context. The obtained improvement represents considerable cost saving considered the scale of this large scale industrial problem.
Detailed analysis of our results does not show a significant difference between the results obtained by both GRASP implementations. However, the results produced by the multi start GRASP algorithm seem to be less stable than the parallel local search, and bears less warranty on the results of the final solution.
Possible future work include testing different randomization methods for the greedy algorithm, as well as looking at other known meta heuristics such as simulated annealing and/or tabu search. Another perspective would be to adapt to this real life problem other existing high performance IRP algorithm for generic problems.

8 ACKNOWLEDGMENT

This research was partially sponsored by a CIFRE contract by the ANRT (French National Agency for Technical Research).

REFERENCES

Abdelmaguid T.F., M.M. Dessouky, F. Ordóñez 2009: Heuristic approaches for the inventory routing distribution problem *Computers Industrial Engineering 56,* pp 1519-1534

Archetti A., L. Bertazzi, A. Hertz, M.G. Speranza 2009: " A Hybrid heuristic for an Inventory Routing Problem" *INFORMS journal of computing* Accepted Paper.

Benoist T., F. Gardi, A. JeanJean, 2011: "Randomized Local Search for Real-Life Inventory Routing" *Transportation Science,* Vol. 45, No. 3, Aug., pp. 381-398.

Bertazzi L., M.G. Speranza, W. Ukovich 2000: "Minimization of Logistic cost with Given Frequency" *Management Science* 46, pp 973-988

Bertazzi L., G. Paletta, M.G. Speranza 2002: "Deterministic Order-Up-To Level Policies in an Inventory Routing Problem" *Transportation Science* 36, pp 119-132

Bertazzi L, G. Paletta, M.G. Speranza 2005: "Minimizing the Total Cost in an Integrated Vendor managed Inventory System " *Journal of heuristics* 11, pp 393-419

Bertazzi L., M. Savelsbergh et M.G. Speranza 2008: « Inventory routing », in *The vehicle Routing Problem: Latest Advances and New Challenges,* B. Golden and E. Wasil edts., Springer, pp. 49-72.

Boudia M., M.A.O Louly, C.Prins 2006: "A reactive GRASP and pat relinking for a combined production-distribution problem" *Computers and operation Research* 34 pp. 3402-3419.

Chien T., A. Balakrihnan, R. Wong 1989: "An integrated inventory allocation and vehicle routing problem" *Transportation Science* 23 pp 67-76

Cung V-D, S.L. Martins, C.C Ribeiro, C.Roucairol 2002: "Strategies for the parallel implementation of Metaheuristics", *Essays and Surveys in Metaheuristics*, pp. 263-308, Kluwer Academic Publisher.

Dejax P. 2001: "Stratégie et implantation et Implantation du Système Logistique", ch.13 dans "*La Maîtrise des Flux*", J.-P. CAMPAGNE, coordonnateur, Editions Hermès.

Feo T.A., M.G.C Resende 1989: A probabilistic for a computationally difficult set covering problem. *Operation Research* 8 pp 67-71

Feo T.A., M.G.C Resende 1995: Greedy randomized Adaptative Search Procedures. *Journal of Global Optimization* 6 pp 109-133

Festa P, M.G.C Resende 2001: GRASP an annotated bibliography. *Essays and Surveys in meta-heuristics* Kluwer.

Golden B., A. Assad, R. Dahl 1984: "Analysis of a large scale vehicle routing problem with an inventory component", *Large Scale Systems* 7 pp 181-190

Grellier E, N. Brahimi, P.Dejax 2004: "GRASP for an Inventory Routing Problem: Design and implementation." *Research Report*, Ecole des Mines de Nantes.

De Kok A.G. et S.C. Graves edts. 2003, « Supply chain management: design, coordination and operations », *Handbooks in Operations Research and Management Science*, Vol.11, Elsevier, 765 pp.

Oppen J.,A. Lokketangen, J. Desrosiers 2010: "Solveing a rich vehicle routing and inventory routing problem using column generation" *Computers and operation Research* 37 pp. 1308-1317.

Resende M.G.C, C.C. Ribeiro 2002 : Greedy randomized Adaptive Search Procedure. *State of the Art Handbook in Metaheuristics*, F.Glover and G.Kochenberger, eds., Kluwer

Savelsbergh M.W.P. and J.-H. Song 2008: An Optimization Algorithm for the Inventory Routing with Continuous Moves, *Computers and Operations Research* 35, pp. 2266-2282.

Solyali, O., H. Süral 2011: "A branch and cut Algorithm Using a Strong Formulation and an A Priori Tour Based Heuristic for an Inventory-Routing Problem" *Transportation Science* 45 n° 3 pp. 335-345.

Solyali, O., J-F. Cordeau, G. Laporte 2010: Robust Inventory Routing under Demand Uncertainty. *Submitted Paper.*

Resende M.G.C, C.C Ribeiro 2003: Greedy Randomized Adaptative Search Procedures. Handbook of Metaheuristics 57 pp. 219-249