RESEARCH ARTICLE | JUNE 21 2016

# A two-phase iterative procedure for the production, inventory and distribution routing problem FREE

Dicky Lim Teik Kyee; Noor Hasnah Moin



*AIP Conf. Proc.* 1750, 030032 (2016)
https://doi.org/10.1063/1.4954568

# A Two-Phase Iterative Procedure for The Production, Inventory and Distribution Routing Problem

Dicky Lim Teik Kyee[1, a)] and Noor Hasnah Moin [2, b)]

[1,2]*Institute of Mathematical Science, Faculty of Science, University of Malaya, 50603 Kuala Lumpur, Malaysia.*

[a)]Corresponding author: dickylim@siswa.um.edu.my
[b)]noor_hasnah@um.edu.my

**Abstract.** In this paper, the PIDRP is modelled as a one-to-many distribution system, in which a single warehouse or production facility is responsible for restocking a set of customers whose demands are deterministic and time-varying. The demand can be satisfied from either inventory held at the customer sites or from daily production. A fleet of homogeneous capacitated vehicles for making the deliveries is also considered. Capacity constraints for the inventory are given for each customer and the demand must be fulfilled on time, without delay. The aim of solving the PIDRP model is to minimize the overall cost of coordinating the production, inventory and transportation over a finite planning horizon. We propose an iterative procedure commonly known as MatHeuristic algorithm, an optimization algorithm designed by the interpolation of metaheuristics and mathematical programming techniques, to solve the model. In Phase 1, we construct routes in each period with the assumption that all the demands are satisfied in the given period by Variable Neighborhood Search, then the mixed integer programming is solved in Phase 2 to obtain the production schedules, quantity to be delivered and the inventory levels at the production facility and the customer sites. Based on the output from Phase 2, the routes are improved and the algorithm iterates until some stopping criteria is met. The model is solved by using Concert Technology of CPLEX 12.5 Optimizers with Microsoft Visual C++ 2010. Computational experiment is conducted to test the effectiveness of the algorithm. We observed that our algorithm performs better compared to the best integer solution obtained from CPLEX.



## INTRODUCTION

In the competitive business environment, many companies face problems with the inventory and distribution management. Thus, they keep searching for ways to design and manufacture new products, and distribute them in an efficient and effective manner. At the planning level, the goal is to coordinate production, inventory and delivery to meet customers demand so that the corresponding costs are minimized. Therefore, integrating production and distribution decisions is a challenging problem for manufacturers that are trying to optimize their supply chain. In general, the problem of optimally coordinating production, inventory and transportation is called the production-inventory-distribution routing problem (PIDRP) that is known to be NP-hard [1]. The PIDRP is sometimes known as production routing problems [2]. PIDRP is defined by a combination of a capacitated lot-sizing problem and a capacitated, multi-period vehicle routing problem (VRP). The PIDRP is also relevant to Vendor Managed Inventory (VMI) where the supplier (manufacturer) monitors the inventory at the retailers and coordinated efficient resource utilization to replenish the retailers.

## LITERATURE REVIEW

The first paper is due to Chandra [3] and Chandra and Fisher [4] and the authors show the benefit of coordinating the three components which result in 3-20% cost savings compared to sequentially solving the problems separately. Lei et al. [5] studied a multi-facility PIDRP with heterogeneous fleet and proposed a two-phase solution approach

*Advances in Industrial and Applied Mathematics*
AIP Conf. Proc. 1750, 030032-1–030032-7; doi: 10.1063/1.4954568
Published by AIP Publishing. 978-0-7354-1407-5/$30.00

where the problem is decomposed into phase 1 in which the model is solved as a mixed integer programming problem and the potential inefficiency of the direct shipment is solved in Phase 2 where the heuristic procedure is applied. An alternative approach using Greedy Randomized Adaptative Search Procedure (GRASP) is proposed by Boudia et al. [6]. The GRASP considered embeds improvement by a reactive tabu search algorithm for solving the problem. Similarly, Bard and Nananukul [7] developed a reactive tabu search algorithm for solving the PIDRP. They used an allocation model in the form of MIP to find good feasible solutions that were used as starting points for the tabu search. Path relinking was also used in post-processing phase to seek out marginal cost reduction.

Armentano et al. [8] extended the ideas in [6, 7] to include multiple products. The authors presented two heuristic approaches that allow trajectories with feasible and infeasible solutions. The first approach is a tabu search with short memory that uses a compound move at each iteration while the second approach makes use of path relinking that is integrated with tabu search. Besides, the approach was also tested on a set of single item instances [13] and they outperformed the memetic algorithm suggested by Boudia and Prins [9] and the reactive tabu search developed by Bard and Nananukul [7]. Adulyasak et al. [10] improves upon the results of Armentano et al. [8] by proposing Adaptative large neighborhood search heuristic to consider binary variables representing the setup and routing variables whilst the continuous variables associated with inventory, production and quantity delivered are handled by solving a network flow problem. The results outperformed all other known heuristics for PIDRP.

We refer the readers to the recent review of Adulyasak et al. [2] which gives a comprehensive state of art of PIDRP.

## PROBLEM DESCRIPTION AND MATHEMATICAL FORMULATION

We consider a PIDRP similar to the one proposed by [11]. The problem consists of a single production facility that produces a single product and distributes it to a set of $n$ customers with time varying and non-negative demands $d_{it}$ in each period and can be stored as the inventory by incurring the unit holding cost of the production facility as well as the customer sited. If production takes place at the facility in period $t$, the a setup cost $f_t$ is incurred for $t = 0,1, \dots, \tau$. In constructing delivery schedules, each customer can be visited at most once per period (split delivery is not allowed) and each of the $\theta$ homogenoues vehicles can make at most one trip per period. The initial inventories at the production facility and customers' sites are assumed to be zero.

Moreover, it is assumed that at the end of planning horizon all inventories are required to be zero. It is also assumed that all deliveries take place at the beginning of the period and arrive at the time to satisfy demand for at least that day. The number of vehicles is given and the total delivery quantity must not exceed vehicle capacity.

The notations used and mathematical formulation of the model follows the full model given in [11].

## SOLUTION METHODOLOGY

In this study, we propose a two-phase iterative procedure to solve the PIDRP model. We decompose the full model into two sub-problems, namely, a production and inventory control and a transportation solution (or vehicle routing) problem [12]. The production and inventory solution and the transportation solution are alternatively modified in an attempt to reduce the overall cost. In Phase 1, initial feasible routes for all periods are constructed by using zero-inventory (at the first iteration) and Giant Tour with Dijkstra's algorithm (in the following iterations). Then, we developed Variable Neighborhood Search (VNS) to generate a set of modified (or improved) routes. For the given set of specified routes, we solve mixed integer program (MIP1) in Phase 2 to obtain optimal allocations, inventories, and so on. The total cost of the solution is the sum of the objective function value of MIP1 except the cost term$\sum_{t\in N}\sum_{i\in N} w_{it}y_{it}$ and the transportation cost from the given routes. If the cost evaluated is better than the current best cost, we construct and improve the routes again from the solutions obtained from the MIP1 and iterates until a maximum iterations allowed is reached. The flow chart of the two-phase iterative procedure is outlined in Figure 1.

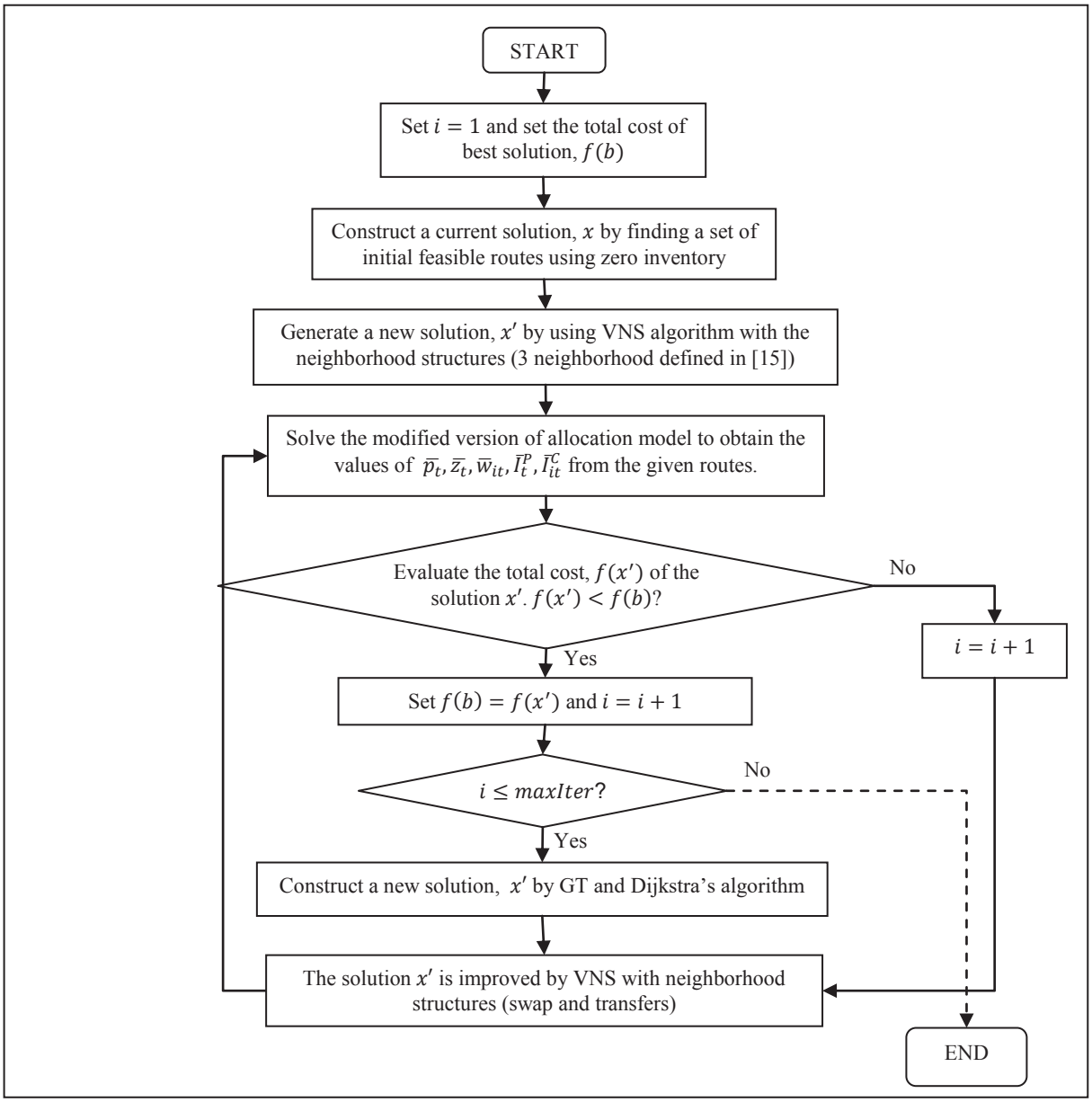


**FIGURE 1.** Flow chart of the two-phase iterative procedure

## Phase 1: Variable Neighborhood Search (VNS)

VNS was initially proposed by Mladenovic and Hansen [13] for solving combinatorial and global optimization problems. Let us denote a finite set of pre-selected neighborhood structures with $N_k$, where $k$=1,…,$k_{max}$, where $k_{max}$ refers to the maximum number of neighborhood used, and $N_k(x)$ the set of solutions in the $k$th neighborhood of x. The stopping condition may be maximum number of iterations, maximum number of iterations between two improvements or maximum CPU time allowed. There are three steps in the main VNS: Shaking, Local Search and Move or Not. The basic VNS heuristic comprises the steps is given in Figure 2.

**Initialization**. Step 0: Define a set of neighborhood structures $\mathcal{N}_k$, $k = 1, \ldots, k_{max}$, that will be used in the search and a set of local searches $\mathcal{R}_l$, $l = 1, \ldots, l_{max}$; generate an initial solution $x$;
choose a stopping condition;
**Repeat** the following steps until the stopping condition is met:
Step 1: Set $k \leftarrow 1$;
Step 2: Until $k = k_{max}$, repeat the following steps:
(a) **Shaking**. Generate a point $x'$ at random from the $k^{th}$ neighborhood of $x$ ($x' \in N_k(x)$)
(b) **Local Search**. Apply some local search method with $x'$ as initial solution; denote with $x$" the so obtained local optimum;
(c) **Move or not**. If this local optimum is better than the incumbent, move there ($x \leftarrow x$"), and go back to (1); otherwise, set $k \leftarrow k + 1$.

**FIGURE 2.** Steps of basic VNS

*Initialization (Step 0)*

The initial solution (i.e. current solution, x) is obtained in two steps; (a) construct a giant tour using sweep algorithm [14] and (b) find the corresponding optimal fleet size by constructing cost network and subsequently applying Dijkstra's algorithm which provides an initial feasible solution that contains routes. At the first iteration, since there are no allocations, we construct the routes by zero-inventory, which the delivery quantities are exactly match the demand. Note that no inventory holding cost will be incurred.

*Shaking Step (Step 2(a))*

A solution $x'$ is picked randomly from the $k$th neighborhood of $x$. This will ensure that the solution is not far from the current best solution, $x$. We consider four moves: forward transfer, backward transfer, swap and transfer for VNS. The algorithm of the shaking step is shown in Figure 3.

```
Set num=1
While (num<=k) do    //the number of changes depend on the value of k
{
        Randomly generate the value of r where 0<r<1. Define p1, p2 and p3
                if (r<=p1)     // the value of p1 represent the chosen probability
                    Apply forward transfer
                else if (p1<r<=p2)
                    Apply backward transfer
                else if(p2<r<=p3)
                    Apply swap
                else
                     Apply transfer
                end if
                num=num+1
}
```

**FIGURE 3.** The algorithm of shaking step

Forward transfer aimed at reducing the inventory holding cost of the customer by transferring the maximum portion of delivery quantities that can be reassigned to the succeeding period without causing a shortage in the current period while backward transfer is opposite of forward transfer, which aimed at reducing the transportation cost by transferring the full amount of delivery quantities to the preceding period. Swap involves an exchange of delivery quantities between two customers $i_1$ in period $t_1$ and $i_2$ in period $t_2$, where $t_2$ is the first period after $t_1$ in which $w_{i_2 t_2} > 0$. In general, a swap produces a change in holding costs and a change in holding costs in both periods $t_1$ and $t_2$. Transfer is similar to backward transfer but we limit the number of periods to be inserted to at least two preceding periods, where $t_2 = \max\{t: t < t_1, \exists \bar{w}_{it} > 0 \text{ for some } i \in N\}$.

Note that in the first iteration, since the inventory at the customer sites is all equal to zero and the production quantity and inventory at the production facility are not considered, we use the three neighborhood structures (used in VRP) defined in Imran et al. [15] with $k_{max} = 3$ in the shaking step. These include the 1-1 lambda interchange as $\mathcal{N}_1$, the 2-0 shift as $\mathcal{N}_2$ and the 2-1 interchange as $\mathcal{N}_3$.

We also incorporate the concept of tabu search which forbid the movement of the customer for a few iterations if the customer has been chosen to transfer or swap. In all the four moves, only moves that result in feasible solutions are allowed so it is necessary to check for violations of the production constraints, inventory bounds as well as the vehicle capacity constraint.

*Local Search (Step 2(b))*

The local search consists of six refinement procedures adopted from [15]. The order of the refinement procedure is as follows: the 1-insertion inter-route as $\mathcal{R}_1$, the 2-opt inter-route as $\mathcal{R}_2$, the 2-opt intra route as $\mathcal{R}_3$, the swap intra route as $\mathcal{R}_4$, 1-insertion intra-route as $\mathcal{R}_5$ and at last, the 2-insertion intra-route as $\mathcal{R}_6$. The process starts by generating a random feasible solution $x'$ from $\mathcal{N}_1$, which is used as a temporary solution. The multi-level approach then starts by finding the best solution $x''$ using $\mathcal{R}_1$. If $x$" is better than $x'$, then replace $x'$ by $x$" and the search return to $\mathcal{R}_1$, otherwise the next refinement procedure $\mathcal{R}_2$ is applied. This process is repeated until $\mathcal{R}_6$ cannot produce a better solution.

## Phase 2: Mixed Integer Program

After the routes are constructed and improved by VNS, we can obtain $y_{it}$ from the specified (improved) routes. Suppose $y_{it} = 1$ if the customer $i$ is visited in period $t$, and 0 otherwise. Note that $y_{it}$ is a parameter which is specified by the given routes. The mixed integer program is presented as follows:

$$\phi_{IP} = \min \sum_{t\in T} fz_t + \sum_{t\in T}\sum_{i\in N} w_{it} y_{it} + \sum_{t\in T_0\setminus\{\tau\}} h^P I_t^P + \sum_{t\in T\setminus\{\tau\}}\sum_{i\in N} h_i^C I_{it}^C \tag{1}$$

Subject to:

$$I_t^P = I_{t-1}^P + p_t - \sum_{i\in N} w_{it}\,, \qquad \forall \mathrm{t} \in \mathrm{T}_0 \tag{2}$$

$$I_{it}^C = I_{i,t-1}^C + w_{it} - d_{it}\,, \qquad \forall i \in N, t \in T \tag{3}$$

$$\sum_{i\in N} w_{it} \le I_{t-1}^P, \qquad \forall t \in T_0 \tag{4}$$

$$p_t \le C z_t, \qquad \forall t \in T_0 \setminus \{\tau\} \tag{5}$$

$$p_0 \ge \sum_{i\in N} (d_{i1} - I_{i0}^C) \tag{6}$$

$$\sum_{i\in N} w_{it} \le 0.8QK, \qquad \forall t \in T \tag{7}$$

$$0 \le I_t^P \le I_{\max}^P,\ 0 \le I_{it}^C \le I_{\max i}^C\,,\ \forall i \in N, t \in T \setminus \{\tau\},\ I_\tau^P = I_{i\tau}^P = 0,\ \forall i \in N \tag{8}$$

$$z_t \in \{0,1\},\ p_t \ge 0,\ y_{it} \in \{0,1\},\ w_{it} \ge 0 \text{ and integer},\ \forall i \ne j \in N_0, t \in T \tag{9}$$

**MIP1**

The optimal production quantity, inventory at both production facility and customer sites and the delivery quantity can be obtained after the model MIP1 is solved.

## COMPUTATIONAL RESULT

All the algorithms are written in Microsoft Visual C++ 2010 and performed on a 3.1 GHz processor with 8GB of RAM. The code for the mixed integer program is written in Concert Technology of Microsoft Visual Studio 2010 linked to the CPLEX 12.5 libraries. CPU times were obtained the time function in C++.

For the algorithm testing, we used a data set provided by Moin et al. [16] consisting of 14 data sets comprises of 12, 20, 50 and 100 customers with 5, 10, 14 and 21 periods. The locations of the customers are generated randomly on a $100 \times 100$ Euclidean grid. All of the data sets have demands in every period, with exception for case 50 customers. The holding cost for each customer is generated within the range [1,10] and the demands are generated randomly within the range [0, 50]. The vehicle capacity is fixed as 100 and the depot is located at (0,0) for all data sets.

**TABLE 1.** Results from randomly generated data sets

| Case | CPLEX | | | MatHeuristic | | |
|---|---|---|---|---|---|---|
| | Best Integer | Time (s) | Number Of Vehicles | Total Cost | Time (sec) | Number Of Vehicles |
| S12T5 | 4416.85 | 929.41 | 19 | **3953.69** | 31.287 | 21 |
| S12T10 | 8674.45 | 1212.55 | 36 | **7374.32** | 52.251 | 39 |
| S12T14 | 11814.85 | 3600.35 | 51 | **11101.92** | 74.885 | 56 |
| S20T5 | 6882.51 | 1194.95 | 28 | **6342.22** | 35.954 | 31 |
| S20T10 | 14280.74 | 1401.64 | 58 | **12760.1** | 89.126 | 61 |
| S20T14 | 19448.52 | 897.93 | 80 | **17496.5** | 154.563 | 84 |
| S20T21 | 26705.26 | 1989.7 | 118 | 29125.26 | 287.265 | 121 |
| S50T5 | 12006.72 | 2881.62 | 52 | 13066.72 | 206.243 | 53 |
| S50T10 | 31335.94 | 713.58 | 114 | **28369.9** | 630.756 | 119 |
| S50T14 | 45082.01 | 830.42 | 161 | **44879.8** | 1408.35 | 197 |
| S50T21 | 117350.35 | 1056.38 | 239 | **65787.2** | 3632.7 | 300 |
| S100T5 | 36017.92 | 3352.85 | - | **26208.7** | 1304.43 | 132 |
| S100T10 | - | - | - | 54502.5 | 14146 | 275 |
| S100T14 | - | - | - | 85561.6 | 34871.8 | 462 |

Table 1 shows the results (best cost), the corresponding number of vehicles and the CPU time. Column 2, 3 and 4 give the solution of best integer from CPLEX, the computational time and the corresponding number of vehicles. The last three columns display the total cost, its computational time and the number of vehicles that obtained from our algorithm. CPLEX 12.5 is allowed to run for 10800 seconds (3 hours) but in the large cases such as S100T10 and S100T14, CPLEX failed to obtain the best integer solution due to running out of memory. The CPLEX solutions are also used as guidelines in order to ensure that our solutions are correct. The total costs (in bold) indicate our algorithm outperforms the CPLEX solutions.



## CONCLUSION

In this paper, we propose a two-phase iterative procedure to solve the production-inventory-distribution routing problem (PIDRP) model. We utilize both phases in an interactive manner. In Phase 1, the initial feasible solution is obtained by using zero-inventory in the first iteration and the initial routes are constructed using Giant Tour and Dijkstra's algorithm. Phase 2 solves the mixed integer program with the information on the routes constructed in Phase 1 to get optimal delivery quantities and inventories. These informations are used in the VNS where it incorporates several inventory updating mechanisms to improve the results from Phase 2. The algorithm iterates until a maximum iteration allowed is reached. Computational experiment is conducted to test the effectiveness of the algorithm. The CPLEX solutions are used as guidelines in order to ensure that our solutions are correct and we observe that our algorithm perform better than the best integer solution from CPLEX in most of the cases.

## ACKNOWLEDGEMENT

This research received support from University of Malaya Research Grant UMRG RG116/10AFR and the first author would like to acknowledge the support from MyMaster programme under Department of Higher Education, Ministry of Education.

## REFERENCES


1. J. F. Bard and N. Nananukul, Computers & Operations Research **37** (12), 2202-2217 (2010).
2. Y. Adulyasak and M. M. Dessouky, Computers & Operations Research **55**, 141-152 (2015).
3. P. Chandra, Journal of the Operational Research Society, **44** (7), 681-692 (1993).
4. P. Chandra and M. Fisher, European Journal of Operational Research, **72** (3), 503-517 (1994).
5. L. Lei, S. Liu, A. Ruszczynski and S. Park, IIE Transactions on Scheduling & Logistics, **38** (11), 955-970 (2006).
6. M. Boudia, M. A. O. Louly, and C. Prins, Computers & Operations Research, **34** (11), 3402-3419 (2007).
7. J. F. Bard and N. Nananukul, Journal of Scheduling, **12** (3), 257-280 (2009).
8. V. A. Armentano, A. L. Shiguemoto and A. Lokketagen, Computers & Operations Research, **195**, 703-715 (2009).
9. M. Boudia and C. Prins, Computers & Operations Research, **38** (8), 1199-1209 (2011).
10. Y. Adulyasak, J. -F. Cordeau and R. Jans, INFORMS Journal on Computing, **26** (1), 103-120 (2014).
11. N. Nananukul, "Lot-sizing and inventory routing for a production-distribution supply chain", PhD dissertation, The University of Texas, Austin, 2008.
12. C-G. Lee, Y.A. Bozer and C.C. White III, "A heuristic approach and properties of optimal solutions to the dynamic inventory routing problem", Working paper, 2003.
13. N. Mladenovic and P. Hansen, Computers & Operations Research, **24**, 1097-1100 (1997).
14. B.E. Gillett and L.R. Miller, Operations Research, **22**, 1974, 340-344 (1974).
15. A. Imran, S. Salhi and N.A. Wassan, European Journal of Operational Research, **197** (2), 509-518 (2009).
16. N.H. Moin and Y. Titi, Mathematical Problems in Engineering, 2015, 1-11 (2015).