A Reduced Variable Neighborhood Search approach for the Travelling Thief Problem

Ioakeim Fotoglou

University of Macedonia, School of Information Sciences,
Department of Applied Informatics, Egnatia Street 156, 54636, Thessaloniki

30 October 2019

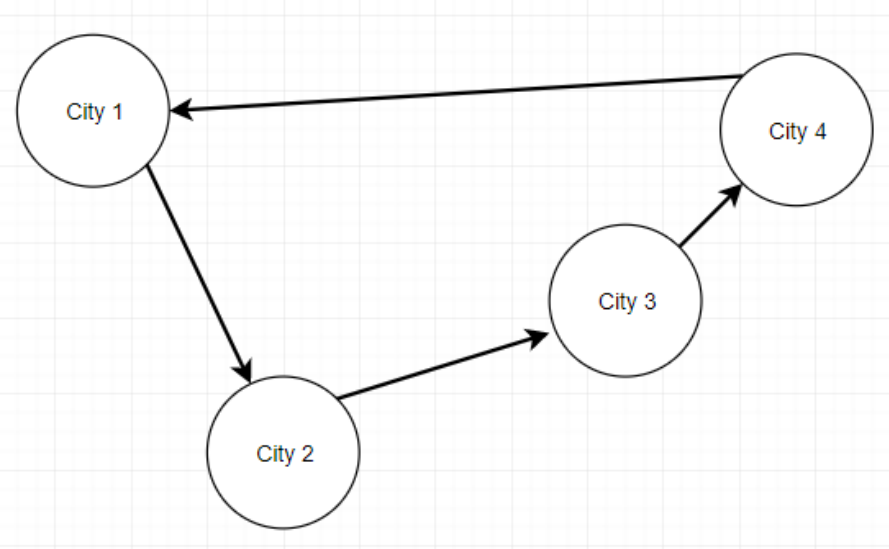
Agenda

- Theoretical Background
- Problem Statement
- Methodology
- Evaluation
- Related Work
- Conclusions and Future Directions

The Travelling Salesman & the Knapsack problem

- The Travelling Salesman Problem (TSP): Given N cities and a starting city, calculate the shortest path that visits every city exactly once and returns to the starting city

- The Knapsack problem (KS): given N items, their associated weights and values and a knapsack with a fixed capacity, select the items that return the maximum profit while not violating the capacity

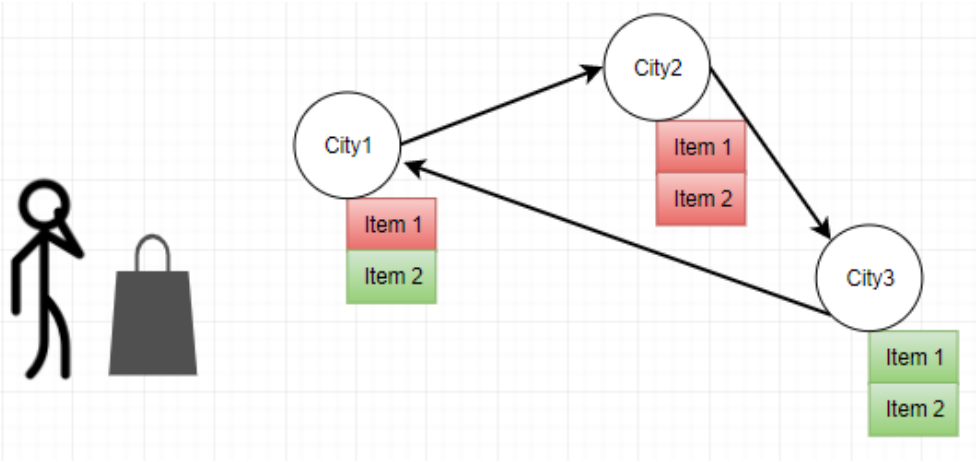


The Travelling Thief Problem

Greedy heuristic

- Interdependency of TSP and KS
- Transition from theoretical problems to problems applicable to the real world

- Given N cities, M items per city and the fact that the weight of the knapsack affects the cost of the route, find the optimal route and packing plan



Objective Function

The objective function for the TTP:

$$maximize \quad Z(\Pi, P) = \sum_{i=1}^{n} \sum_{k=1}^{m_i} p_{ik} y_{ik} - R\left(\frac{d_{x_n x_1}}{u_{max} - v W_{x_n}} + \sum_{i=1}^{n-1} \frac{d_{x_i x_{i+1}}}{u_{max} - v W_{x_i}}\right)$$

where:

Π = the tour, i.e., $\{x_1, ..., x_n\}, x_i \in N$

P = the packing plan, i.e., $\{y_{21}, ..., y_{nm_i}\}$

y_{ik} = binary variable, selection of item k from city i

R = the renting rate paid each time unit being on a way

v = $v_{max} - v_{min}$

W_i = the knapsack weight when leaving city i

W = the maximum knapsack's weight

(Reduced) Variable Neighborhood Search (R)VNS

VNS

initialize solution x
while *stopping criteria are not met* **do**
  $k = 1$
  while $k \leq k_{max}$ **do**
    generate x' a random solution from neighborhood $N_k(x)$
    $x'' = localSearch(x')$
    if $evaluate(x'') < evaluate(x)$ **then**
      $x = x''$
      $k = 1$
    else
      $k = k + 1$
    end
  end
end
return x;

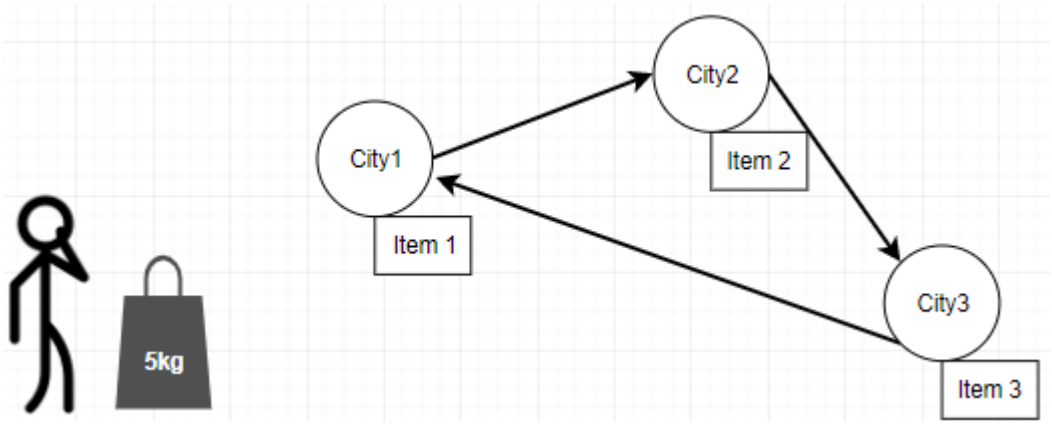
RVNS

initialize solution x
while *stopping criteria are not met* **do**
  $k = 1$
  while $k \leq k_{max}$ **do**
    generate x' a random solution from neighborhood $N_k(x)$
    if $evaluate(x') < evaluate(x)$ **then**
      $x = x'$
      $k = 1$
    else
      $k = k + 1$
    end
  end
end
return x;

Construction functions – Greedy 1

Greedy heuristic

- TSP construction: Construct tour with nearest neighbor rule
- KS construction: Rank items based on their value/weight ratio and select items with the highest rank until no more items fit



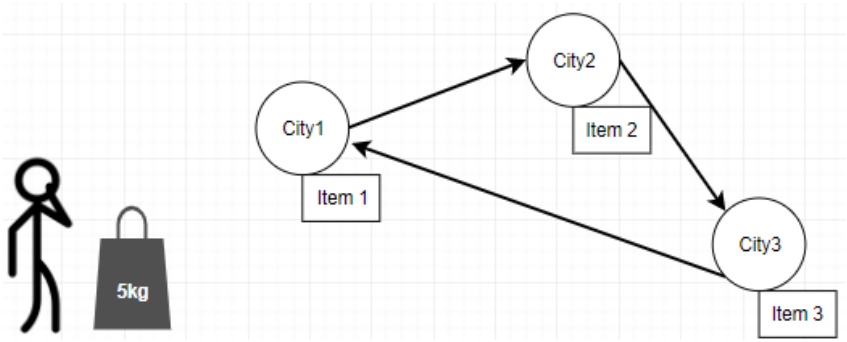
	City 1	City 2	City 3
City 1	0	1	2
City 2	1	0	1
City 3	2	1	0

	Value	Weight	Ratio
Item 1	10	4	2.5
Item 2	15	3	5
Item 3	4	1	4

Construction functions – Greedy 2

Greedy heuristic

- TSP construction: Construct tour with nearest neighbor rule
- KS construction: Calculate ratios like before and assign a risk factor to each item depending on how many cities are left



	City 1	City 2	City 3
City 1	0	1	2
City 2	1	0	1
City 3	2	1	0

	Value	Weight	Ratio	Risk
Item 1	10	4	2.5	1.2
Item 2	15	3	5	0.4
Item 3	8	2	4	0.25

$$risk_i = (1 + p_s) \cdot \frac{1}{r_i}$$

where p_s denotes the number of outstanding cities to reach at the last city of the permutation, counting from city s

Solution Representation and Neighborhood Definitions

Solution representation is a combination of the typical:

- TSP representation: permutation of *N* cities, e.g. s_1=[1,3,2,4]
- KS representation: a binary vector with length *NM*, *M*: #items per city, e.g. s_2=[0,1,0,1]

TTP representation: s={s_1,s_2}

The two neighborhoods we use:

- First neighborhood: Given s, s' in N1(s) has *one item flipped* from the existing packing plan
- Second neighborhood: Given s, s' in N2(s) has *one selected item removed from a city at the beginning of the route and one item from a city added at the end of the route*

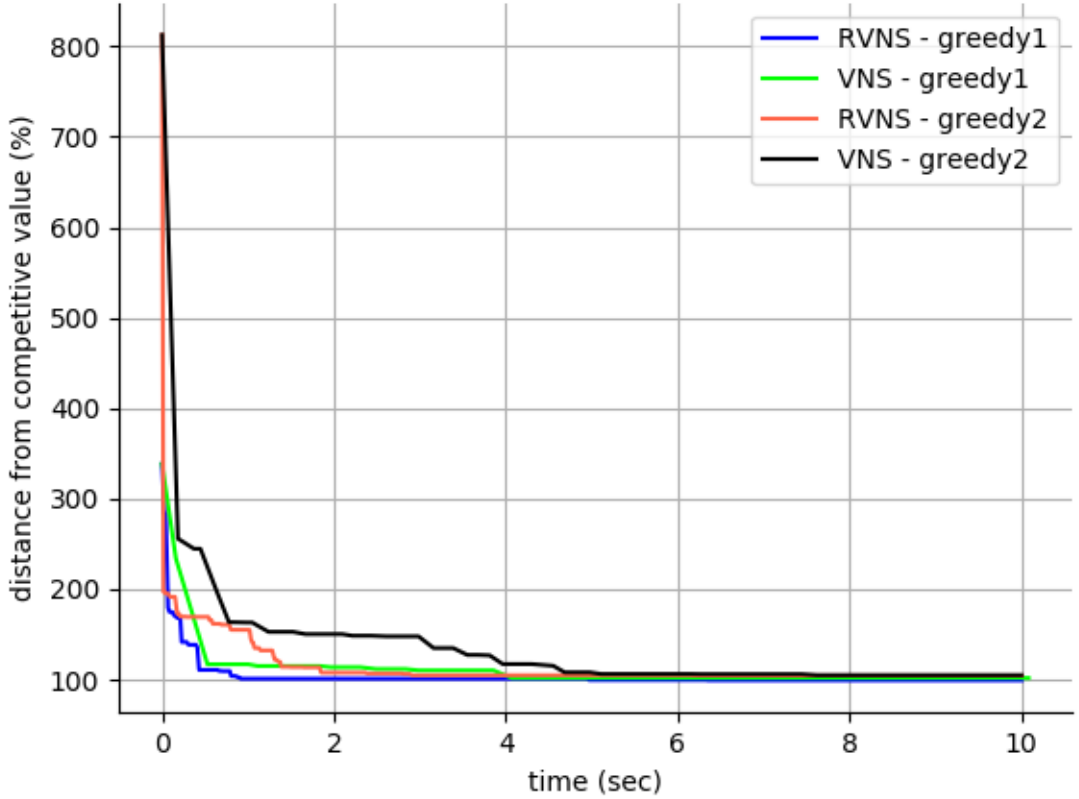
Related Datasets

We utilize four instances from four different datasets on the TTP problem:

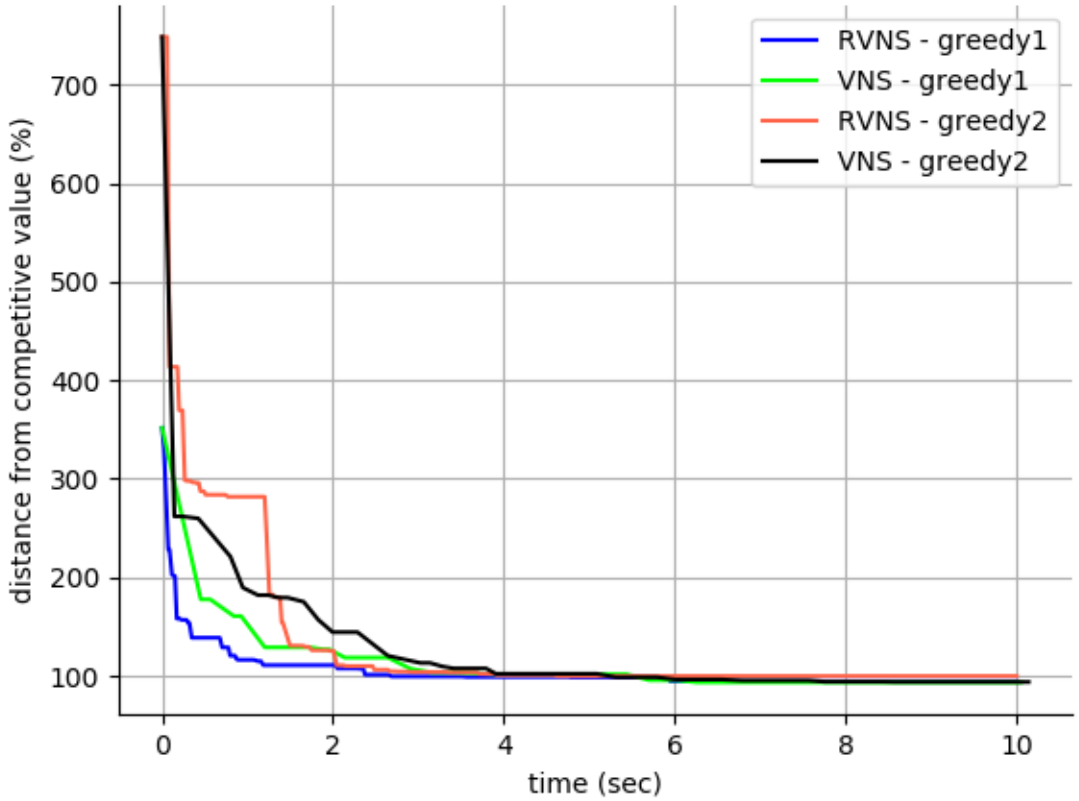
Id	Cities	Items/city	Weight/Profit corr.
Eli76-tpp	76	1	Strongly correlated
kroA100-tpp	100	1	Strongly correlated
Ch130-tpp	130	1	Strongly correlated
U159-tpp	159	1	Strongly correlated

These instances were selected to enhance comparability because they were used for testing in the relevant research literature.

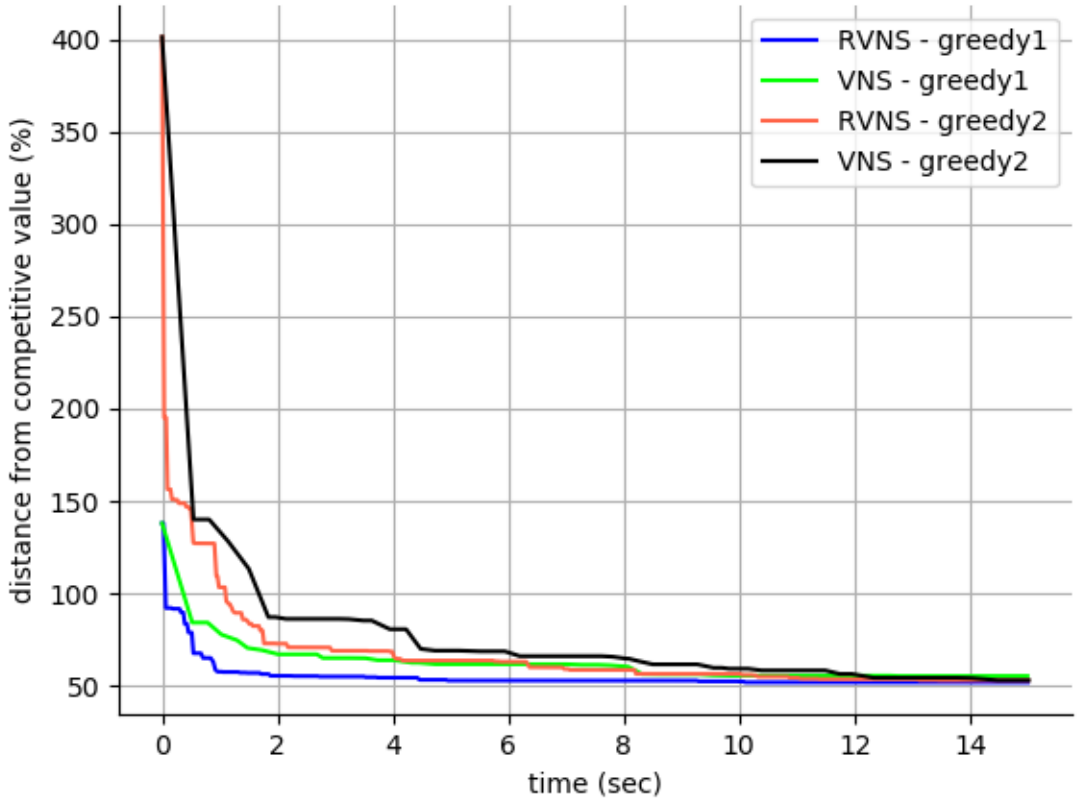
Indicative Performance – eli76



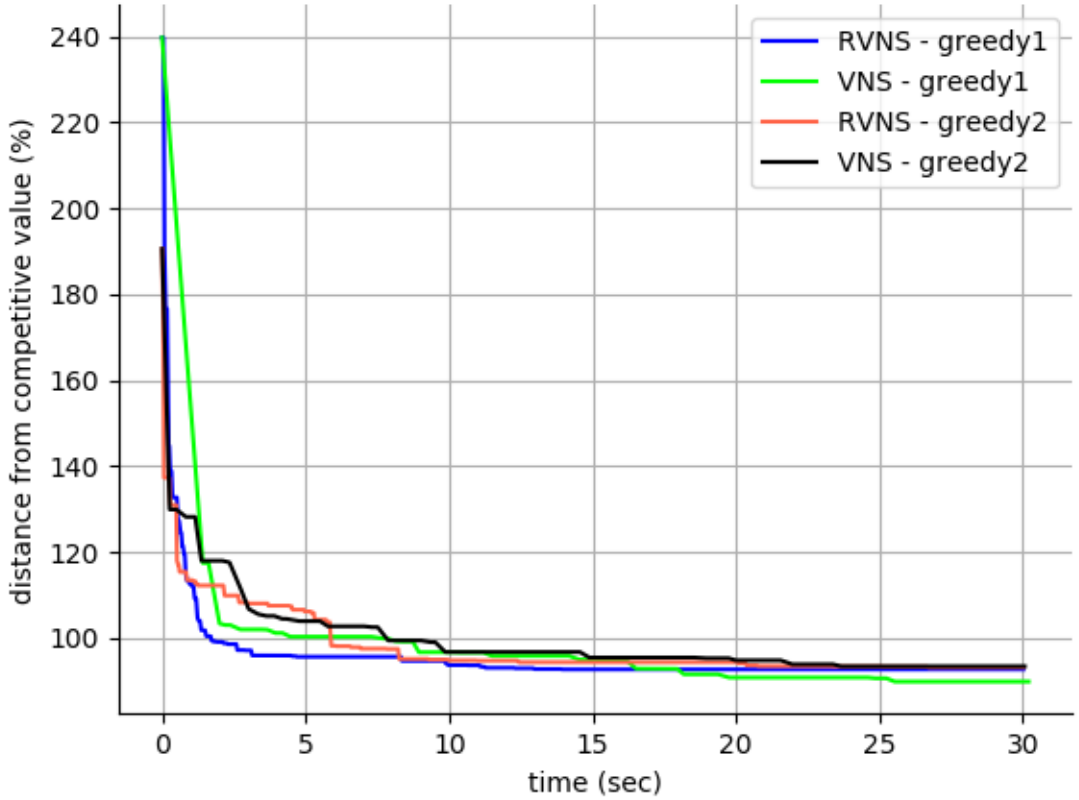
Indicative Performance – kroA100



Indicative Performance – ch130



Indicative Performance – u159



Experimental Evaluation - parameters of VNS and RVNS

dataset	time	algorithm
eli76-tpp	600 sec	CS2SA-R
kroA100	67 sec	MATLS
ch130	600 sec	S5
u159	600 sec	S5

(El Yafrani and Ahiod 2018)

Conclusions

- Both VNS and RVNS can be successfully applied to the Travelling Thief Problem
- The size of the instance, seems to be the major limiting factor for unmodified RVNS
- The starting conditions appear to play a significant role in the quality of the outcome
- RVNS demonstrates potential in areas where frequent and rapid re-calculation of efficient solutions is more important that the actual target value of the objective function

Future Directions

- Differences between the greedy1 and greedy2 construction function heuristics indicate research on a different construction heuristic has potential
- RVNS iterates through many loop flow structures there is significant potential for parallelization
- A combined approach has potential to yield efficient solutions while maintaining, or even improving, the low runtimes of RVNS

End of presentation

Thank you!

Any comments or questions?

End of presentation

BACKUP SLIDES

Bibliography for tests and reference values

Main reference:
El Yafrani, M. and B. Ahiod (2018). *Efficiently solving the traveling thief problem using hill climbing and simulated annealing*. *Information Sciences 432*, p.231-244

- MATLS: A memetic algorithm by (Bonyadi, Michalewicz, and Barone 2013)
- S5 (aka PackIterative): A simple heuristic by (Faulkner, Polyakovskiy, Schultz, and Wagner 2015)
- CS2SA* & CS2SA-R: An approach combining Hill Climbing and Simulated Annealing, implementing the CoSolver framework by (El Yafrani and Ahiod 2018)