A reinforcement learning-VNS for the cVRP

University of Macedonia
Applied Informatics

RL-VNS, AI-DRIVEN OPTIMIZATION STRATEGIES

EXPLORING COMBINATORIAL LANDSCAPE WITH AI-DRIVEN METAHEURISTICS

Presented by Panagiotis Kalatzantonakis

May 17, 2023

Kalatzantonakis P., Sifaleras A., and Samaras N.

University of Macedonia, School of Information Sciences, Department of Applied Informatics

RESEARCH GOALS

In this study, we aim to:

- Improve the search process in metaheuristics, specifically Variable Neighborhood Search (VNS), by integrating Reinforcement Learning (RL) for intelligent navigation.
- Achieve improved solution quality and decreased execution time in solving combinatorial optimization problems (COPs).
- Lay the foundation for an emerging research discipline by introducing a novel, advanced concept - a hybrid RL and VNS model.
- Validate our approach using the Capacitated Vehicle Routing Problem (CVRP), a classic example of COPs, and contrast it with traditional VNS.

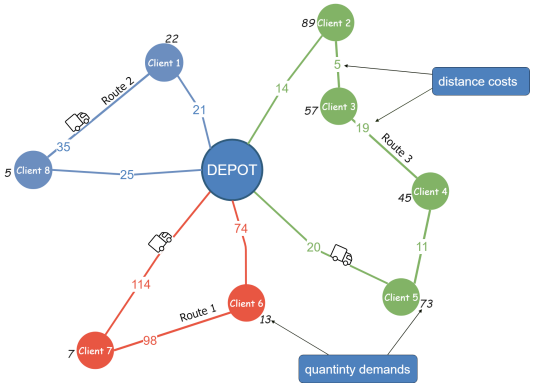
THE CAPACITATED VEHICLE ROUTING PROBLEM

The CVRP was initially introduced by Dantzig and Ramser in their landmark 1959 publication, *"The truck dispatching problem"* [2]

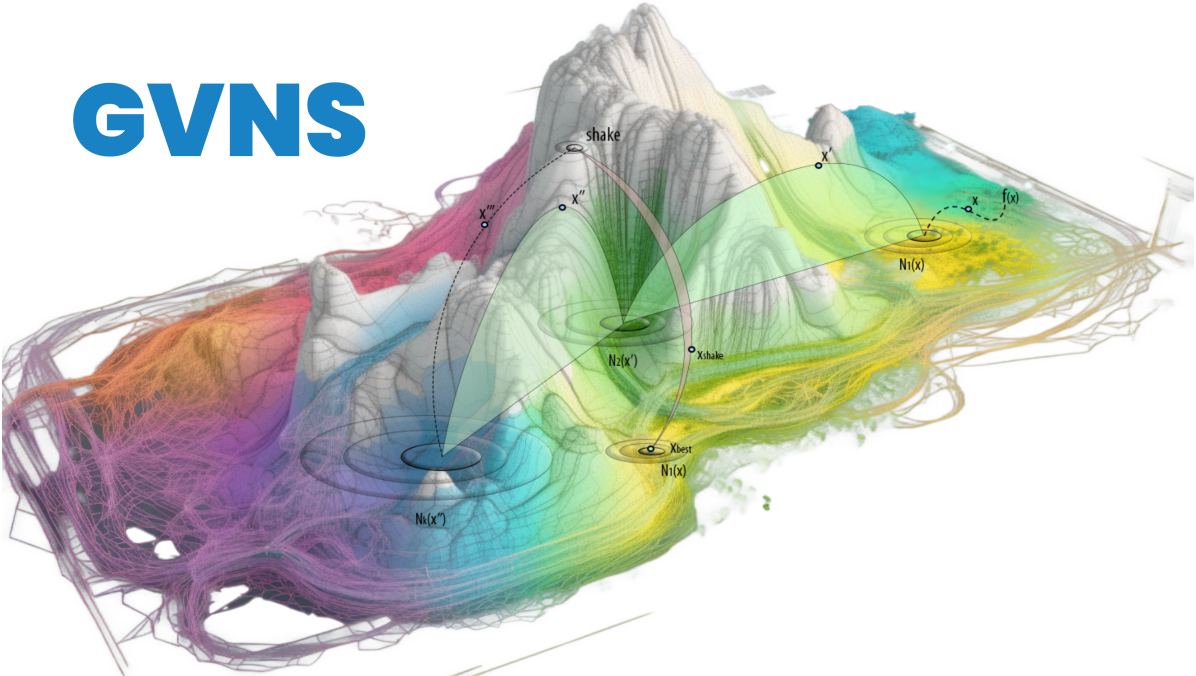
CVRP: CORE SPECIFICATIONS

The Capacitated Vehicle Routing Problem (CVRP) is characterized by:

- Vehicles start and end at the depot
- Each customer visited once by a single vehicle
- Vehicle capacity cannot be exceeded by demand



GVNS



GENERAL VARIABLE NEIGHBORHOOD SEARCH

A meta/hyper-heuristic framework for optimization

- Key concepts:
  - Systematic change of neighborhood structures
  - Local search within each neighborhood
  - Shake operation to escape local optima
- Advantages of GVNS:
  - Explores a larger search space by combining multiple neighborhood structures
  - Can adapt to various optimization problems
  - Balances exploration and exploitation to enhance search efficiency

EXPLORING THE ROLE OF NEIGHBORHOODS

Core Concepts and Strategic Functions

A. Core concept

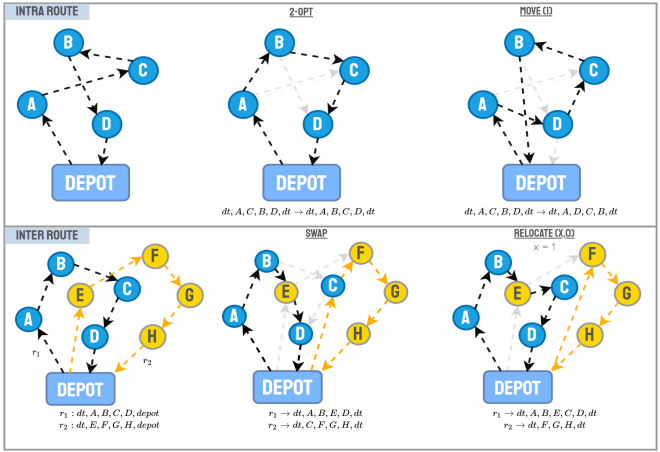
- 1. **Neighborhoods**: sets of nearby candidate solutions
- 2. **Proximity** is determined by specific distance metrics or solution representations

B. Strategic Function

- 1. Boost refined local search by probing neighboring solutions
- 2. Skillfully balance exploration and exploitation in heuristic search

NEIGHBORHOOD STRUCTURES IN ACTION

A Visual Representation within the GVNS Framework



Intra-route:

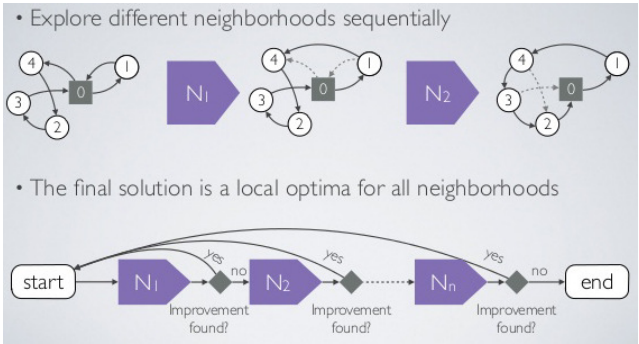
- $2 - OPT$,
- $MOVE(1)$

Inter-route:

- $SWAP$,
- $RELOCATE(X, 0)$

EXPLORING THE GVNS ALGORITHM

An Examination of Search Space Exploration Strategies



GVNS is an advanced version of the Variable Neighborhood Search (VNS) methodology. It systematically adjusts neighborhood structures to thoroughly explore the search space. GVNS utilizes Variable Neighborhood Descent (VND) as a local search procedure, alternating neighborhoods deterministically or randomly for a comprehensive search. This approach has been successfully applied in various fields.

REINFORCEMENT LEARNING

A brief overview

- Reinforcement Learning (RL) is a branch of Machine Learning focused on decision-making.
- RL learns from interaction with the environment to achieve a specific goal.
- Key components of RL:
  - **Agent:** The decision-maker that interacts with the environment.
  - **Environment:** The context in which the agent operates.
  - **State:** A representation of the agent's current situation.
  - **Action:** A choice made by the agent that affects the environment.
  - **Reward:** Immediate feedback received by the agent after performing an action.
- The agent's objective is to find an optimal policy

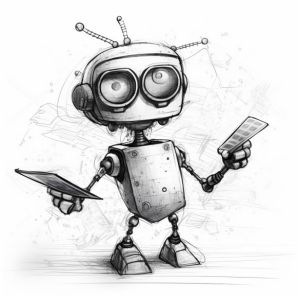
MULTI-ARMED BANDITS

A Special Case of RL Problems

- **Multi-Armed Bandit (MAB) problems:**
  - Formally defined as a tuple (A, R), where A is **a set of actions** (arms) and $R : A \times T \to \mathbb{R}$ is a reward function that maps each action and time step to a real-valued reward.
  - **Aim: Maximize the total reward** over a sequence of decisions, $\sum_{t=1}^{T} R(a_t, t)$, where a_t is the action taken at time step t.
  - **Challenge: Balance exploration** (trying new actions) with **exploitation** (using the best-known option) to maximize cumulative reward.
- **Connection to VNS:**
  - Each neighborhood in VNS can be considered as an arm in MAB.
  - The sequence of local search operators in VNS corresponds to a sequence of arm selections in MAB.
  - Learning to effectively **balance exploration and exploitation** in MAB can lead to improved performance in VNS.

UPPER CONFIDENCE BOUND

A Bridge connecting RL with GVNS



At moment t, to choose an action A_i, we:

- Record the average reward $(\hat{r}_i)$,
- Record total actions tried (n),
- Track the action's execution count (n_i).

We try the action maximizing empirical and theoretical reward (UCB1 algorithm):

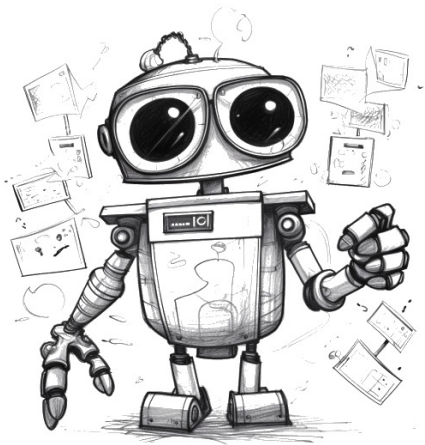
$$Action \doteq \underset{i=\{1\ldots k\}}{\arg\max} \left[\hat{r}_i + C\sqrt{\frac{2\log n}{n_i}} \right]$$

C controls the level of exploration

After execution, we:

- Calculate a reward,
- Update operator's value according to the reward.

BANDIT VNS: A NOVEL APPROACH



Building on the principles of UCB and MAB, Bandit VNS introduces a dynamic approach to neighborhood selection in the VNS framework [5].

This method enhances the search strategy by dynamically exploring and exploiting neighborhoods based on their historical performance, thereby achieving improved solution quality and efficiency in VNS.

INTEGRATING RL IN GVNS

- Training phase:
  - RL agent is trained online with zero initial knowledge using the VNS algorithm.
- Execution phase:
  - RL agent calculates the expected reward for each neighborhood.
  - The action with the highest expected reward is selected by the agent.
  - The agent receives a reward based on the expected reward of the selected action.
  - Continuous analysis of the stream reward is performed, allowing for the detection and mitigation of reward drift.
- Benefits of RL VNS:
  - Convergence to higher quality solutions through adaptive neighborhood selection.
  - Invariable to problem size, enabling it to handle complex problems with large solution spaces.

BANDIT VNS ALGORITHM: VNS WITH UCB1

Algorithm 1: UCB1 operator selection

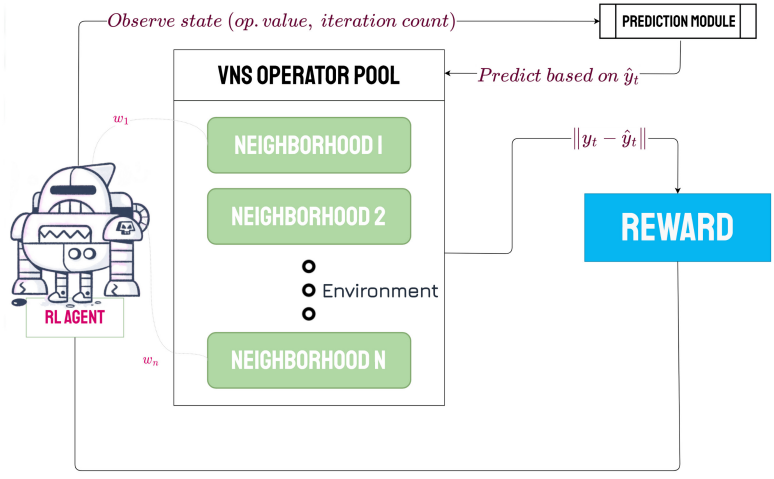
Input : C, it, n, $\hat{p}$
Output: *Selected neighborhood index*

if *operators that have not been selected exist* **then**
  $x \leftarrow$ *uniform selection from operator pool* ; ▶ *like VNS*
else
  $x \leftarrow \arg\max_{i \in \{1 \ldots k\}} (\hat{p}_i + C\sqrt{\frac{2\log(it)}{n_i}})$; ▶ *C ranges btw 0.5 and 2*
return $x \in N_k$;

The selection process is based on:

- Operator execution count
- Operator past assessment value
- Total execution count

SCHEMATIC OF BANDIT VNS PROCESS



LIMITATIONS OF THE UCB ALGORITHM

Addressing the Challenge

Challenge: Concept Drift

- UCB algorithm's decisions are based on the assumption of stationary reward distributions.
- Concept drift events occur when these distributions change in dynamic environments.

Approach: Dynamic Adjustment Using ADWIN

To handle these changes, the UCB algorithm employs the ADWIN drift detector. It resets the learning process upon major concept drift events and re-calibrates its estimations, thereby maintaining accuracy.

CONCEPT DRIFT EVENTS

Impact on GVNS

Concept drift emerges in the reward stream, when the agent traverses the solution domain.

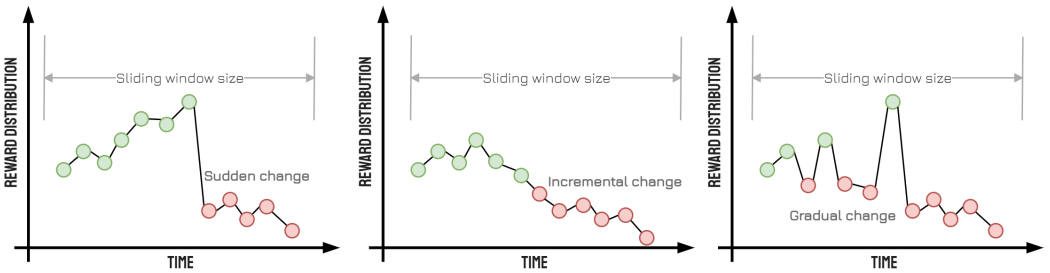
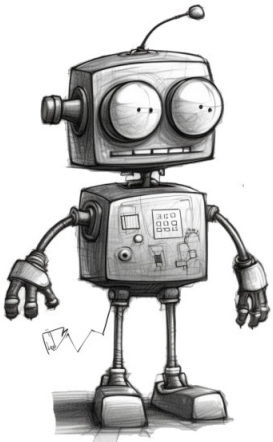


Figure: Three types of Concept drift events

CONCEPT DRIFT IN GVNS

The Role of ADWIN Algorithm



- Monitoring concept drift is crucial for accurate predictions in GVNS.
- The ADWIN Algorithm effectively detects and estimates concept drift.
- Upon detecting any type of change in mean rewards, ADWIN triggers the learning process reset, thereby enhancing system adaptability.

COMPUTING ENVIRONMENT

- We ran the experiments on a computer with the following characteristics:
  - Windows 10 64bit
  - Intel Core i9 CPU 7940X at 3.80 GHz / 20 MB Cache
  - 48GB DDR4 at 3333MHz
- The implementation was done using:
  - Python 3.10,
  - Multiprocessing library for code parallelization,
  - Time-critical parts are implemented in Cython

BENCHMARK INSTANCES

In order to provide a well-balanced test environment, tests were carried out using carefully selected instances from CVRP instance library (CVRPLib) in order to reduce the impact of instance characteristics (customer co-ordinates, depot positioning, customer demands).

- Sets A, B, X and CMT were used.
- All instances from sets A, B, CMT have optimum values.

RESULTS AND PERFORMANCE COMPARISON

Benchmarking Bandit VNS Against GVNS and Five Diverse Algorithms

- **Bandit VNS performance against GVNS:**
  - The solution quality is improved by up to 26.5%, outperforming conventional GVNS, as demonstrated by Kalatzantonakis et al. (2023) [5].
  - Suspension of inefficient neighborhoods leads to a decrease in total execution time.
- **Algorithm comparison:**
  - Five algorithms from three competitive categories: exact solvers (**Gurobi**), heuristics (**LKH3:** Lin et al., 1973; Helsgaun, 2017, **OR-Tools** by Google and **Clarke et al.**), and **RL methods:** Nazari et al. (2018) and Kool et al. (2018), constrained within a time limit of 1 minute for each benchmark instance.
- **Performance metrics:** Solution quality, and total execution time.
- **Parameter tuning:** Fine-tuning UCB, can further improve execution time and solution quality.

EXPERIMENTAL RESULTS

Analysis of BanditVNS and GVNS

Table: Comparison BanditVNS vs GVNS and overall improvement

	Set A	Set B	Set X
BanditVNS avg. error	1.61%	1.65%	6.02%
GVNS avg. error	2.04%	1.83%	6.29%
Overall improvement of BanditVNS over GVNS	26.24%	10.69%	4.42%

EXPERIMENTAL RESULTS

Method		CMT1			CMT2			CMT3			CMT11		
		Avg. error	Gap (%)	Time (sec)	Avg. error	Gap (%)	Time (sec)	Avg. error	Gap (%)	Time (sec)	Avg. error	Gap (%)	Time (sec)
Exact solver	Gurobi (AMPL)	548.40	11.40	60	888.70	6.40	60	924.90	12.00	60	1108.10	6.30	60
Heuristic	CW	1408.00	62.74	0.02	1944.00	57.03	0.06	2957.00	72.06	0.13	2210.00	52.84	0.27
	OR-Tools	556.50	6.10	60	890.80	6.60	60	875.10	5.90	60	1178.10	13.00	60
RL	Kool	531.90	1.40	1.50	867.20	3.80	1.70	882.50	6.80	1.90	1207.70	15.90	3.00
	Nazari	562.10	7.10	7.20	907.50	8.60	11.80	915.30	10.80	17.50	1221.80	17.20	21.80
Hybrid RL Heuristic	BanditVNS	524.61	0.00	25.42	855.26	2.33	60	851.42	2.96	60	1181.34	11.72	60
Optimum CVRPLib		524.61	0.00	-	835.26	0.00	-	826.14.	0.00	-	1042.11	0.00	-

EXPERIMENTAL RESULTS

		5 GVNS iterations			
Rank #	Model	Mean error	Median error	$Executions/sec/thread$	Mean CPU time
1	BanditVNS	6.1832	6.1972	0.2450	27.1341 sec
2	GVNS	6.7467	6.7689	0.2313	36.1216 sec
		10 GVNS iterations			
Rank #	Model	Mean error	Median error	$Executions/sec/thread$	Mean CPU time
1	BanditVNS	5.2497	5.3582	0.2735	47.4241 sec
2	GVNS	5.9587	6.0004	0.2357	69.9459 sec
		20 GVNS iterations			
Rank #	Model	Mean error	Median error	$Executions/sec/thread$	Mean CPU time
1	BanditVNS	4.1632	4.0848	0.2972	67.9731 sec
2	GVNS	5.1062	5.2150	0.2378	128.2903 sec

Table: Model ranking based on best solution quality, neighborhood execution efficiency, and computational economy.

» High Executions/sec/thread ($ExTS$) values denote early execution of resource-intensive neighborhoods due to initial high rewards, with less frequency as returns decrease. Low Mean CPU times indicate efficient blocking of certain neighborhoods, signifying better resource use than standard GVNS.

FINDINGS

Our results show that:

- the RL component improves solution's quality in 75% - 90% of instances,
- it has tiny computation footprint,
- it's fairly easy to implement.

The results have been published in the following paper [5]:

Kalatzantonakis P., Sifaleras A., and Samaras N., "A reinforcement learning - variable neighborhood search method for the capacitated vehicle routing problem", Expert Systems with Applications, Elsevier, Vol. 213, Article ID 118812, 2023.

POTENTIAL FOR FURTHER RESEARCH!

- For collaboration, amendments, or ideas, please feel free to mail at pkalatzantonakis@uom.edu.gr
- The presentation can be accessed on https://github.com/CodedK/LATNA_DOCS/

BACK-UP SLIDES

Adaptive Window: Handling Concept Drift

- **Adaptive Windowing (ADWIN) Algorithm**
  - Efficient concept drift detector and estimation algorithm.
  - Monitors and detects abrupt variations in the standard deviation of mean rewards.
  - Resets the learning process upon a major concept drift event.
- **Concept Drift in UCB [5]**
  - UCB assumes stationary reward distribution, but reward distributions may change in dynamic environments.
  - Concept drift events affect the agent's ability to predict the optimal neighborhood structure.
  - Adjusting UCB estimation dynamically maintains prediction accuracy in changing environments.

BIBLIOGRAPHY

[1] Geoff Clarke et al. “Scheduling of vehicles from a central depot to a number of delivery points”. In: *Operations Research* 12.4 (1964), pp. 568–581.

[2] George B Dantzig et al. “The truck dispatching problem”. In: *Management science* 6.1 (1959), pp. 80–91.

[3] Google. *(Google’s Operations Research tools, 2022)*. URL: `https://developers.google.com/optimization/`.

[4] Keld Helsgaun. “An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems”. In: *Roskilde: Roskilde University* (2017), pp. 24–50.

[5] Panagiotis Kalatzantonakis et al. “A reinforcement learning-Variable neighborhood search method for the capacitated Vehicle Routing Problem”. In: *Expert Systems with Applications* 213 (2023), p. 118812.

[6] Wouter Kool et al. “Attention, learn to solve routing problems!” In: *arXiv preprint arXiv:1803.08475* (2018).

[7] Shen Lin et al. “An effective heuristic algorithm for the traveling-salesman problem”. In: *Operations research* 21.2 (1973), pp. 498–516.

[8] Mohammadreza Nazari et al. “Reinforcement learning for solving the vehicle routing problem”. In: *Advances in neural information processing systems* 31 (2018).