Adaptive GVNS heuristics for solving the Pollution Location Inventory Routing Problem

Panagiotis Karakostas[1][0000−0002−1936−8468], Angelo Sifaleras[2][0000−0002−5696−7021], and Michael C. Georgiadis[1][0000−0002−2016−5131]

[1] Department of Chemical Engineering, Aristotle University of Thessaloniki, University Campus, 54124 Thessaloniki, Greece,
pkarakost@cheng.auth.gr, mgeorg@auth.gr
[2] Department of Applied Informatics, School of Information Sciences, University of Macedonia, 156 Egnatia Str., Thessaloniki 54636, Greece,
sifalera@uom.gr

Abstract. This work proposes Adaptive General Variable Neighborhood Search metaheuristic algorithms for the efficient solution of Pollution Location Inventory Routing Problems (PLIRPs). A comparative computational study, between the proposed methods and their corresponding classic General Variable Neighborhood Search versions, illustrates the effectiveness of the intelligent mechanism used for automating the re-ordering of the local search operators in the improvement step of each optimization method. Results on 20 PLIRP benchmark instances show the efficiency of the proposed metaheuristics.

Keywords: Adaptive General Variable Neighborhood Search · Intelligent Optimization Methods · Pollution Location Inventory Routing Problem · Green Logistics.

1 Introduction

The Pollution Location Inventory Routing Problem (PLIRP) is an NP-hard combinatorial optimization problem, which involves both economic and environmental decisions [7]. It simultaneously addresses strategic decisions, such as the location of candidate depots and the allocation of customers to the opened depots, tactical decisions, as the inventory levels and the replenishment rates and finally operational decisions, such as routing schedules. The objective of this problem is the minimization of the total cost, which consists of facilities opening costs, inventory control costs, general routing costs and fuel consumption costs.

It should be mentioned that, there are several factors affecting the fuel consumption. The main factors are the speed, the acceleration of the vehicle, the traveled distance and the total weight of the vehicle, consisting of the curb and freight weight [2].

In this work, several Adaptive General Variable Neighborhood Search (AGVNS) heuristic algorithms have been developed for the efficient solution of recently proposed PLIRP instances [7]. The proposed AGVNS schemes are compared

Please cite this paper as:

Karakostas P., Sifaleras A., and Georgiadis M. C., "Adaptive GVNS heuristics for solving the Pollution Location Inventory Routing Problem", in N. Matsatsinis, Y. Marinakis, and P. M. Pardalos (Eds.), Learning and Intelligent Optimization. LION13. Lecture Notes in Computer Science, Vol. 11968, pp. 157-170, 2020.

The final publication is available at Springer via https://doi.org/10.1007/978-3-030-38629-0_13

both with their corresponding GVNS methods as well as with the only available heuristic algorithm in the literature for this problem variant. The remainder of this paper is organized as follows: Section 2 provides the mathematical formulation of the problem. Section 3 describes the developed solution approaches and their algorithmic details, while Section 4 provides extensive numerical analyses for testing the efficiency of the proposed methods on 20 PILRP instances. Finally, Section 5 draws up main concluding remarks and highlights direction for future work.

2 Problem Statement

For the readers clarity sake, the mathematical formulation of the problem is presented in this section. Model notations are summarized in Tables 1, 2 and 3.

Table 1. Model sets.

Indices & Explanation
V is the set of nodes
J is the set of candidate depots
I is the set of customers
H is the set of discrete and finite planning horizon
K is the set of vehicles
R is the set of speed levels

Table 2. Model decision variables.

Notation	**Explanation**
y_j	1 if j is opened; 0 otherwise
z_{ij}	1 if customer i is assigned to depot j; 0 otherwise
x_{ijkt}	1 if node j is visited after i in period t by vehicle k
q_{ikt}	product quantity delivered to customer i in period t by vehicle k
w_{itp}	quantity delivered to customer i in period p to satisfy its demand in period t
a_{vikt}	load weight by travelling from node v to the customer i with vehicle k in period t
$zz_{v_1 v_2 ktr}$	1 if vehicle k travels from node $v1$ to $v2$ in period t with speed level r

Table 3. Model parameters.

Notation	Explanation	Value
f_j	fixed opening cost of depot j	Instance-depended
C_j	storage capacity of depot j	Instance-depended
h_i	unit inventory holding cost of customer i	Instance-depended
Q_k	loading capacity of vehicle k	Instance-depended
d_{it}	period variable demand of customer i	Instance-depended
c_{ij}	travelling cost of locations pair (i, j)	Instance-depended
s_r	the value of the speed level r	Instance-depended
ϵ	fuel-to-air mass ratio	1
g	gravitational constant (m/s^2)	9.81
ρ	air density (kg/m^3)	1.2041
CR	coefficient of rolling resistance	0.01
η	efficiency parameter for diesel engines	0.45
f_c	unit fuel cost $(/L)$	0.7382
f_e	unit CO_2 emission cost $(/kg)$	0.2793
σ	CO_2 emitted by unit fuel consumption (kg/L)	2.669
$HVDF$	heating value of a typical diesel fuel (kj/g)	44
ψ	conversion factor (g/s to L/s)	737
θ	road angle	0
τ	acceleration (m/s^2)	0
CW_k	curb weight (kg)	3500
EFF_k	engine friction factor $(kj/rev/L)$	0.25
ES_k	engine speed (rev/s)	39
ED_k	engine displacement (L)	2.77
CAD_k	coefficient of aerodynamics drag	0.6
FSA_k	frontal surface area (m^2)	9
$VDTE_k$	vehicle drive train efficiency	0.4

The following utilization of formulas simplifies the fuel consumption components of the objective function: $\lambda = \frac{HVDF}{\psi}$, $\gamma_k = \frac{1}{1000VDTE\eta}$, $\alpha = \tau + gCR\sin\theta + gCR\cos\theta$ and $\beta_k = 0.5CAD\rho FSA_k$.

$$\min \sum_{j\in J} f_j y_j + \sum_{i\in I} h_i \sum_{t\in H}\left(\frac{1}{2}d_{it} + \sum_{p\in H, p<t} w_{itp}(t-p) + \sum_{p\in H, p>t} w_{itp}(t-p+|H|)\right)$$
$$+ \sum_{i\in V}\sum_{j\in V}\sum_{t\in H}\sum_{k\in K} c_{ij}x_{ijkt} + \sum_{i\in V}\sum_{j\in V}\sum_{k\in K}\sum_{t\in H}\left\{\lambda\left(f_c + (f_e\sigma)\right)\left(\sum_{r\in R}\frac{(zz_{ijktr}EFF_kES_kED_kc_{ij})}{s_r}\right.\right.$$
$$\left.\left.+\left(\alpha\gamma_k\left(CW_kx_{ijkt} + a_{ijkt}\right)c_{ij}\right) + \left(\beta_k\ \gamma_k \sum_{r\in R}\left(s_r\ zz_{ijktr}\right)^2\right)\right)\right\} \tag{1}$$

Subject to

$$\sum_{r\in R} zz_{ijktr} = 0 \quad \forall i, j \in V, \forall k \in K, \forall t \in H \tag{2}$$

$$\sum_{i\in V} a_{ijkt} - \sum_{i\in V} a_{jikt} = q_{jkt}PW \quad \forall j \in I, \forall k \in K, \forall t \in H \tag{3}$$

$$\sum_{j\in V} x_{ijkt} - \sum_{j\in V} x_{jikt} = 0 \qquad \forall i \in V, \forall k \in K,\ \forall t \in H \tag{4}$$

$$\sum_{j\in V}\sum_{k\in K} x_{ijkt} \le 1 \quad \forall t \in H,\ \forall i \in I \tag{5}$$

$$\sum_{j\in V}\sum_{k\in K} x_{jikt} \le 1 \quad \forall t \in H,\ \forall i \in I \tag{6}$$

$$\sum_{i\in I}\sum_{j\in J} x_{ijkt} \leq 1 \quad \forall k \in K,\ \forall t \in H \tag{7}$$

$$x_{ijkt} = 0 \quad \forall i,j \in J,\ \forall k \in K,\ \forall t \in H,\ i \neq j \tag{8}$$

$$\sum_{i\in I} q_{ikt} \leq Q_k \quad \forall k \in K,\ \forall t \in H \tag{9}$$

$$\sum_{j\in J} z_{ij} = 1 \quad \forall i \in I \tag{10}$$

$$z_{ij} \leq y_j \quad \forall i \in I,\ \forall j \in J \tag{11}$$

$$\sum_{i\in I}\left(z_{ij}\sum_{t\in H} d_{it}\right) \leq C_j \quad \forall j \in J \tag{12}$$

$$\sum_{u\in I} x_{ujkt} + \sum_{u\in V\setminus\{i\}} x_{iukt} \leq 1 + z_{ij} \quad \forall i \in I,\ \forall j \in J,\ \forall \mathrm{k} \in \mathrm{K}, \forall t \in H \tag{13}$$

$$\sum_{i\in I}\sum_{k\in K}\sum_{t\in H} x_{jikt} \geq y_j \quad \forall j \in J \tag{14}$$

$$\sum_{i\in I} x_{jikt} \leq y_j \quad \forall j \in J,\ \forall k \in K,\ \forall t \in H \tag{15}$$

$$\sum_{p\in H} w_{itp} = d_{it} \quad \forall i \in I,\ \forall \mathrm{t} \in \mathrm{H} \tag{16}$$

$$\sum_{t\in H} w_{itp} = \sum_{k\in K} q_{ikp} \quad \forall i \in I,\ \forall p \in H \tag{17}$$

$$q_{ikt} \leq M\sum_{j\in V} x_{ijkt} \quad \forall i \in I,\ \forall t \in H,\ \forall k \in K \tag{18}$$

$$\sum_{j\in V} x_{ijkt} \leq M q_{ikt} \quad \forall i \in I,\ \forall t \in H,\ \forall \mathrm{k} \in \mathrm{K} \tag{19}$$

$$\mathrm{x}_{ijkt} \in \{0,1\} \quad \forall i \in I,\ \forall j \in J,\ \forall t \in H,\ \forall k \in K \tag{20}$$

$$y_j \in \{0,1\} \quad \forall j \in J \tag{21}$$

$$z_{ij} \in \{0,1\} \quad \forall i \in I,\ \forall j \in J \tag{22}$$

$$q_{ikt} \leq \min\left\{Q_k, \sum_{p \in H} d_{ip}\right\} \quad \forall i \in I,\ \forall j \in J\ ,\forall k \in K \tag{23}$$

$$w_{itp} \leq d_{ip} \quad \forall i \in I,\ \forall t, p \in H \tag{24}$$

The objective goal of the problem is the minimization of the total cost which consists of facilities opening costs, inventory holding costs, general routing costs and fuel consumption and CO_2 emissions costs. The constraints of the model can be grouped into routing-, inventory- and location-related constraints. For example, the routing-related set of Constraints 8 prevent vehicles from traveling between two depots in each time period, while the location-related set of Constraints 10 guarantee that a customer will be served by exactly one depot. Also, an example of inventory-related constraints are the set of Constraints 9 which they ensure that the delivered amount of product quantities will not exceed the capacity of the used vehicle in a specific time period.

3 Solution Approach

3.1 Initialization

A three-phase construction heuristic has been developed in order to build quick initial feasible solutions for the PLIRP. Location and allocation decisions are determined in the first phase and inventory-routing decisions are made in the second one. Finally, the speed levels for travelling through the nodes of a network are selected.

More specifically, the opening of the required depots is based on a ratio-based selection technique. For each candidate depot, the ratio $\frac{fixed\_opening\_cost}{Capacity}$ is calculated, where the "fixed_opening_cost" represents the cost for opening each depot, and the "capacity" is the maximum amount of product units that the selected depot can offer. The depot with the minimum ratio is selected to be opened. The number of the opened depots depends on their capacities and the total demand of customers. The allocation of customers to the opened depots is made in a serial way. Serial allocation means that for a selected depot, the set of customers is being ran and the first unallocated customer, whose total demand does not exceed the remaining capacity of the depot, is assigned to that depot. The allocation process is completed when all customers have been assigned to the opened depots.

In order to determine the inventory-routing decisions, a Random Insertion method [3] is applied for building the vehicles routes and each customer receives the demanded quantity of product in each time period. The selection of speed levels is performed randomly.

3.2 Neighborhood Structures

Three neighborhood structures are proposed for the efficient exploration of the solution space, as follows:

Inter-route Relocate: In this local search operator two customers (i and b) from different routes are selected. The Inter-route Relocated removes i from his route R_i and moves him in the route servicing customer b, R_b in the next position of b, in each time period. Both customers can be serviced by the same depot or by different depots over the time horizon. A replenishment shifting may be performed in order to avoid vehicle capacity violation in route R_b.

Exchange Opened-Closed Depots: In this neighborhood a closed depot i is being exchanged with an opened one j. The capacity of the depot i must be at least equal to the capacity of j for having a valid move. Also, a reordering of the routes allocated on depot j may occur according to the minimum insertion cost criterion of depot i.

2-2 Replenishment Exchange: Two time periods t_1 and t_2 are randomly selected in this operator, and then, the two most distant customers i and b are found. Both of those customers must be serviced in t_1 and t_2. The cost changes of removing i and b from their routes in periods t_1 and t_2 respectively and shifting their receiving deliveries from t_1 to t_2 for customer i and from t_2 to t_1 for b are calculated. The move is applied only if no violations over the vehicles capacities occurred.

3.3 Shaking Scheme

The main scope of a shaking procedure is to help the algorithm escaping from local optimum solutions [5]. In each shaking iteration, a new random solution S' is obtained by a randomly selected neighborhood from a predefined set of neighborhoods and according to a given solution S. In this work, an intensified shaking method has been developed with two neighborhood structures, the Exchange Opened-Closed Depots and the Intra-route Relocate.

The Exchange Opened-Closed Depots operator is applied as described in subsection 3.2. The Intra-route Relocate operator removes a randomly selected customer from its current position in its route and moves him in a different position in the same route. The pseudo-code of this diversification method is summarized in Algorithm 1. The Shake function receives an incumbent solution S and the number of iterations k (where $1 < k < k_{max}$ and $k_{max} = 12$), which indicates the times that, one randomly selected neighborhood operator (of the two in total) will be applied for generating a new solution S'.

3.4 Adaptive General Variable Neighborhood Search

Variable Neighborhood Descent (VND). The VND is the deterministic variant of the well-known metaheuristic framework Variable Neighborhood Search (VNS). In a VND method the local search operators are ordered in a specific sequence and applied successively until no more improvements can be noticed.

Algorithm 1 Shaking Procedure

```
procedure SHAKE(S, k)
    l = random_integer(1, 2)
    for i ← 1, k do
        select case(l)
        case(1)
        S′ ← Intra − route_Relocate(S)
        case(2)
        S′ ← Exchange_OpenedClosed_Depots(S)
        end select
    end for
    Return S′
end procedure
```

According to the neighborhood change step, there are different VND schemes. Two of the most well-known VND schemes are the cyclic VND (cVND) and the pipe VND (pVND). In the first one, the search continuous in the next neighborhood in the set independently of the improvement criterion, while in the pVND the exploration continuous in the same neighborhood while an improvement is occurred [5]. In this work both cVND and pVND are used. Moreover, it should be mentioned that the parameter l_{max} in both VND pseudo-codes denotes the number of the used neighborhood structures. The pseudo-codes of the proposed VND schemes are given in Algorithms 2 and 3.

Algorithm 2 cyclic VND

```
procedure cVND(S, l_max)
    l = 1
    while l ≤ l_max do
        select case(l)
        case(1)
        S′ ← Inter_Relocate(S)
        case(2)
        S′ ← Exchange_OpenedClosed_Depots(S)
        case(3)
          S′ ← 2 − 2ReplenishmentExchange(S)
          end select
          if f(S′) < f(S) then
              S ← S′
              l = l + 1
          else
              l = l + 1
          end if
    end while
    Return S
end procedure
```

The local search operators are applied with an adaptive search strategy, which combines the first and best improvement search strategies [8]. More specifically, if the number of customers in a problem instance is more than 90, the fist improvement search strategy is applied, otherwise the algorithm explores the neighborhoods with the best improvement strategy.

General Variable Neighborhood Search (GVNS). The GVNS is a variant of the VNS, which combines deterministic and stochastic components during the search. More specifically, it adopts one of the VND schemes as its main im-

Algorithm 3 pipe VND

```
procedure pVND(S, l_max)
    l = 1
    while l ≤ l_max do
        select case(l)
        case(1)
        S' ← Inter_Relocate(S)
        case(2)
        S' ← Exchange_OpenedClosed_Depots(S)
        case(3)
        S' ← 2 − 2ReplenishmentExchange(S)
        end select
        if f(S') < f(S) then
            S ← S'
        else
            l = l + 1
        end if
    end while
    Return S
end procedure
```

provement step [5, 10]. Based on the two proposed VND methods, two GVNS schemes are shaped and provided in the following pseudo-codes.

Algorithm 4 $GVNS_{cVND}$

```
procedure GVNS_cVND(S, k_max, l_max, max_time)
    while time ≤ max_time do
        S* = Shake(S, k)
        S' = cVND(S*, l_max)
        if f(S') < f(S) then
            S ← S'
        end if
    end while
    return S
end procedure
```

Algorithm 5 $GVNS_{pVND}$

```
procedure GVNS_pVND(S, k_max, l_max, max_time)
    while time ≤ max_time do
        S* = Shake(S, k)
        S' = pVND(S*, l_max)
        if f(S') < f(S) then
            S ← S'
        end if
    end while
    return S
end procedure
```

Adaptive mechanism. The order of the neighborhood structures is a crucial component for the successful performance of a VNS-based algorithm [4, 6]. Consequently, it is important to employ an intelligent mechanism for the re-ordering of the neighborhood structures. According to the literature, some adaptive variants of the VNS have been proposed for that case. Todosijevic et al. [12] proposed an Adaptive GVNS in which a re-ordering of the neighborhoods is applied in each iteration based on their success in previous solution process. Li and Tian [9] also, proposed an adaptive version of VNS in which a probabilistic selection mechanism is used for deciding the sequence of the neighborhoods. In this work, an adaptive neighborhoods re-ordering mechanism is proposed. In each iteration, the sequence of the neighborhoods is re-formed based on the number of improvements in the previous iteration. The parameter "Improvements_Counter" is an array and its positions keep the improvements achieved by each neighborhood structure. The initial order is based on the complexity of each local search operator. Thus, the initial order is the following:

1. Inter-route Relocate.
2. Exchange Opened-Closed Depots.
3. 2-2 Replenishment Exchange.

The same order is adopted any time all neighborhoods are unable to provide any improved solution. The adaptive mechanism is summarized in Algorithm 6.

Algorithm 6 Adaptive_Order

```
procedure ADAPTIVE_ORDER(N_Order, Improvements_Counter)
    if no improvement is found in any neighborhood then
        Keep the same order
    end if
    if an improvement is found then
        New_N_Order ← Descending_Order(N_Order, Improvements_Counter)
    end if
    N_Order ← New_N_Order
    return N_Order
end procedure
```

The pseudo-codes of the Adaptive GVNS schemes, proposed in this work, are provided in Algorithms 7 and 8.

Algorithm 7 $AGVNS_{cVND}$

```
procedure AGVNS_cVND(S, k_max, l_max, max_time, N_Order, Improvements_Counter)
    while time ≤ max_time do
        S* = Shake(S, k)
        N_Order ← Adaptive_Order(N_Order, Improvements_Counter)
        S' = cVND(S*, l_max)
        if f(S') < f(S) then
            S ← S'
        end if
    end while
    return S
end procedure
```

Furthermore, it is examined if the initial order of the neighborhoods affects the performance of the AGVNS schemes. Consequently, an alternative of the previous mentioned adaptive mechanism is applied, which it uses random re-ordering either in the first iteration or each time no improvements achieved through the VND methods. The random re-ordering is achieved by applying a shuffle method over the neighborhoods order set.

Algorithm 8 $AGVNS_{pVND}$

```
procedure AGVNS_pVND(S, k_max, l_max, max_time, N_Order, Improvements_Counter)
    while time ≤ max_time do
        S* = Shake(S, k)
        N_Order ← Adaptive_Order(N_Order, Improvements_Counter)
        Improvements_Counter ← 0
        S' = pVND(S*, l_max)
        if f(S') < f(S) then
            S ← S'
        end if
    end while
    return S
end procedure
```

4 Computational Results

4.1 Computer Environment & Benchmark Instances

The proposed algorithms were implemented in Fortran. They ran using Intel Fortran compiler 18.0 with optimization option /O3 on a desktop PC running Windows 7 Professional 64-bit with an Intel Core i7-4771 CPU at 3.5 GHz and 16 GB RAM. The parameter k_{max} was set at 12 and the maximum execution time for each VNS-based algorithm is $60s$.

The benchmark instances used for testing the efficiency of the proposed algorithms was initially proposed in [7] and can be found at: `http://pse.cheng.auth.gr/index.php/publications/benchmarks`. The form of each instance name is $X - Y - Z$, where X is the number of candidate depots, Y the number of customers and Z the number of time periods.

4.2 Numerical Analysis

In Table 4, the AGVNS1 represents an adaptive GVNS scheme with the complexity-based initial neighborhoods order, while the AGVNS2 represents the scheme with the random initial order.

Table 4. Computational results of the proposed methods

Instance	$GVNS_{CVND}$	$AGVNS1_{CVND}$	$AGVNS2_{CVND}$	$GVNS_{PVND}$	$AGVNS1_{PVND}$	$AGVNS2_{PVND}$
4-8-3	23069.96	22944.88	22936.8	22937.11	22935.84	**22935.53**
4-8-5	19829.12	19693.68	19731.48	19489.22	**19373.37**	19475.87
4-10-3	17415.26	17452	17697.38	17612.43	**17604.76**	17630.7
4-10-5	23954.56	**23932.88**	23968.77	23944.69	23937.55	23946.91
4-15-5	22115.26	22090.42	22270.71	**22049.63**	22068.88	22174.31
5-9-3	18496.52	18488.41	18570.3	18496.69	**18431.97**	18456.86
5-12-3	24752.46	24747.32	24748.96	24738.99	**24735.39**	24741.45
5-15-3	17502.49	**17469.69**	17495.32	17487.35	17481.19	17479.19
5-18-5	19485.86	19358.93	19312.34	19082.68	**19048.77**	19079.43
5-20-3	17238.63	**17082.93**	17202.05	17248.24	17159.07	17179.59
6-22-7	20042.48	19998.92	**19982.33**	20008.2	19998.96	20023.52
6-25-5	22031.12	21877.13	21864.83	21877.2	21738.51	**21706.41**
7-25-5	29958.96	29798.39	29965.02	29226.52	29183.52	**29173.52**
7-25-7	22559.16	**22239.8**	22366.23	22273.87	22253.14	22331.7
8-25-5	19103.74	19029.96	19178.87	18982.77	**18925.32**	19099.61
8-30-7	20714.09	22706.62	20653.14	20594.12	**20454.58**	20544.96
8-50-5	23106.05	23054.47	23097.66	22922.63	**22365.3**	22533.46
8-65-7	26419.61	**23980.33**	24985.91	27288.72	25496.35	26857.53
9-40-7	21456.76	20908.32	**20860.35**	21243.43	20996.73	20929.3
9-55-5	23254.2	**22563.17**	22734.83	23296.72	22754.5	22703.21
Average	21625.57	21470.91	21481.16	21540.06	**21347.19**	21450.45

The results illustrate that all the Adaptive GVNS perform better than the classic GVNS schemes. However, the $AGVNS1_{pVND}$ is the method which provides the best solutions in average. The $AGVNS2_{pVND}$ is ranked as the second method and the methods $AGVNS1_{CVND}$ and $AGVNS2_{CVND}$ hold the third and the fourth place respectively. The $GVNS_{pVND}$ takes the fifth place and the last one is the $GVNS_{cVND}$.

Karakostas et al. [7], recently proposed a Basic Variable Neighborhood Search (BVNS) heuristic algorithm for solving the tested PLIRP instances. Their algorithm used two local search operators, the Inter-route Exchange and the Exchange Opened-Closed Depots.

Figure 1 illustrates the performance of the $AGVNS1_{pVND}$ and the BVNS on the 20 PLIRP instances. It is clear that the $AGVNS1_{pVND}$ outperforms the BVNS algorithm, especially on larger PLIRP instances.

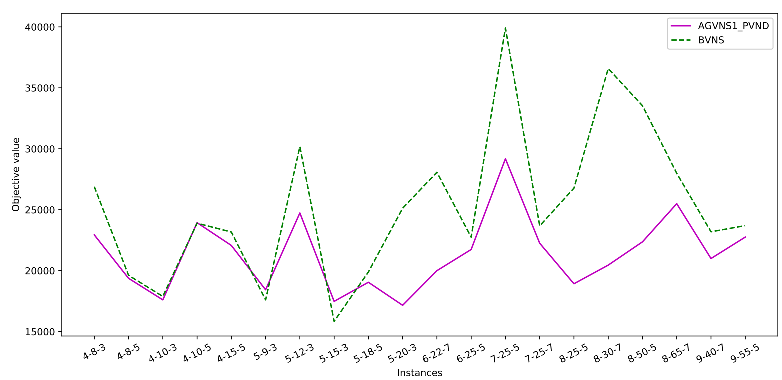

Fig. 1. $AGVNS1_{pVND}$ vs BVNS on 20 PLIRP instances

Table 5 depicts the best values found by all the proposed methods. Most of them were produced using the $AGVNS1_{pVND}$ algorithm.

Table 5. Best found values of the proposed methods

Instance	$GVNS_{CVND}$	$AGVNS1_{CVND}$	$AGVNS2_{CVND}$	$GVNS_{PVND}$	$AGVNS1_{PVND}$	$AGVNS2_{PVND}$
4-8-3	23033.9	22935.29	**22934.45**	**22934.45**	**22934.45**	**22934.45**
4-8-5	19600.14	19622.18	19717.81	19436.24	**19368.56**	19379.93
4-10-3	**17327.69**	17411.45	17504.14	17607	17587.21	17607.18
4-10-5	23935.3	23922.35	23942.93	23926.63	**23921.99**	23931.95
4-15-5	22091.05	22067.85	22219.5	**21907.34**	22048.38	22118.42
5-9-3	18480.68	18480.68	18515.44	18494.14	**18425.61**	18439.89
5-12-3	24748.28	24746.24	24746.24	24732.45	**24730.78**	24735.6
5-15-3	17470.64	17468.54	17470.64	**17465.72**	17475.51	**17465.72**
5-18-5	19370.63	19342.65	19300.07	19037.44	19009.86	**18960.25**
5-20-3	17121.62	**17063.64**	17178.92	17221.55	17130.9	17129.98
6-22-7	**19967.57**	**19967.57**	19967.58	20008.2	19980.53	20008.21
6-25-5	21859.47	21830.73	21787.44	21779.89	21701.27	**21623.56**
7-25-5	29854.82	29754.79	29936.99	29158.77	29154.19	**29119.54**
7-25-7	22288.16	**22124.58**	22229.73	22199.59	22239.25	22297.28
8-25-5	**18729.09**	**18729.09**	19021.24	18827.73	18838.97	18775.41
8-30-7	20629.6	20606.69	20553.74	20420.14	**20414.56**	20515.41
8-50-5	23090.5	22980.6	23066.54	**22284.9**	22348.25	22306.72
8-65-7	25744.94	**24424.65**	**24424.65**	26908.92	25176.64	25176.64
9-40-7	21181.06	20700.37	20773.95	21041.51	**20655.25**	20846.73
9-55-5	22698.68	22372.37	22437.05	22649.32	22503.26	22358.33
Average	21461.19	21327.62	21386.45	21402.1	**21282.27**	21286.56

The second column of Table 6 presents the current best known values of the 20 PLIRP instances. In the third column the overall best values achieved by the proposed methods of this work are provided. As it can be seen, new best solutions have been reported in 13 out of 20 PLIRP instances.

Table 6. BKS vs Best found values of the proposed methods

Instance	BKS	Overall_Best
4-8-3	22647.63	22934.45
4-8-5	18282.71	19368.56
4-10-3	16929.96	17327.69
4-10-5	23895.99	23921.99
4-15-5	22013.99	**21907.34**
5-9-3	16700.29	18425.61
5-12-3	24152.36	24730.78
5-15-3	15842.7	17465.72
5-18-5	19891.27	**18960.25**
5-20-3	24605.64	**17063.64**
6-22-7	28074.69	**19967.57**
6-25-5	22747.42	**21623.56**
7-25-5	39914.72	**29119.54**
7-25-7	23675.7	**22124.58**
8-25-5	26773.1	**18729.09**
8-30-7	36582.34	**20414.56**
8-50-5	33536.73	**22284.9**
8-65-7	27986.69	**24424.65**
9-40-7	23176.14	**20655.25**
9-55-5	23688.55	**22358.33**

5 Conclusions

This work presents several Adaptive GVNS-based algorithms for solving PLIRP instances. Two variants of the adaptive mechanism were developed based on the initial order of the neighborhoods. The proposed algorithms were compared both with their corresponding classic GVNS methods and other heuristic methods [7] for this specific problem. The computational results reveal the superiority of the Adaptive GVNS, which uses the pipe-VND as its main improvement step and the complexity-based initial neighborhoods order in the adaptive re-ordering mechanism. Furthermore, new best values have been reported for 13 out of 20 PILRP instances.

Future work can focus on the determination of lower bounds in order to evaluate the efficiency of the proposed methods. An other future research direction can examine the use of more sophisticated adaptive mechanisms for the re-ordering of the neighborhoods structure. Also, the adaptability may be applied both on the improvement and the shaking step, in order to generate neighborhood patterns that they will produce high quality solutions. Furthermore, this work can be generalized in order to address other environmental implications, such as the emission of pollutants during production [11]. Finally, such computational difficult problems that combine decisions for location, inventory and routing, can be benefited a lot by using parallel computing techniques [1].

Acknowledgements

The second author has been funded by the University of Macedonia Research Committee as part of the "Principal Research 2019" funding scheme (ID 81307).

References

1. Antoniadis, N., Sifaleras, A.: A hybrid CPU-GPU parallelization scheme of variable neighborhood search for inventory optimization problems. Electronic Notes in Discrete Mathematics **58**, 47–54 (2017)
2. Cheng, C., Yang, P., Qi, M., Rousseau, L.M.: Modeling a green inventory routing problem with a heterogeneous fleet. Transportation Research Part E **97**, 97–112 (2017)
3. Glover, F., Gutin, G., Yeo, A., Zverovich, A.: Construction heuristics for the asymmetric TSP. European Journal of Operational Research **129**, 555–568 (2001)
4. Hansen, P., Mladenovic, N.: Variable Neighborhood Search. In: Burke, E., Kendall, G. (eds.) Search Methodologies: Inroductory Tutorials in Optimization and Desicion Support Techniques, chap. 12, pp. 313–337. Springer Science+Business Media, New York (2014)
5. Hansen, P., Mladenovic, N., Todosijevic, R., Hanafi, S.: Variable neighborhood search: Basics and variants. EURO Journal on Computational Optimization **5**, 423–454 (2017)
6. Huber, S., Geiger, M.: Order matters – A Variable Neighborhood Search for the Swap-Body Vehicle Routing Problem. European Journal of Operational Research **263**, 419–445 (2017)
7. Karakostas, P., Sifaleras, A., Georgiadis, C.: Basic VNS algorithms for solving the pollution location inventory routing problem. In: ICVNS 2018. Lecture Notes in Computer Science. vol. 11328, pp. 64–76. Springer, Cham (2019)
8. Karakostas, P., S.A., Georgiadis, C.: A general variable neighborhood search-based solution approach for the location-inventory-routing problem with distribution outsourcing. Computers & Chemical Engineering **126**, 263–279 (2019)
9. Li, K., Tian, H.: A two-level self-adaptive variable neighborhood search algorithm for the prize-collecting vehicle routing problem. Applied Soft Computing **43**, 469–479 (2016)
10. Sifaleras, A., Konstantaras, I.: General variable neighborhood search for the multi-product dynamic lot sizing problem in closed-loop supply chain. Electronic Notes in Discrete Mathematics **47**, 69–76 (2015)
11. Skouri, K., Sifaleras, A., Konstantaras, I.: Open problems in green supply chain modeling and optimization with carbon emission targets. In: Pardalos, P.M., Migdalas, A. (eds.) Open Problems in Optimization and Data Analysis, pp. 83–90. Springer Optimization and Its Applications (2018)
12. Todosijevic, R., Mladenovic, M., Hanafi, S., Mladenovic, N., Crevits, I.: Adaptive general variable neighborhood search heuristics for solving the unit commitment problem. Electrical Power and Energy Systems **78**, 873–883 (2016)