

GRuVeR: A General Ruby library for solving Routing Vehicle Problems

Abstract

Routing Vehicle Problems are a generic class of NP-hard computation problems which all have in common the same target: how to find the best route in terms of speed, cost and/or efficiency among all possible ones, taking into account the amount of vehicles available, stops needed, available routes and other additional constraints. Although there are several variations of this problem with different levels of difficulty each, they all iterate around this basic principle.

RVPs are really common in organizations that provide transportation and delivery services such as the post and courier services, food and diary distribution or even the kindergarten, which has to find a short route in order to take kids back home. Routing Vehicle Problems can also be found in various computer science and telecommunications topics like network routing. It is obvious that having a way to find the “best” possible route in an acceptable amount of time is more than necessary.

What I propose is an implementation of an open source ruby library, which will be able to solve such problems efficiently given the necessary input (nodes, routes, costs, etc) and optional constraints. It will be based on genetic algorithms and it could also facilitate heuristics and constraint programming, using existing ruby libraries, such as *charlie* and *gecode/r*.

So, as soon as this project is finished, it will be easy to integrate it in existing or new applications with vehicle routing needs. This will fill the gap that there is today, since no such open library exists.

Description

Vehicle Routing Problems and variations

The basic VRP is pretty simple: given a set of routes, depots, vehicles and a number of geographically dispersed cities or customers how can we deliver the products to all customers on the minimum cost? But since in most cases several constraints must be met there is a number of variants of the basic and general problem. These constraints may be:

- Every vehicle has a limited capacity (Capacitated VRP - CVRP)
- Every customer has to be supplied within a certain time window (VRP with time windows - VRPTW)
- The vendor uses many depots to supply the customers (Multiple Depot VRP - MDVRP)
- A vehicle may have to collect items to return to the depot (VRP with Pick-Up and Delivering - VRPPD)
- A vehicle may be able to unload only the last loaded item because of size / packing (VRP with LIFO)

- The customers may be served by different vehicles (Split Delivery VRP - SDVRP)
- Some values (like number of customers, their demands, serve time or travel time) are random (Stochastic VRP - SVRP)
- The deliveries may be done in some days (Periodic VRP - PVRP)

Project targets

Setting a realistic target for what we want this project to deliver in the end of GSoC is more than crucial. So, instead of saying that everything will get implemented in 3 months it is way more useful to define a smaller set of features that we target for this available time. Before that, some general guidelines that I have in mind are:

- Write the code in a way that it will be easily extensible afterwards.
- Prioritize on one specific RVP variation implementation: Solve one first, then move to the other. When the next one is finished refactor and optimize the all the previous if possible (based on the experience gained).

Based on these, I believe that setting, as primary goals, the following tasks is realistic:

- Solve at least 2 variations of the RVP class of problems (for example Vehicle Routing Problem with Time Windows and Capacitated Vehicle Routing Problem)
- Provide a way for multiple input and output data support (for example a matrix in ruby code, a yaml file, etc...).

Additionally, a nice and possibly wanted feature might be defining (or implementing) a way for providing geospatial data as input and output. So it would be possible for example to only give the coordinates of the stops and get a result based on distance by default, without giving any more parameters. Then it would be easy to export the data and plot them on a mapping application.

Methodology that will be followed

As mentioned earlier, the main part of the implementation will use genetic algorithms and genetic programming in general. For this the charlie library will be used since it provides a nice set of features for implementing genetic algorithms in ruby. Additionally, the possibility to also use constraint programming will be investigated and it will be based on the Gecode/R library.

The focus will be on the official ruby 1.8.6 implementation but I will also try to make it compatible with most of the current alternatives if possible. This can help exposing differences between them in terms of speed.

Other libraries that will be used are RSpec in conjunction with ZenTest for the specification / testing part. Since Behavior Driven Development is my preferred way of writing code, it will be followed in most parts, since it makes easier to test the code base, eliminate bugs early and write documentation.

During the development phase I plan to do small and targeted commits on the code repository thus making it easier for both the mentor and me to follow the

evolution of the code and also allowing my mentor or any other person involved (community-wise) to provide feedback and suggestions.

Benefits

The benefits from this project will be multiple and diverge.

First of all it would result in the creation of a free and open RVP library in Ruby that could be used and extended from the community itself as an alternative to proprietary/closed source implementations that already exist. Even better it could be the first choice among the other ones, at least for Ruby developers.

Secondly, since using existing Ruby libraries, the whole procedure could propagate “backwards” feedback, extensions and modifications to all engaged projects and even Ruby itself (including different ruby implementations like rubinius, jruby, Ruby 1.9).

Last but definitely not least, this project (and projects like this in general) can really help bringing Ruby and Ruby-based projects (Rails, Merb and such) closer to enterprises by providing enterprise-scoped tools in a free as in speech manner. This will enable organizations to customize, build-upon and use this particular project (and ruby in general) while providing feedback, new solutions and possible patches back to the projects’ code base.

Deliverables

- Source code with gem packaging
- Documentation (including RDoc and usage examples)
- Library specifications (RSpec) and/or tests (Unit::Test)
- Website supporting all of the above
- Weekly blog posts about updates and progress, in order to facilitate a community around the project

Timeline

Phase 1 (before official start):

Gather all necessary research work and discuss the direction that should be followed during development with the mentor. Also set up the necessary environment including a bug-tracker (possibly Redmine) and a blog.

Phase 2a (official start - midterm evaluation):

Development phase - specifications and implementation at the same time (BDD):

Implement the first RVP variant.

Phase 2b (midterm evaluation - suggested pencils down date):

Development phase - specifications and implementation at the same time (BDD):

Implement the second RVP variant and the various data input/output helpers.

Phase 3 (suggested pencils down date - firm pencils down date):

Write documentation, usages examples and final evaluation. Also package library as a ruby gem in Rubyforge.

Possible variations in timeline:

Good case:

If things are going well there are chances that a gem (and so an early 0.1 public version) might be released before GSoC ends and more than 2 variants are implemented. This way I could get some feedback from the community to make further improvements and changes.

Bad case:

Of course there is always the possibility that things can go wrong. In this case in the final evaluation main emphasis will be given in the detailed description and explanation of the difficulties that were faced during the development phase and which led to this unexpected and unsatisfactory result. This way, the evaluation could be used as a basis for redesign and further brainstorming on the project itself, hopefully leading in an implementation outside the scope of GSoC.

Related work

Research work

There is a plethora of research work in this particular topic since this is a problem with both practical and scientific interest and the solutions proposed vary both in terms of implementation approach and results. What interest us more are approaches based mainly on genetic algorithms with optional usage of heuristics and / or other methods for optimization of result or computational speed. Mentioning all or most of these are out of the scope of this proposal but some are given in the references. Also, as mentioned later (in Timeline), part of the preparation that will take place for this project is a more detailed examination of most of them, so the most appropriate ones can be combined and used.

Implementations

At the time being only closed source and commercial implementations do exist. Also all of them are either in binary format (executables) or in Java / J2EE. A commercial example is JOpt.SDK and a freeware one is OR-Objects. A more comprehensive list is available at http://www.lionhrtpub.com/orms/surveys/Vehicle_Routing/vrss.html.

Background

I am an undergraduate with a major on Telecommunications Science & Technology, currently on my last year of studies. Throughout my studies I have mostly given emphasis on programming, networking and signal processing and taken courses in subjects as OOP (Java and C/C++ programming), network and distributed services and applications, pattern recognition and image/sound analysis and successfully implemented various programming tasks in these subjects.

Over the last two years I have been constantly using Ruby and Rails for personal projects and while working part-time (mainly on my summer break) for a start-up company composed of young people which use and contribute on open source projects on a regular basis. Through this I had the opportunity to be engaged in the Ruby, Rails and other libraries' code base allowing me to

understand their concepts even better as well as open source's ecosystem, ideas, principles and procedures.

Motivation

I'm really an advocate of Open Source tools, development and ideals and really passionate with problem-solving programming and thinking. It is my belief that bringing more quality and useful tools, frameworks and libraries to the Open Source Community can push it even further to a wider audience, composed by enthusiasts, end-users and to enterprises. I feel that I have the necessary background, motivation and will in order to successfully complete this project and this way I can give back to the community from which I have taken so much thus far while having the chance to gain invaluable experience in subjects that really interest me.

Contact Details

My name is Nikos Dimitrakopoulos and I can be contacted via email at demisone@gmail.com. I also use the nickname demisone in Freenode.

References

General info on RVP:

Vehicle Routing Problems, IDSIA - Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, Monaldo Mastrolilli:

<http://www.idsia.ch/~monaldo/vrp.html>

The VRP Web:

<http://neo.lcc.uma.es/radi-aeb/WebVRP/>

Related work (implementations):

Vehicle Routing Software Survey, OR/MS Today, February 2008:

http://www.lionhrtpub.com/orms/surveys/Vehicle_Routing/vrss.html

JOpt.SDK - Vehicle Routing Software Library:

<http://dna-evolutions.com/joptsdk.html>

Relevant research work (some examples only):

The Vehicle Routing Problem, Paolo Toth, Daniele Vigo, 2001:

<http://www.ec-securehost.com/SIAM/DT09.html>

A New Genetic Algorithm for VRPTW, KQ Zhu, 2000:

<http://citeseer.ist.psu.edu/311264.html>

Hybrid Genetic Algorithm, Simulated Annealing and Tabu Search Methods for Vehicle Routing Problems with Time Windows, Thangiah et al., 1998:

<http://osiris.tuwien.ac.at/~wgarn/VehicleRouting/neo/data/articles/vrptw4.pdf>

Relevant Ruby libraries:

Charlie (most possible to be used):

<http://charlie.rubyforge.org/>

Gecode/R:

<http://gecoder.rubyforge.org/s>

General Genetic Algorithms for Ruby:

<http://gga4r.rubyforge.org/>

Artificial Intelligence for Ruby :: ai4r:

<http://ai4r.rubyforge.org/index.html>