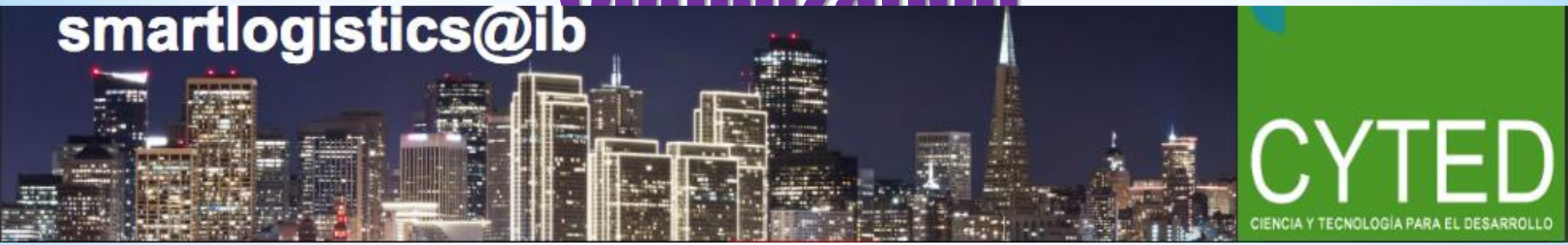
Extending the Variable Neighborhood Search Metaheuristic into a Simheuristic for Stochastic Combinatorial Optimization



November 27-28 Madrid

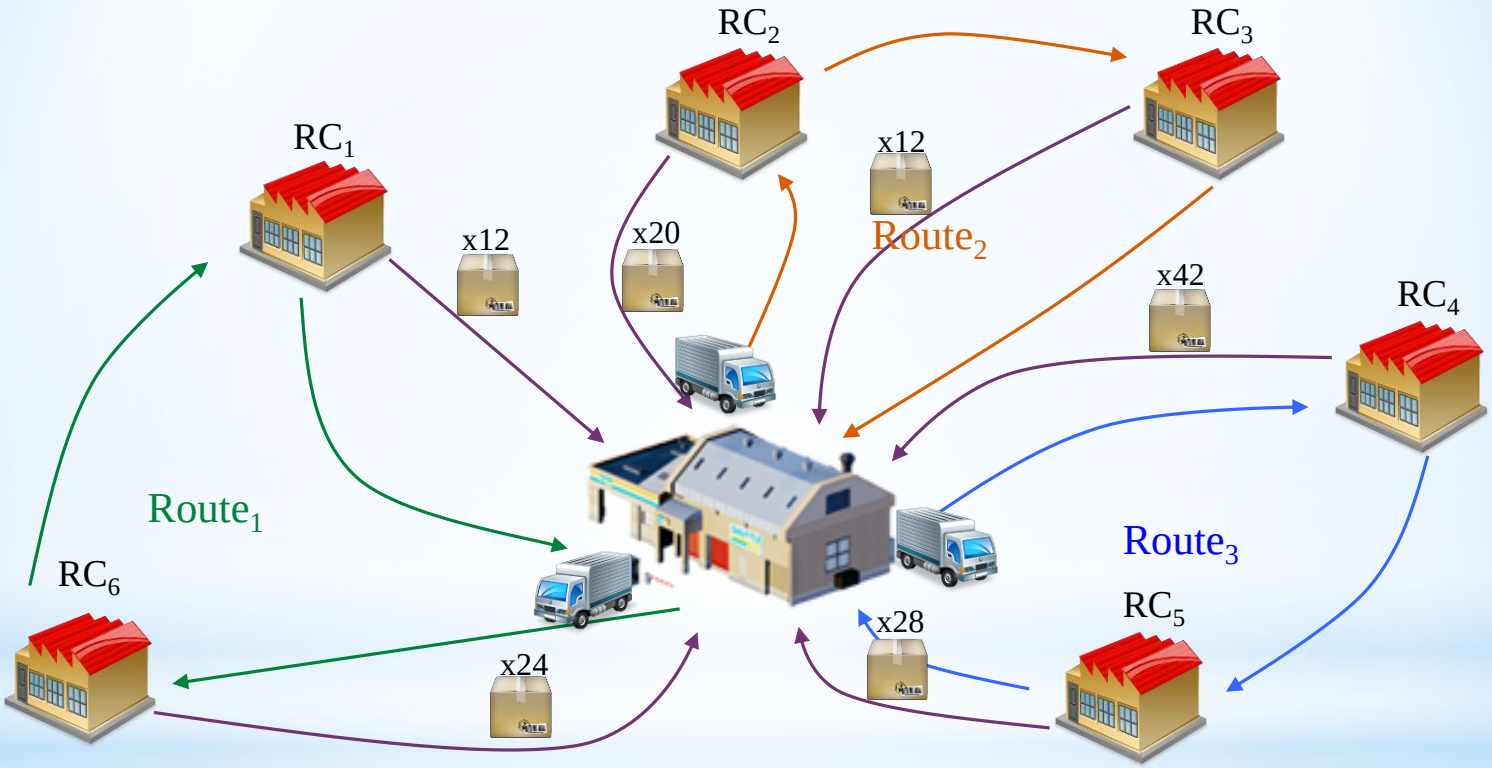
Authors: Jesica de Armas, Aljoscha Gruler, Javier Panadero, José Andres Moreno Perez, Angel Juan

Universitat Oberta de Catalunya
IN3 INTERNET INTERDISCIPLINARY INSTITUTE
http://in3.uoc.edu
ICSO INTERNET COMPUTING & SYSTEMS OPTIMIZATION
http://dpcs.uoc.edu

ULL
Universidad de La Laguna

Introduction

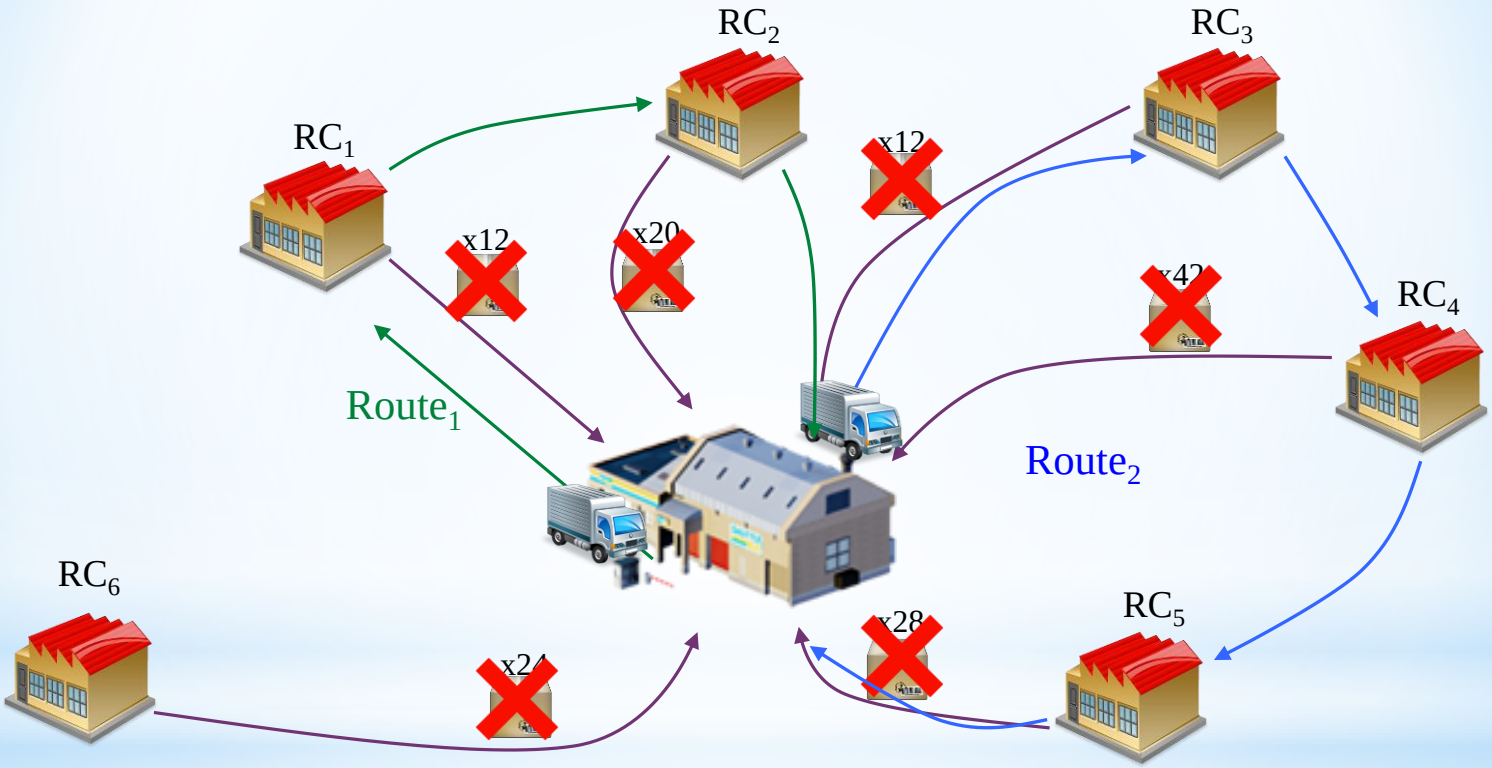
One the most important parading in supply chain management is to move from descentralized decisions to cooperative decisions



Each Retail Center determines its own actions independently of others Retail Centers

Introduction

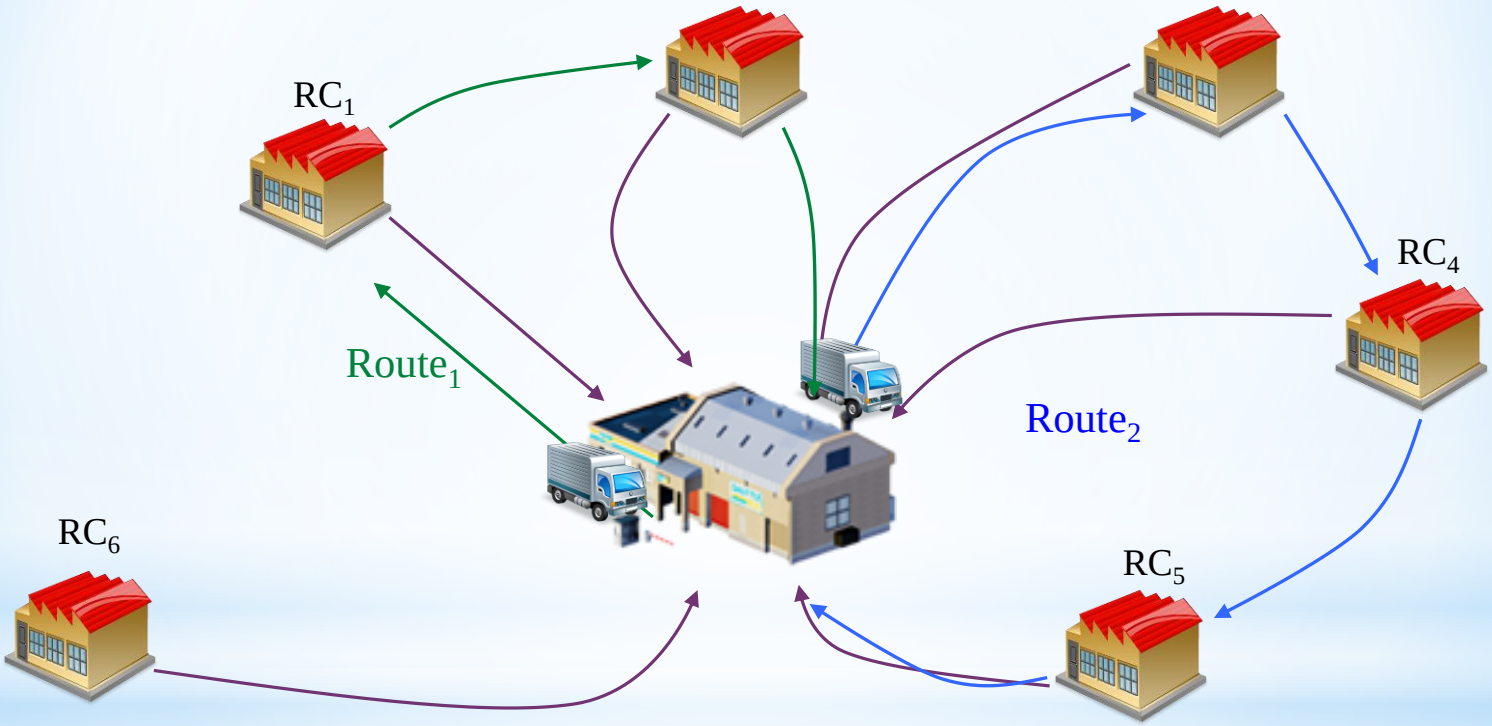
One the most important parading in supply chain management is to move from descentralized decisions to cooperative decisions



Vendor (Depot) takes decisions about the inventory levels of its retail centers

Introduction

One the most important parading in supply chain management is to move from descentralized decisions to cooperative decisions



Iventory Routing Problem (IRP)

Introduction

Iventory Routing Problem (IRP)

- The vehicle routing problem (VRP)
- Inventory Management \ Vendor Managed Inventory (VMI)

➢ Trade-off decisions:

- When to deliver a customer?
- How much to deliver a customer?
- Which delivery routes to use?

Take better decissions in the global sytem

Minimize the Total **Cost** (Inventory cost + Routing cost) for the planning period

Introduction

Iventory Routing Problem (IRP)

- The vehicle routing problem (VRP)
- Inventory Management \ Vendor Managed Inventory (VMI)

➢ Trade-off decisions:

- When to deliver a customer?
- How much to deliver a customer?
- Which delivery routes to use?

High Level of complexity resulting for the integration

Methods to obtain optimal or quasi-optimal solutions in a bounded time

Index

* Introduction
* **Objetives**
* Formal description problem
* Proposed Methodology

Objectives

Main Objective

Methodology to solve the IRP using a simheuristic approach in which a metaheuristic solution technique (VNS) is combined with Monte Carlo Simulation (MCS)

Context

Variable Neighborhood Search (VNS)

IRP with stochastic demands

Consider initial stock levels and possible inventory stock-outs

Single Period

Index

* Introduction
* Objetives
* **Formal description Problem**
* Proposed methodology
* Experimental validation
* Conclusions and future work

Formal description problem

IRP
- Inventory Decision
- Vehicle Routing Problem (VRP)

Total Inventory Cost

$$I(r_i, D_i; I \in V^*) = \sum_{\in} f(\ ,\ )$$

L_i Current (Initial) Inventory Level

L_i^* Maximun Stock Capacity

$\leq \quad \leq$ (Percentage of the Max. Stock Capacity)

$= \quad \times$ (Replenishment level)

$$= \begin{cases} - & < \\ & \geq \end{cases}$$ (Order quantity)

(Function cost)

(MCS)

$\geq \quad = \quad >$

Suplus $=$ $-$

Stock-Out

$$f(\ ,\ ) = \begin{cases} (\ -\ ) & \geq \\ & \leq \end{cases}$$

Formal description problem

IRP

- Inventory Decision
- Vehicle Routing Problem (VRP)

Total Inventory Cost

$$I(r_i, D_i; I \in V^*) = \sum_{\in} f(\ ,\ )$$

$G = (V, A)$ (Complete and undirected graph)

$V = V^* \cup 0$

$V^* = \{1,2,3,\ldots,n\}$

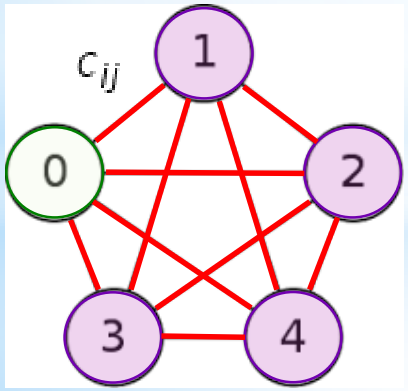
$A = \{(i,j) \mid i \in V, j \in V\}$

$c_{ij} = c_{ji}$

Total Routing Cost

$$(\ ) = \sum_{\in} \sum_{\in} \sum_{\in} (\ )$$

$X \in \{0,1\}$

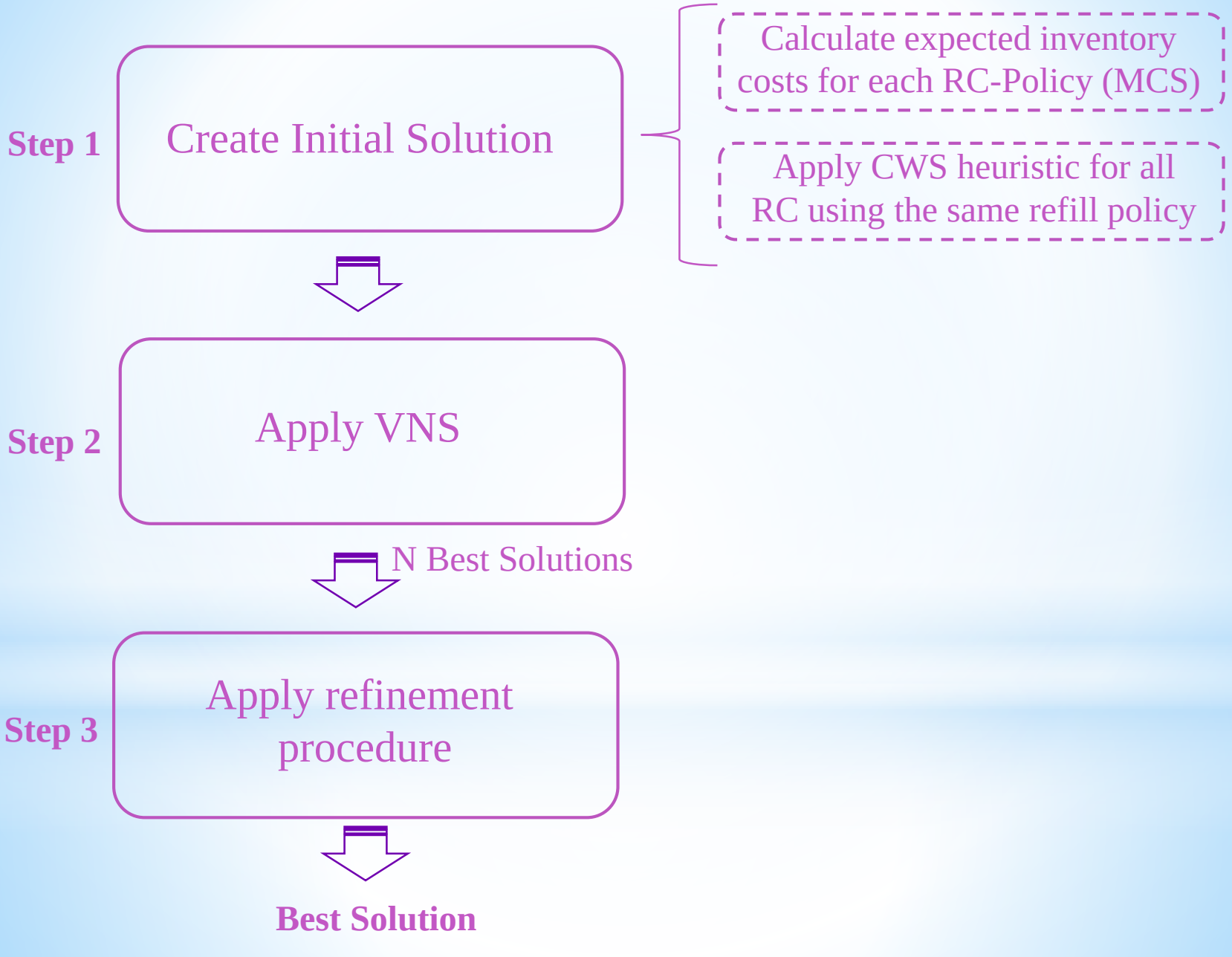


Minimize { $E[I(r_i,D_i)] + R(x)$ }

Index

* Introduction
* Objetives
* Formal description Problem
* **Proposed methodology**
* Experimental validation
* Conclusions and future work

Proposed methodology



Proposed methodology

Algorithm 1: Calculate expected inventory costs

Input: set V^* of RCs, Inventory policies p, number of simulation runs $nSim$, initial stock $initStock_i$ of each RC i

```
1  foreach i ∈ V* do
2      simRun ← 0
3      while simRun ≤ nSim do
4          d_i ← MSC()      // Simulate from probability distribution
5          foreach Inventory Policy p do
6              expInvCost = 0
7              surplus = unitstoServe(p) + initStock_i - d_i
                       // unitsToServe is FIXED, as it only depends on
                       policy
8              expInvCost = calcInvCost(surplus)
                                                  // Stock-out or holding
9              invCosts_i(p) += expInvCost
           end
10         simrun++
       end
11     foreach Inventory Policy p do
12         invCosts_i(p) = invCosts_i(p)/nSim
       end
   end
13 return expected inventory cost invCosts_i(p) for each policy at each RC
```

Proposed methodology

Proposed methodology

Algorithm 3: VNS framework

```
Input: initSol
while stopping criteria not reached do
    bestSolList ← ∅
    bestSolList.add(initSol)
    baseSol ← initSol
    shuffle(neighborhoods_k)
    k ← 1
    repeat
        newSol ← shaking(baseSol, k)
        improving ← true
        while improving                                  // Local Search
            else
                improving ← false
            end
        end
        if totalCosts(newSol) < totalCosts(baseSol) then
            bestSolList.add(newSol)
            baseSol ← newSol
            k ← 1
        end
        else
            k ← k+1
        end
    until k > k_max
end
return bestSolList
```

Operator	Description
Random policy change	Randomly change inventory policy of $k\%$ customers.
Splitting	Destroy and repair part of current *baseSol*.

Proposed methodology

Proposed methodology

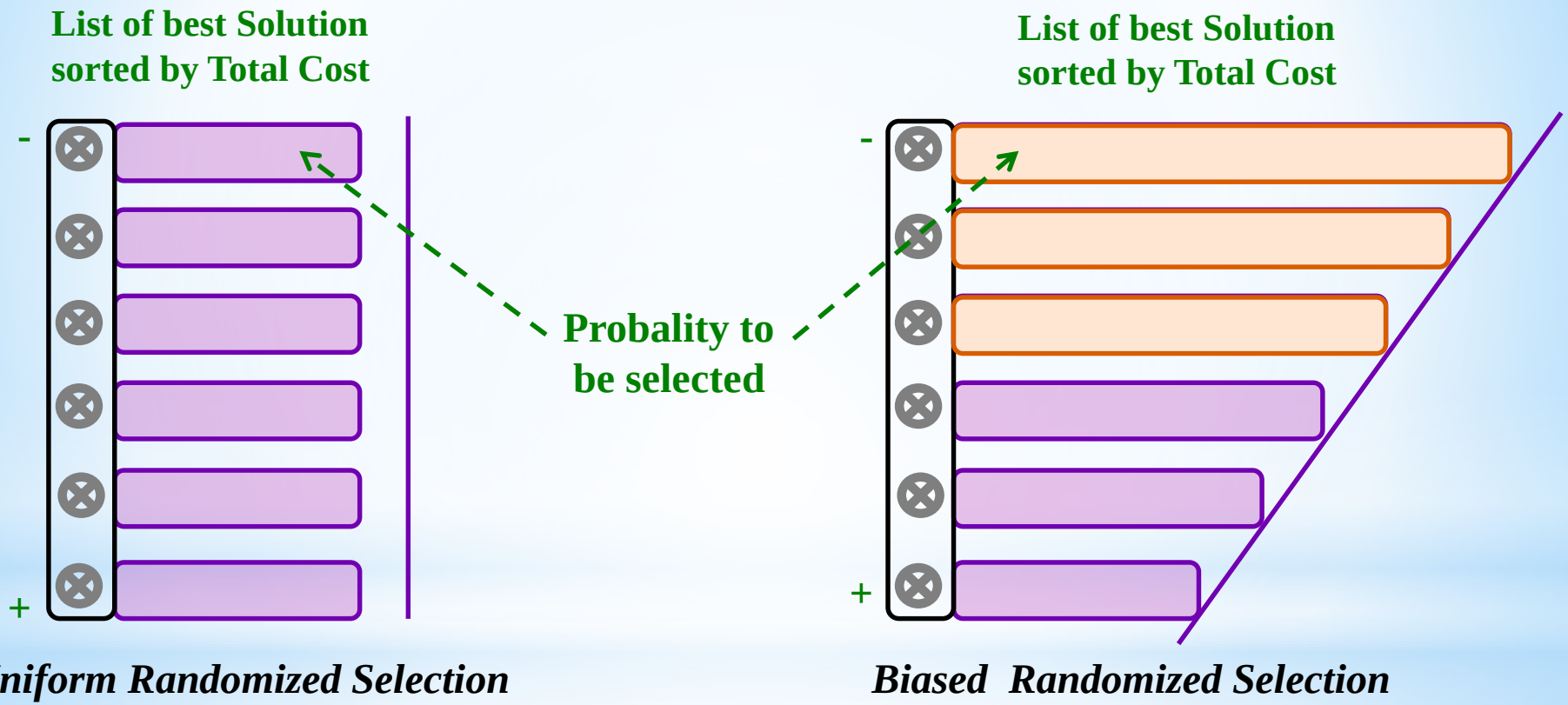
Algorithm 3: VNS framework

```
Input: initSol
while stopping criteria not reached do
    bestSolList ← ∅
    bestSolList.add(initSol)
    baseSol ← initSol
    shuffle(neighborhoods_k)
    k ← 1
    repeat
        newSol ← shaking(baseSol, k)
        improving ← true
        while improving                                  // Local Search
            end
            if totalCosts(newSol) < totalCosts(baseSol) then
                bestSolList.add(newSol)
                baseSol ← newSol
                k ← 1
            end
            else
                k ← k+1
            end
        until k > k_max
    end
return bestSolList
```

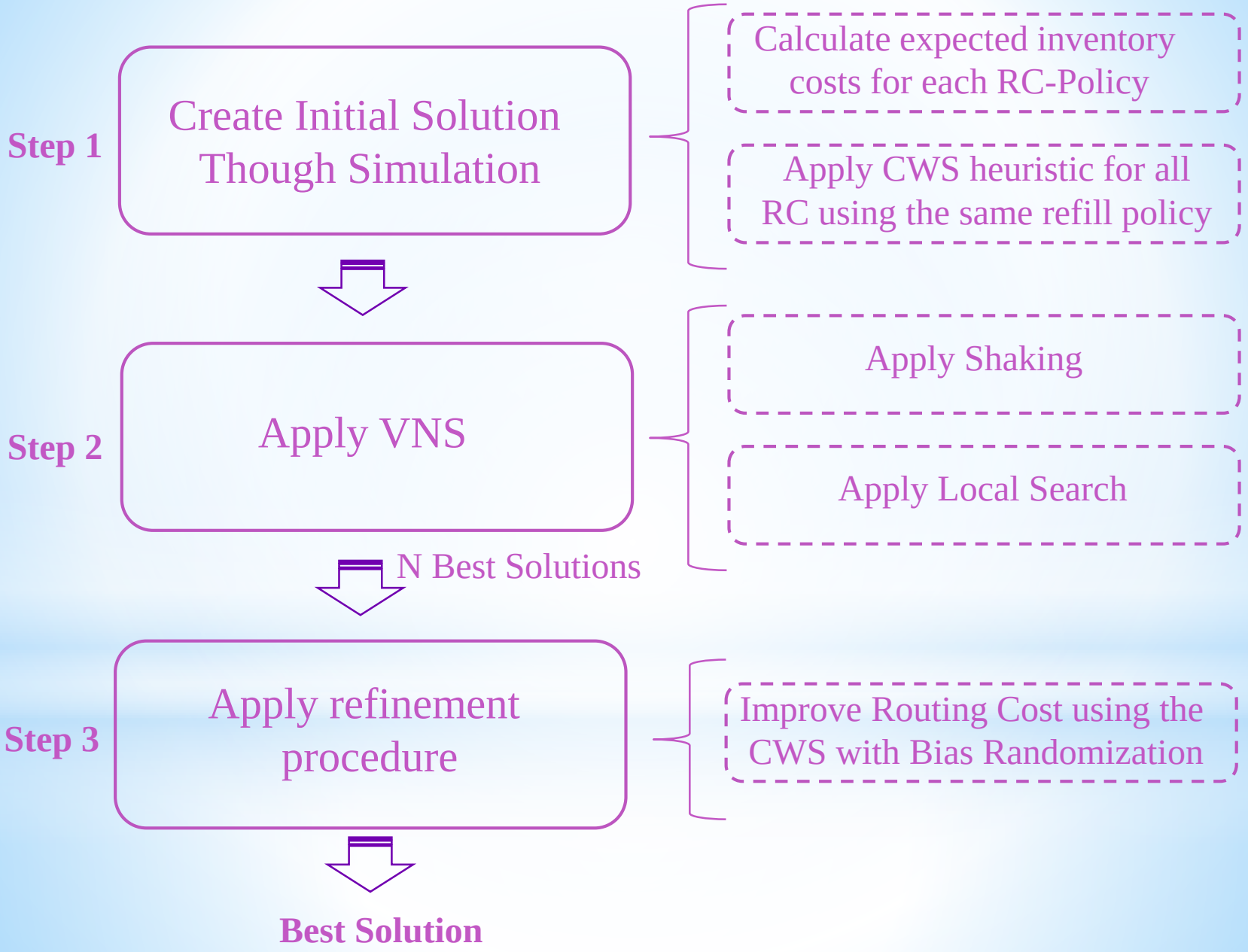
Operator	Description
Greedy heuristic	One by one, the replenishment strategy for each client is adapted to the one with the highest associated decrease in the overall objective function.
Biased randomized heuristic	Biased changing of single customer inventory decisions while considering global inventory and routing costs.

Proposed methodology

Biased Randomization



Proposed methodology



Index

* Introduction
* Objetives
* Formal description problem
* Proposed Methodology
* **Experimental validation**
* Conclusions and future work

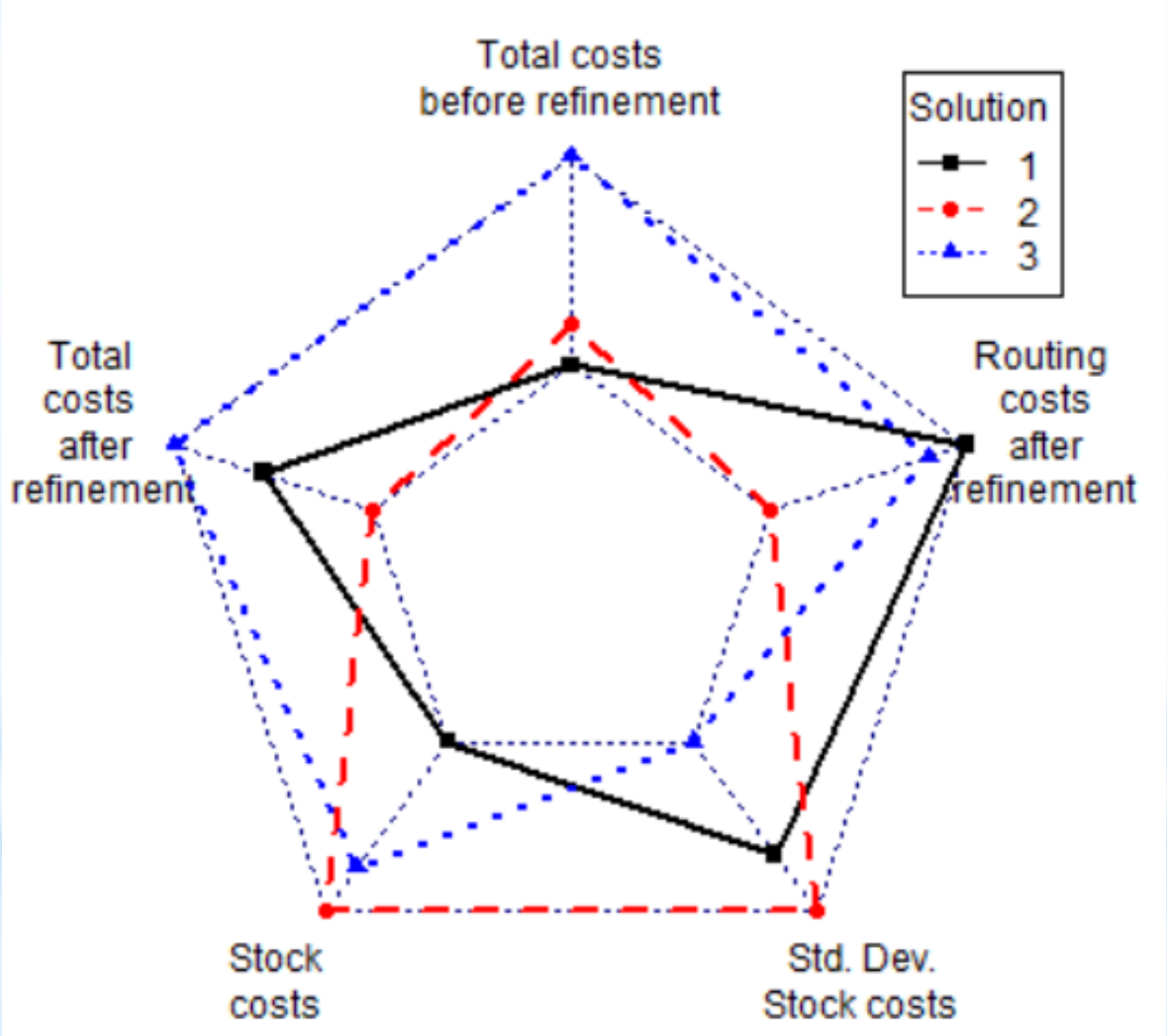
Experimental Validation

- We use the benchmarks for the VRP provided by Augerat et al. (1995).
- The data set consists of 27 instances ranging from 27-80 Retail Centers.
- Each instance is tested with different λ values (0.01/0.25/0.5/0.75/1.0) and three demand variance levels (0.25/0.5/0.75), leading to a total of 15 test per instance (405 total test).
- We have defined 5 *refill policies* (no refill, 0.25%, 0.50%, 0.75%, 1%)
- We compare our results with the obtained results in Juan et al. (2014) (BKS), using the same benchmarks. The algorithm proposed by Juan et al. (2014) has been executed in order to compare our results in the same machine. (30 seconds for each execution in both cases)
- The algorithm is implemented using Java Standard Edition 7 and computational experiments have been performed using a 2.3 Ghz Quad-Core AMD Opteron(tm) processor with 8GB of RAM running under CentOS release 6.6.

Experimental Validation

λ/ Variance-combination	(1) BKS	(2) Routing costs[1]	(3) Inventory costs[1]	(4) Total Costs[1]	(5) Total Costs[2]	%-Gap (1)-(4)	%-Gap (1)-(5)
0.01/0.25	821.47	783.64	30.70	815.99	814.88	-0.67	-0.80
0.01/0.5	891.29	797.24	83.11	877.72	876.6	-1.52	-1.65
0.01/0.75	961.81	800.46	137.53	937.99	937.13	-2.48	-2.57
0.25/0.25	907.4	771.98	114.24	891.81	890.57	-1.72	-1.85
0.25/0.5	976.09	785.48	167.97	963.77	962.63	-1.26	-1.38
0.25/0.75	1049.02	799.88	230.23	1030.11	1029.05	-1.80	-1.90
0.5/0.25	991.52	766.90	201.84	979.74	978.13	-1.19	-1.35
0.5/0.5	1062.43	780.98	257.15	1050.38	1049.34	-1.13	-1.23
0.5/0.75	1138.28	787.01	332.32	1119.33	1117.98	-1.66	-1.78
0.75/0.25	1074.92	756.95	304.68	1061.67	1059.05	-1.23	-1.48
0.75/0.5	1149.34	772.25	365.07	1137.33	1136.04	-1.05	-1.16
0.75/0.75	1224.37	777.59	433.68	1211.29	1209.22	-1.07	-1.24
1/0.25	1154.91	743.32	376.14	1141.86	1140.71	-1.13	-1.23
1/0.5	1234.49	760.18	437.37	1224.17	1222.26	-0.84	-0.99
1/0.75	1311.33	760.24	536.12	1296.38	1296.13	-1.14	-1.16
Average	***1063.24***	***776.27***	***267.21***	***1049.31***	***1047.99***	***-1.33***	***-1.45***

Experimental Validation



Index

* Introduction
* Objetives
* Formal description problem
* Proposed Methodology
* Experimental validation
* **Conclusions and future work**

Conclusions and future work

Conclusions

- We have presented an initial Simheuristic approach in which the VNS metaheuristic is combined with Monte Carlo Simulation.
- Our algorithm is easy-to-to implement and provides solutions to large IRP problem settings in only a few seconds
- A range of experiments underline the algorithm's competitiveness compared to previously used heuristic methodologies.

Future Work

- Extend the Single Period IRP to Multiperiod.
- Extend to multidepot IRP.

Thanks!


Universitat Oberta de Catalunya
IN3
INTERNET INTERDISCIPLINARY INSTITUTE
http://in3.uoc.edu
ICSO
INTERNET COMPUTING & SYSTEMS OPTIMIZATION
http://dpcs.uoc.edu


ULL
Universidad de La Laguna