CSE317-Artificial Intelligence
Solving static version of vehicle routing problem with time windows using HillClimbing

Hasan Masum

1805052

Department of Computer Science and Engineering,

Bangladesh University of Engineering & Technology

March 15, 2023

Contents

1 Introduction

The EURO Meets NeurIPS 2022 Vehicle Routing Competition[2] is a challenge that has been designed to encourage researchers and practitioners to develop algorithms for solving the vehicle routing problem with time windows (VRPTW). In the quickstart repository, a public dataset with 250 static VRPTW instances is provided. The baseline solver of the quickstart codebase internally uses Hybrid Genetic Search (HGS) to support time windows. With time windows, a vehicle must arrive at a customer between an earliest and latest arrival time, after which it requires a certain service time before it can continue to the next customer.

HGS is a population-based algorithm that combines elements of genetic algorithms (GAs) and local search to solve vehicle routing problems (VRPs). It maintains a pool (or population) with feasible and a pool with infeasible solutions. Initially, 100 random solutions are created, by using the SPLIT algorithm[5] on a random ordering of the customers and inserted in the right pool based on feasibility. In every iteration, two parents (feasible or infeasible) are selected from the pools using a binary tournament, which is combined using an ordered crossover[3] to create a new offspring solution, which is then improved using a local search.

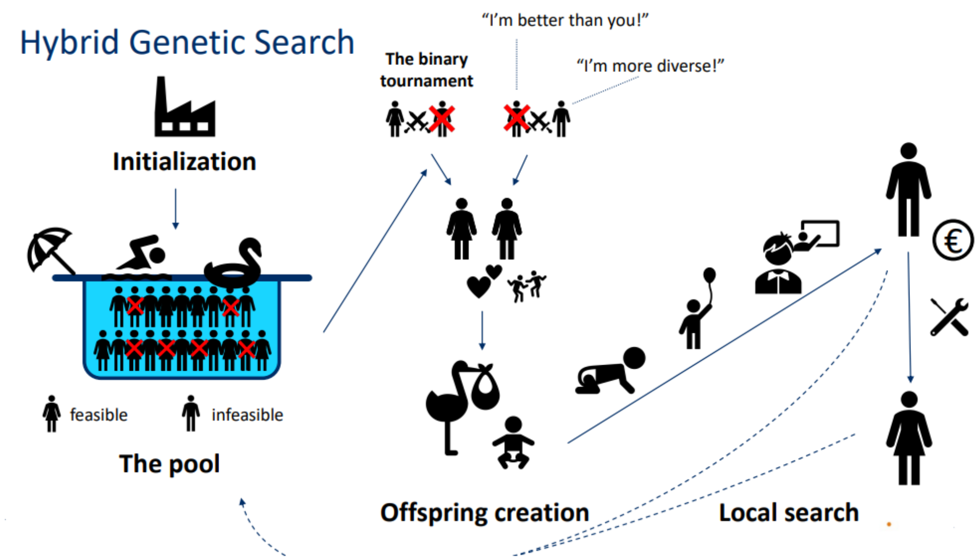

Figure 1: Hybrid Genetic Search(HGS).[1]

In this assignment, we implemented Simple Hill Climbing Local Search (SHC) in place of HGS which is a single-solution-based approach that iteratively explores the neighborhood of a given solution.

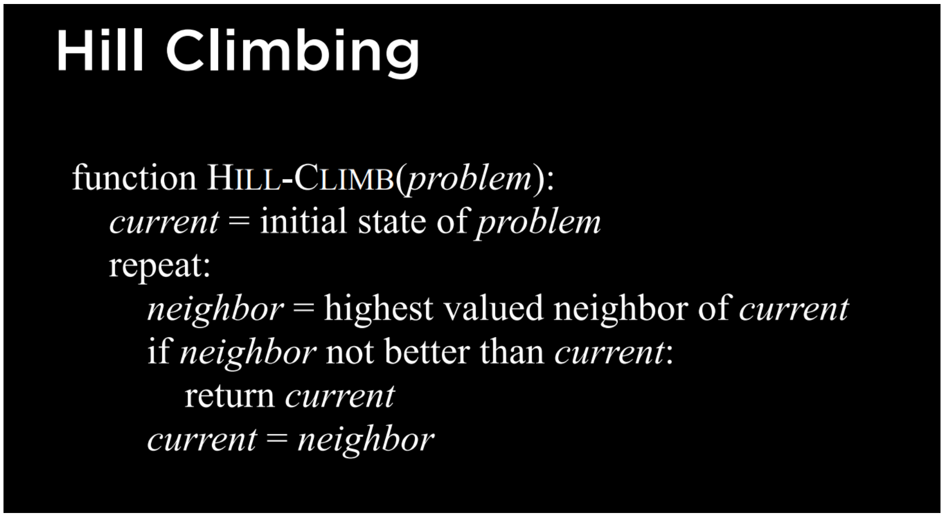

Figure 2: Simple Hill Climbing Local Search(SHC).[4]

2 Implementation

The Vehicle Routing Problem (VRP) is a combinatorial optimization problem that seeks to find the optimal set of routes for a fleet of vehicles to serve a set of customers while minimizing total travel distance or time. The problem is NP-hard, and exact algorithms are often impractical for large instances. Local search algorithms, such as hill climbing, can be used to find high-quality solutions efficiently.

2.1 Hill climbing

For implementing the hill climbing we added a method `Genetic::hill_climbing()` in `Genetic.h`. The method is implemented in `Genetic.cpp` file.

```
// baselines/hgs_vrptw/Genetic.cpp
...
void Genetic::hill_climbing(){
    // Generate an inital solution
    Individual *initial_soln = population->getBestFound();

    /* LOCAL SEARCH */
    //Generate a new solution by making a small change to the current solution
    localSearch->run(initial_soln, params->penaltyCapacity, params->penaltyTimeWarp);

    // add to population(best solution from the population is exported in main.cpp)
    population->addIndividual(initial_soln, true);
}
...
```

Here is the approach to solving the VRP using hill climbing:

- **Step 1-Generate Initial Solution:** We need to start with an initial solution that assigns each customer to a vehicle and constructs a set of routes that satisfy the capacity and distance/time constraints. Here we use existing HGS implementation for the initial solution. The HGS implementation in the quickstart codebase generates 100 initial solutions in `baselines/hgs_vrptw/Population.cpp`'s

`Population::generatePopulation()` method using *nearest*, *farthest*, *sweep*, and *random* strategy. This population of the initial solution is used for offspring generation in HGS. To generate an initial solution(instance of `Individual` which is an object to represent one individual/solution of a population) for hill climbing, we use `Population::getBestFound()`.

- **Step 2-Local Search:** We need to Generate a set of neighboring solutions by applying small perturbations to the current solution. We use the existing implementation of local search by calling the function `LocalSerach::run()` which improves the initial solution by making changes using swap, relocate moves between near neighbors, swap, 2-opt, 2-opt*, and swap*.
- **Step 3-Save the final solution:** To save and export the solution we add the individual(solution) by calling the method `Population::addIndividual(const Individual* indiv, bool updateFeasible)`. The best solution from the population is exported in `main.cpp`.

```
// baselines/hgs_vrptw/main.cpp
...
// Main class of the algorithm. Used to read from the parameters from the command line,
// create the structures and initial population, and run the hybrid genetic search
int main(int argc, char* argv[]){
    ...
        // Export the best solution, if it exist
    if (population.getBestFound() != nullptr)
    {
        population.getBestFound()->exportCVRPLibFormat(commandline.config.pathSolution);
        population.exportSearchProgress(commandline.config.pathSolution + ".PG.csv',
            commandline.config.pathInstance, commandline.config.seed);
        if (commandline.config.pathBKS != "")
        {
            population.exportBKS(commandline.config.pathBKS);
        }
    }
    ...
}
```

2.2 Additional changes

For generating data some additional changes are made in the quickstart codebase. For running the static instances with hill climbing an argument `--hill_climbing` is added in `solver.py`[1]. A flag variable is also added in `Params.h`[2] for the same purpose.

[1] https://github.com/hmasum52/CSE317-AI-assignment/commit/f91773d06f8e2837ef0fb39c4ccf4ba0de125231#diff-b57eb9e315f8ca91758296cb3d16227c356e87705e0df982fb66eaf799261040

[2] https://github.com/hmasum52/CSE317-AI-assignment/commit/f91773d06f8e2837ef0fb39c4ccf4ba0de125231#diff-e03cc0b3a96034f5a966a1d185608603684821305ba9c2cbd316e17c763ddde3

3 Data

`my-test.sh`[3] script is used to generate data for the first 20 static instances for both Hybrid Genetic Search(HGS) & Hill climbing and the output is saved inside `data.csv` file.

File name	**HGS**	**Hill Climbing**
ORTEC-VRPTW-ASYM-00c5356f-d1-n258-k12.txt	353499	359653
ORTEC-VRPTW-ASYM-01829532-d1-n324-k22.txt	726208	1328705
ORTEC-VRPTW-ASYM-02182cf8-d1-n327-k20.txt	448859	454021
ORTEC-VRPTW-ASYM-04c694cd-d1-n254-k18.txt	418223	447167
ORTEC-VRPTW-ASYM-0797afaf-d1-n313-k20.txt	362638	371527
ORTEC-VRPTW-ASYM-08d8e660-d1-n460-k42.txt	391376	401877
ORTEC-VRPTW-ASYM-0bdff870-d1-n458-k35.txt	511092	514397
ORTEC-VRPTW-ASYM-0dc59ef2-d1-n213-k25.txt	432234	449283
ORTEC-VRPTW-ASYM-11527044-d1-n470-k27.txt	537544	547455
ORTEC-VRPTW-ASYM-13db18b2-d1-n310-k20.txt	907453	1373607
ORTEC-VRPTW-ASYM-152bab99-d1-n215-k20.txt	393512	652106
ORTEC-VRPTW-ASYM-16b82253-d1-n457-k30.txt	452731	463311
ORTEC-VRPTW-ASYM-19eafe1e-d1-n458-k35.txt	440552	446034
ORTEC-VRPTW-ASYM-1a452a2c-d1-n391-k23.txt	558446	766251
ORTEC-VRPTW-ASYM-1bc16246-d1-n503-k43.txt	470283	697353
ORTEC-VRPTW-ASYM-1bdf25a7-d1-n531-k43.txt	469301	474392
ORTEC-VRPTW-ASYM-1cd538a9-d1-n400-k25.txt	508561	516664
ORTEC-VRPTW-ASYM-1de83915-d1-n262-k15.txt	277923	286402
ORTEC-VRPTW-ASYM-1f1fffc4-d1-n332-k25.txt	281265	290333
ORTEC-VRPTW-ASYM-21e8376e-d1-n507-k30.txt	453249	464728

Table 1: Cost of solution for first 20 static dataset for both Hybrid Genetic Search(HGS) & Hill climbing

[3] `https://github.com/hmasum52/CSE317-AI-assignment/commit/f91773d06f8e2837ef0fb39c4ccf4ba0de125231#diff-5a67dcdf7d3b121fce8e88eca31e2ff09abef92d66feb0990b352c2a23333d2f`

4 Conclusion

From the data above we can see that HGS can potentially find better solutions. It is because HGS can search the can explore a wide range of solutions, even if the initial solutions are far from optima, whereas Simple Hill Climbing(SHC) only explores the local neighborhood of a given solution. Again, HGS tends to converge to a near-optimal solution faster than SHC because it maintains a diverse population of solutions throughout the search process. SHC, on the other hand, may get stuck in local optima.

5 Source code

Here is the link to the source code repository: https://github.com/hmasum52/CSE317-AI-assignment

References

[1] Wouter Kool. *Solving Vehicle Routing Problems with Time Windows.* `https://wouterkool.github.io/pdf/slides-hgs-vrptw.pdf`. Accessed on March 14, 2023. 2021.

[2] Danilo Numeroso et al. "EURO Meets NeurIPS 2022 Vehicle Routing Competition". In: ().

[3] IM Oliver, DJd Smith, and John RC Holland. "A study of permutation crossover operators on the traveling salesman problem". In: *Genetic Algorithms and their Applications.* Psychology Press. 2013, pp. 224–230.

[4] Harvard University. *Week 3 Optimization - CS50's Introduction to Artificial Intelligence with Python.* 2020. URL: `https://cs50.harvard.edu/ai/2020/notes/3/`.

[5] Thibaut Vidal. "Split algorithm in O (n) for the capacitated vehicle routing problem". In: *Computers & Operations Research* 69 (2016), pp. 40–47.