Cooperative Vs Non-Cooperative Parallelization Strategies for Variable Neighborhood Search metaheuristic: A case study on the Capacitated Vehicle Routing Problem

Panagiotis Kalatzantonakis

University of Macedonia, School of Information Sciences, Department of Applied Informatics,
156 Egnatia Str., 54636 Thessaloniki, Greece

Oct 30, 2019

Publications

List of publications derived from this thesis:

- Kalatzantonakis P., Sifaleras A., and Samaras N., *"Cooperative Vs Non-Cooperative Parallel Variable Neighborhood Search Strategies: A case study on the Capacitated Vehicle Routing Problem"*, re-submitted to *Journal of Global Optimization*, Springer, 2019 (minor revision).
- Kalatzantonakis P., Sifaleras A., and Samaras N., *"On a cooperative VNS parallelization strategy for the capacitated vehicle routing problem"*, to appear in N. Matsatsinis, Y. Marinakis, and P. M. Pardalos (Eds.), *Learning and Intelligent Optimization*. LION13. Lecture Notes in Computer Science, 27-31 May, Chania, Crete, Greece, 2019.
- Kalatzantonakis P., Sifaleras A., Samaras N., Migdalas Ath., *"Parallel Variable Neighborhood Search for the Capacitated Vehicle Routing Problem"*, presented at the 6th *International Conference on Variable Neighborhood Search*. ICVNS, Sithonia, Halkidiki, Greece, October 4-7, 2018, p. 32 (book of abstracts).

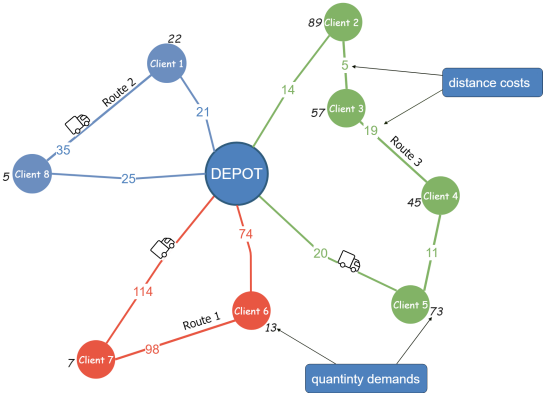
Outline

Objectives

The objectives of this thesis are as follows:

- To survey recent successful parallel implementations of the VNS metaheuristic regarding several variants of vehicle routing problems.
- To present three parallelization strategies using the General VNS variant to tackle the CVRP, in which we examine different approaches of exchanging solutions among parallel executions.
- To experimentally evaluate how the level of cooperation can affect the performance between non-cooperative and cooperative models.
- To improve the quality of the solutions by creating a new parallel model that filters communication based on the tests performed.

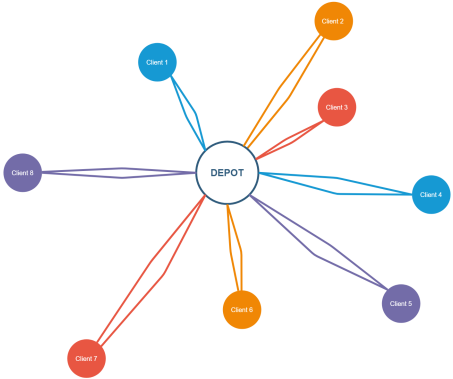
Capacitated Vehicle Routing Problem (CVRP)



CVRP was introduced in:

Dantzig, G. B. and J. H. Ramser (1959).
The truck dispatching problem.
Management Science 6(1),80–91.

Clarke and Wright construction heuristic (1/2)

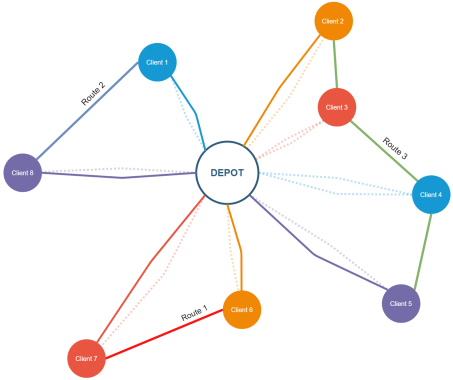


Clarke and Wright heuristic was introduced in:

Clarke, G. and J. W. Wright (1964). *Scheduling of vehicles from a central depot to a number of delivery points*. Operations Research 12 (4), 568–581.

- The algorithm starts with one route for every customer.

Clarke and Wright construction heuristic (2/2)



- In every iteration the number of vehicles is reduced unifying two routes that give max savings.

Variable Neighborhood Search (VNS)

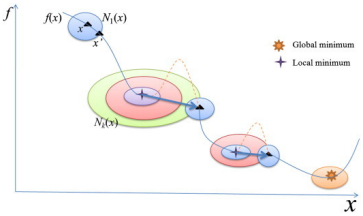


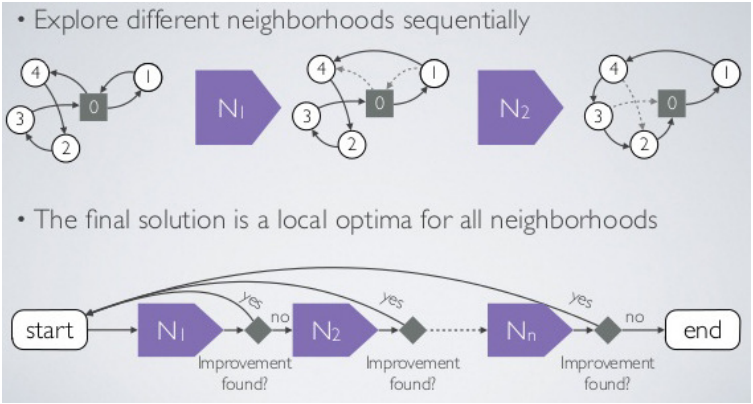
Figure: Systematic neighborhood changes.

VNS is a metaheuristic, or framework for building heuristics, proposed in:
Mladenović, N., Hansen, P.: *Variable neighborhood search*. Computers & Operations Research24(11), 1097–1100 (1997)

- **Fact 1** A local minimum w.r.t. one neighborhood structure is not necessary so with another;
- **Fact 2** A global minimum is a local minimum w.r.t. all possible neighborhood structures.

General Variable Neighborhood Search (GVNS)

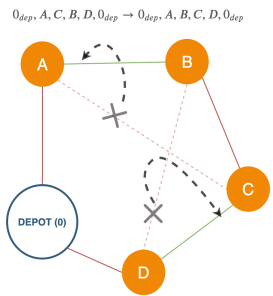
When the change of neighborhoods is performed in a deterministic way (Variable Neighborhood Descent) then a more generic VNS emerges, called **General Variable Neighborhood Descent** (GVNS).



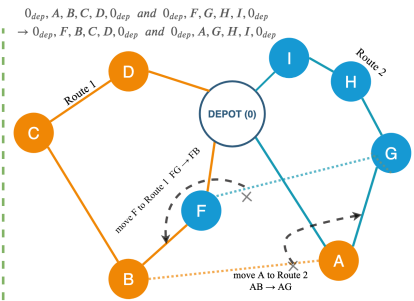
Neighborhood structures

Three operators were used:

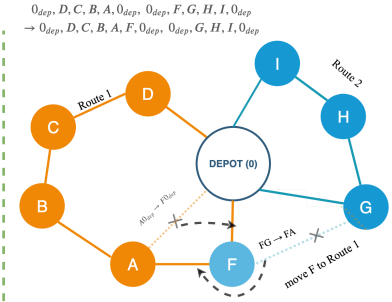
2-opt (Intra-route), Swap (Inter-route), Relocate (Inter-route)



Intra route 2-opt



Inter route swap



Inter route relocate

Computing environment

- We ran the experiments on a computer with the following characteristics:
  - Windows 10 64bit
  - Intel Core i9 CPU 7940X at 3.50 GHz with 20 MB Cache
  - 32GB DDR4 at 3333MHz
- The parallel implementation was done using:
  - Python 3.7 with the multiprocessing library
  - Cython for the compilation.

Benchmark instances

Tests were carried out using the CVRP benchmark instance library (CVRPLib) that can be found at: *http://vrp.galgos.inf.puc-rio.br/index.php/en/*

The sets A, B, E, M, *Golden* and X were used.

- All instances from the sets A,B,E and M have optimum values.
- Many instances from the Golden and the X set do not have an optimum value.

Stopping criteria

Our models are executed until one of the stopping criteria is met. The stopping criteria for this experiment are the following:

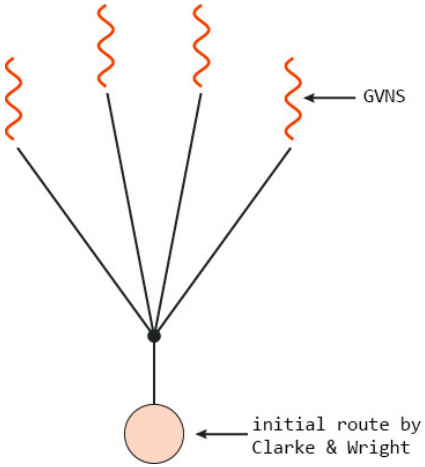
- the maximum execution time; set to 3600 seconds.
- the maximum number of GVNS iterations; set to 300.
- when the optimal value is reached, if it exist.

Parallel strategies

We propose three different strategies to parallelize GVNS:

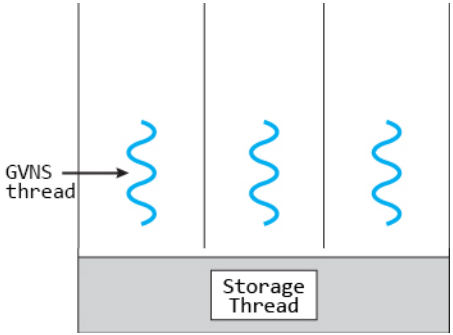
- The first one parallelizes the whole GVNS method while keeping every thread completely isolated.
- The second one parallelizes the whole GVNS, but all threads communicate the best solution through manager proxy, maintaining a dense communication.
- The third one parallelizes the whole GVNS. Communication is sparse and handled through a manager proxy. Solution exchange is based on a self-adaptive criterion called θ.

Parallel strategies, Non-cooperative model



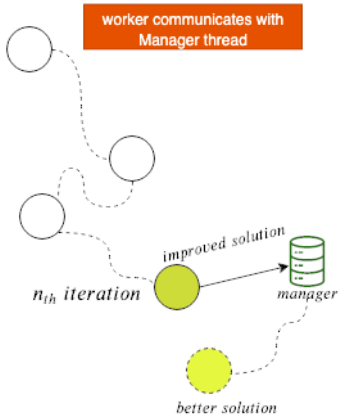
- Island-based design; every thread runs the GVNS algorithm isolated.
- Threads are autonomous and utilize identical search procedures.
- When a stopping criteria is met the best solution is picked.

Parallel strategies, Managed Cooperative Model (MCM)



- Initial solution is given using Clarke and Wright.
- One of the threads is used as a solution warehouse (Server Manager).
- Threads communicate with the Server Manager.
- No broadcasting takes place.

Parallel strategies - MCM communication strategy



MCM communication strategy

- Upon solution improvement then and only then a threads communicates with the Server Manager.
- Communication between the threads and the server manager is very dense at the start (intensification).

GVNS algorithm - MCM

Algorithm 1 GVNS, Managed Cooperative Model (MCM)

function GVNS(fr, $kmax$, $timelimit$, $manager$)
  while true **do**
    $t \leftarrow CpuTime()$
    $fr' \leftarrow Shake(fr, kmax)$
    $fr'' \leftarrow VND(fr', t, timelimit)$
    if $thread\_solution < thread\_current\_best$ **then**
      $thread\_current\_best \leftarrow thread\_solution$
      if $thread\_current\_best < manager\_global$ ▷ Communicate with warehouse
      then $manager\_global \leftarrow thread\_current\_best$
      else $thread\_current\_best, thread\_solution \leftarrow manager\_global$
    if $t > timelimit$ **then** break
    if $manager\_global == optimum$ **then** break ▷ if optimum value exists
  end while
end function

Parallel strategies - Parameterized Cooperation Model (PCM)

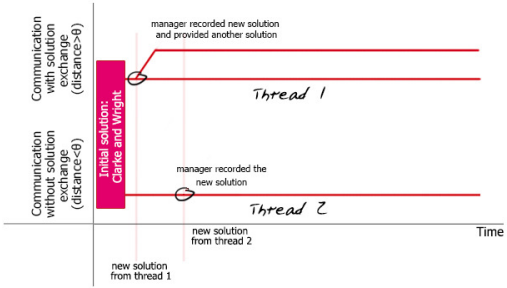


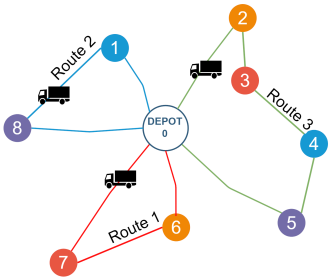
Figure: Thread-1 changes its solution.
Thread-2 continues to work with the same solution.

Bridging the gap

- Same communication strategy as MCM.
- A self-adaptive parameter (θ) is used.
- Solutions are compared using Levenshtein.
- Solution exchange is filtered using distance and θ.

PCM - Levenshtein distance

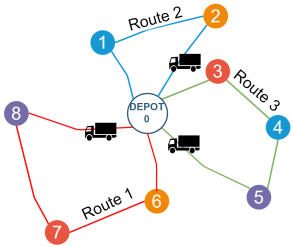
The distance between two solution is calculated using the Damerau - Levenshtein distance:



4 edit operations

+8 -8 +2 -2

06780 — 08120 — 02 3450



Solution 1
string representation: 06700810023450
Length:14

Solution 2
string representation: 06780012003450
Length=14

Normalized distance = 0.2857

PCM - Criterion calculation

θ criterion

θ is based on the remaining time and GVNS iterations as depicted in Algorithm 2:

Algorithm 2 Parameter self-adaptation

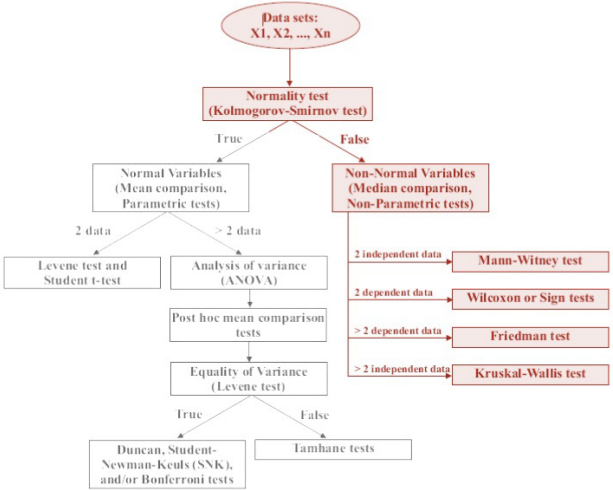
$\theta \leftarrow 1$
if $(niter_i > 0.2 \times niter_{\max}) \vee (currenttime > 0.2 \times time_{\max})$
then $\theta = \min\left((1 - \frac{niter}{niter_{max}}), (1 - \frac{time}{time_{max}})\right)$

GVNS algorithm

Algorithm 3 GVNS, Parameterized Cooperative Model (PCM)

function GVNS(*fr*, *kmax*, *timelimit*, *manager*)
  while true **do**
    $t \leftarrow CpuTime()$;
    $fr' \leftarrow Shake(fr, kmax)$;
    $fr'' \leftarrow VND(fr', t, timelimit)$;
    $\theta \leftarrow dynamic\_decrease(time, iter\_count)$;
    if $thread\_solution < thread\_current\_best$ **then**
      $thread\_current\_best \leftarrow thread\_solution$
      if $thread\_current\_best \leq manager\_global$ **then** ▷ Communicate with manager
        $manager\_global \leftarrow thread\_current\_best$
      else
        if $\theta < distance(manager\_global, thread\_current\_best)$ **then**
          $thread\_current\_best, thread\_solution \leftarrow manager\_global$ ▷ solution exchange
    if $(t > timelimit)$ or $(manager\_global = optimum)$ **then** break
  end while
end function

Statistical analysis



- Kolmogorov-Smirnov test was used.
- The distribution deviates significantly from the normal distribution.
- non-parametric tests were used (Wilcoxon, Friedman, Kruskal).

Experimental results

In the results shown in the table 1 we can observe essential differences among the compared methods.

Table: Comparison of the three GVNS parallel variants at 300 GVNS iterations

	NCM	MCM	PCM
average error	0.837%	1.260%	0.729%
average CPU time (secs)	76.880	92.100	88.640
Optimal number of solutions	21	16	21

Findings

Our results show that:

- The full cooperative model (MCM) produced almost the same quality of solutions as the non-cooperative model when solving easier instances.
- Filtered cooperation (PCM) and the MCM produced superior quality solutions regarding the more computationally difficult test instances.
- Filtered cooperation (PCM) produced superior quality solutions in all instances.
- Although, the non-cooperative model (NCM) has a more intense diversification phase, the sparse communication between processes achieved by the communication coordination in the Parameterized Cooperative Model is producing better solution quality by maintaining a balanced level of interaction between the threads.

Conclusions

- We used several known instances in order to compare and analyze the co-operation strategies.
- We have proposed three parallel GVNS versions based on python multiprocessing library for solving the CVRP.
- The communication strategy used by the PCM is a novel approach that, reduces the communication overhead and producing better solution quality.
- There seems to be a strong indication that, the cooperation strategy can have a decisive influence on the quality of the solutions.

End of presentation

Thank you for your attention!