Variable Neighborhood Search*

Jason Brownlee
jasonb@CleverAlgorithms.com
The Clever Algorithms Project
http://www.CleverAlgorithms.com

February 6, 2010
Technical Report: CA-TR-20100206-1

Abstract

The Clever Algorithms project aims to describe a large number of Artificial Intelligence algorithms in a complete, consistent, and centralized manner, to improve their general accessibility. The project makes use of a standardized algorithm description template that uses well-defined topics that motivate the collection of specific and useful information about each algorithm described. This report describes the Variable Neighborhood Search algorithm using the standardized template.

Keywords: `Clever, Algorithms, Description, Optimization, Variable, Neighborhood, Search`

1 Introduction

The Clever Algorithms project aims to describe a large number of algorithms from the fields of Computational Intelligence, Biologically Inspired Computation, and Metaheuristics in a complete, consistent and centralized manner [1]. The project requires all algorithms to be described using a standardized template that includes a fixed number of sections, each of which is motivated by the presentation of specific information about the technique [2]. This report describes the Variable Neighborhood Search algorithm using the standardized template.

2 Name

Variable Neighborhood Search, VNS

3 Taxonomy

Variable Neighborhood Search is a Metaheuristic and a Global Optimization technique that manages a Local Search technique. It is related to the Iterative Local Search Metaheuristic.

4 Strategy

The strategy for the Variable Neighborhood Search involves iterative exploration of larger and larger neighborhoods for a given local optima until an improvement is located after which time the search across expanding neighborhoods is repeated. The strategy is motivated by three principles: i) a local minimum for one neighborhood structure may not be a local minimum for a different neighborhood structure, ii) a global minimum is a local minimum for all possible neighborhood structures, and iii) local minimum are relatively close to global minimum for many problem classes.

5 Procedure

Algorithm 1 provides a pseudo-code listing of the Variable Neighborhood Search algorithm for minimizing a cost function. The pseudo code shows that the systematic search of expanding neighborhoods for a local optimum is abandoned when an global improvement is achieved (shown with the `Break` jump).

Algorithm 1: Pseudo Code Listing for the Variable Neighborhood Search algorithm.

```
Input: Neighborhoods
Output: S_best
S_best ← RandomSolution();
while ¬ StopCondition() do
    foreach Neighborhood_i ∈ Neighborhoods do
        Neighborhood_curr ← CalculateNeighborhood(S_best, Neighborhood_i);
        S_candidate ← RandomSolutionInNeighborhood(Neighborhood_curr);
        S_candidate ← LocalSearch(S_candidate);
        if Cost(S_candidate) < Cost(S_best) then
            S_best ← S_candidate;
            Break;
        end
    end
end
return S_best;
```

6 Heuristics

- Approximation methods (such as stochastic hill climbing) are suggested for use as the Local Search procedure for large problem instances in order to reduce the running time.
- Variable Neighborhood Search has been applied to a very wide array of combinatorial optimization problems as well as clustering and continuous function optimization problems.
- The embedded Local Search technique should be specialized to the problem type and instance to which the technique is being applied.
- The Variable Neighborhood Descent (VND) can be embedded in the Variable Neighborhood Search as a the Local Search procedure and has been shown to be most effective.

7 Code Listing

Listing 1 provides an example of the Variable Neighborhood Search algorithm implemented in the Ruby Programming Language. The algorithm is applied to the Berlin52 instance of the Traveling Salesman Problem (TSP), taken from the TSPLIB. The problem seeks a permutation of the order to visit cities (called a tour) that minimized the total distance traveled. The optimal tour distance for Berlin52 instance is 7542 units.

The Variable Neighborhood Search uses a stochastic 2-opt procedure as the embedded local search. The procedure deletes two edges and reverses the sequence in-between the deleted edges, potentially removing 'twists' in the tour. The neighborhood structure used in the search is the number of times the 2-opt procedure is performed on a permutation, between 1 and 20 times. The stopping condition for the local search procedure is a maximum number of iterations without improvement. The same stop condition is employed by the higher-order Variable Neighborhood Search procedure, although with a lower boundary on the number of non-improving iterations.

```
MAX_NO_MPROVEMENTS = 50
LOCAL_SEARCH_NO_IMPROVEMENTS = 70
NEIGHBORHOODS = 1...20
BERLIN52 = [[565,575],[25,185],[345,750],[945,685],[845,655],[880,660],[25,230],[525,1000],
 [580,1175],[650,1130],[1605,620],[1220,580],[1465,200],[1530,5],[845,680],[725,370],[145,665],
 [415,635],[510,875],[560,365],[300,465],[520,585],[480,415],[835,625],[975,580],[1215,245],
 [1320,315],[1250,400],[660,180],[410,250],[420,555],[575,665],[1150,1160],[700,580],[685,595],
 [685,610],[770,610],[795,645],[720,635],[760,650],[475,960],[95,260],[875,920],[700,500],
 [555,815],[830,485],[1170,65],[830,610],[605,625],[595,360],[1340,725],[1740,245]]

def euc_2d(c1, c2)
  Math::sqrt((c1[0] - c2[0])**2.0 + (c1[1] - c2[1])**2.0).round
end

def random_permutation(cities)
  perm = Array.new(cities.length){|i|i}
  for i in 0...perm.length
    r = rand(perm.length-i) + i
    perm[r], perm[i] = perm[i], perm[r]
  end
  return perm
end

def stochastic_two_opt!(perm)
  c1, c2 = rand(perm.length), rand(perm.length)
  c2 = rand(perm.length) while c1 == c2
  c1, c2 = c2, c1 if c2 < c1
  perm[c1...c2] = perm[c1...c2].reverse
  return perm
end

def cost(permutation, cities)
  distance =0
  permutation.each_with_index do |c1, i|
    c2 = (i==permutation.length-1) ? permutation[0] : permutation[i+1]
    distance += euc_2d(cities[c1], cities[c2])
  end
  return distance
end

def local_search(best, cities, maxNoImprovements, neighbourhood)
  noImprovements = 0
  begin
    candidate = {}
    candidate[:vector] = Array.new(best[:vector])
    neighbourhood.times{stochastic_two_opt!(candidate[:vector])}
```

```
    candidate[:cost] = cost(candidate[:vector], cities)
    if candidate[:cost] < best[:cost]
      noImprovements, best = 0, candidate
    else
      noImprovements += 1
    end
  end until noImprovements >= maxNoImprovements
  return best
end

def search(cities, neighbourhoods, maxNoImprovements, maxNoImprovementsLS)
  best = {}
  best[:vector] = random_permutation(cities)
  best[:cost] = cost(best[:vector], cities)
  iter, noImprovements = 0, 0
  begin
    neighbourhoods.each do |neighbourhood|
      candidate = {}
      candidate[:vector] = Array.new(best[:vector])
      neighbourhood.times{stochastic_two_opt!(candidate[:vector])}
      candidate[:cost] = cost(candidate[:vector], cities)
      candidate = local_search(candidate, cities, maxNoImprovementsLS, neighbourhood)
      puts " > iteration #{(iter+1)}, neighborhood=#{neighbourhood}, best: c=#{best[:cost]}"
      iter += 1
      if(candidate[:cost] < best[:cost])
        best = candidate
        noImprovements = 0
        puts "New best, restarting neighbourhood search."
        break
      else
        noImprovements += 1
      end
    end
  end until noImprovements >= maxNoImprovements
  return best
end

best = search(BERLIN52, NEIGHBORHOODS, MAX_NO_MPROVEMENTS, LOCAL_SEARCH_NO_IMPROVEMENTS)
puts "Done. Best Solution: c=#{best[:cost]}, v=#{best[:vector].inspect}"
```

Listing 1: Variable Neighborhood Search algorithm in the Ruby Programming Language

8 References

8.1 Primary Sources

The seminal paper for describing Variable Neighborhood Search was by Mladenovic and Hansen in 1997 [9], although an early abstract by Mladenovic is sometimes cited [8]. The approach is explained in terms of three different variations on the general theme. Variable Neighborhood Descent (VND) refers to the use of a Local Search procedure and the deterministic (as opposed to stochastic or probabilistic) change of neighborhood size. Reduced Variable Neighborhood Search (RVNS) involves performing a stochastic random search within a neighborhood and no refinement via a local search technique. Basic Variable Neighborhood Search is the canonical approach described by Mladenovic and Hansen in the seminal paper.

8.2 Learn More

There are a large number of papers published on Variable Neighborhood Search, its applications and variations. Hansen and Mladenovic provide an overview of the approach that includes its

recent history, extensions and a detailed review of the numerous areas of application [5]. For some additional useful overviews of the technique, its principles, and applications, see [3, 6, 4].

There are many extensions to Variable Neighborhood Search. Some popular examples include: Variable Neighborhood Decomposition Search (VNDS) that involves embedding a second heuristic or metaheuristic approach in VNS to replace the Local Search procedure [7], Skewed Variable Neighborhood Search (SVNS) that encourages exploration of neighborhoods far away from discovered local optima, and Parallel Variable Neighborhood Search (PVNS) that either parallelizes the local search of a neighborhood or parallelizes the searching of the neighborhoods themselves.

9 Conclusions

This report described the Variable Neighborhood Search algorithm for global optimization and highlighted the family of extensions and applications that have sprung up around it.

10 Contribute

Found a typo in the content or a bug in the source code? Are you an expert in this technique and know some facts that could improve the algorithm description for all? Do you want to get that warm feeling from contributing to an open source project? Do you want to see your name as an acknowledgment in print?

Two pillars of this effort are i) that the best domain experts are people outside of the project, and ii) that this work is wrong by default. Please help to make this work less wrong by emailing the author 'Jason Brownlee' at jasonb@CleverAlgorithms.com or visit the project website at http://www.CleverAlgorithms.com.

References

[1] Jason Brownlee. The clever algorithms project: Overview. Technical Report CA-TR-20100105-1, The Clever Algorithms Project http://www.CleverAlgorithms.com, January 2010.

[2] Jason Brownlee. A template for standardized algorithm descriptions. Technical Report CA-TR-20100107-1, The Clever Algorithms Project http://www.CleverAlgorithms.com, January 2010.

[3] P. Hansen and N. Mladenovic. *Meta-heuristics, Advances and trends in local search paradigms for optimization*, chapter An introduction to Variable neighborhood search, pages 433–458. Kluwer Academic Publishers, 1998.

[4] P. Hansen and N. Mladenovic. *Handbook of Applied Optimization*, chapter Variable neighbourhood search, pages 221–234. Oxford University Press, 2002.

[5] P. Hansen and N. Mladenovic. *Handbook of metaheuristics*, chapter 6: Variable Neighborhood Search, pages 145–184. Springer, 2003.

[6] Pierre Hansen and Nenad Mladenovic. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467, 2001.

[7] Pierre Hansen, Nenad Mladenovic, and Dionisio Perez-Britos. Variable neighborhood decomposition search. *Journal of Heuristics*, 7(4):1381–1231, 2001.

[8] N. Mladenovic. A variable neighborhood algorithm - a new metaheuristic for combinatorial optimization. In *Abstracts of papers presented at Optimization Days*, 1995.

[9] N. Mladenovic and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, 1997.

[10] Jose Andres Moreno Perez, Nenad Mladenovic, Belen Melian Batista, and Ignacio J. Garcia del Amo. *Metaheuristic Procedures for Training Neutral Networks*, chapter 5: Variable Neighbourhood Search, pages 71–86. Springer, 2006.