Satyam Chand Deopa

11806590

RKM103B51

Submitted By:

Submitted to:

Ms. Pooja Rana

20992

**INT246**

Soft Computing Techniques

**Term Project**

Traffic light optimization   
using Particle Swarm Intelligence

**Abstract**

Well-timed traffic light can tremendously decrease vehicles’ time spent at red lights, but timing the lights well is a very complex, difficult problem to solve. In this paper, we implement swarm optimization method for optimizing traffic light behavior, which allow us to minimize a function with knowing its analytic form. We apply particle swarm optimization (PSO) in our simulation. This method yields decrease in average total travel time for vehicles.

**Introduction**

Traffic lights are a fixture of everyday life with seemingly simple governing rules but with complex effects on traffic patterns. As thousands of hours are wasted everyday waiting in slow traffic or for red lights, a question of great practical application is how to design a traffic light’s behavior or a system of traffic lights so as to minimize the amount of time wasted in traffic. Ideally, we would want to model the problem formally to guide the optimization process and get theoretical guarantees on any solution found. However, in general it is very difficult to model a traffic system, as there are a huge number of variables: road layout, traffic volume, traffic light behavior, etc. Therefore, we attempt to optimize without doing so. Particle swarm optimization is an optimization technique that falls into the broad category of metaheuristics. As such, it makes no assumptions about the function to be optimized, but come at the cost of having no guarantees on the quality of solution found or time needed. The approach utilizes a group of agents, a "swarm", that each explore the solution space and interact with each other to search intelligently. The rest of this paper is organized as follows: we describe the problem of traffic optimization and formally define particle swarm optimization. Finally, we describe our experiment and discuss results.

**Problem Description**

As we are interested in optimization techniques for arbitrary traffic systems, we only seek to optimize the network with respect to the behavior of the traffic lights in the system. We have modeled this problem as follows: we model the traffic light behavior as a 2-dimensional vector, where we have n features that determine a traffic light’s behavior and m traffic lights in the system. For example, a straightforward feature representation is to have a 4 phase light cycle, and a single light is represented by 4 numbers, each corresponding to the duration of a phase in the cycle. If our traffic system has 25 lights, then a behavior vector for this entire traffic light system is in R25×4. We want to find min x f(x) Note that we make no assumptions about the analytic form of f and make no attempts to model the underlying traffic system. Instead, we take f to be some sort of black box function that can be evaluated, but we know nothing else about it. As such, we have very little to work with in optimizing f, and therefore must turn to extremely general optimization methods.

**Particle Swarm Optimization (PSO)**

Particle swarm optimization (PSO) is a metaheuristic optimization algorithm that makes very few assumptions about the function to be optimized, and thus is suitable for solving a broad class of functions, including complicated non-convex function. Inspired by the social behavior of various animals, PSO uses a swarm of particles, each associated with a position in the solution space and a velocity. The algorithm consists of alternating between calculating the velocity based on the performance found across all neighbors and updating the position based on the particle’s velocity. The particles tend to cluster together in the search space, thereby forming a swarm.

We define the map as a plane on which intersections are scattered uniformly. There exist bounds on the length and breadth of the map.

Here, the function we are trying to minimize using swarm is the distance between the current position and the target point. This can be given by:

f(x)=(xf – xp)­­­­­­­­­2 + (yf – yp)2

The particles are scattered on the map and the path of the best particle is selected for the updating the time intervals.

For the current execution, we disperse 4 particles or boids to search for the minimum of the above function. The personal best that a single particle can look upto can be given by the mentioned function by first finding the derivative of the point and moving in that direction upto a single unit. We keep track of the global best using another variable which is updated in parallel. We use the synchronous technique for this problem as asynchronous technique is known to mature much faster and leaves less chances of variation between two boids or particles.

To avoid the chances of path getting outside the vicinity of map(or city), bounds are applied which are the same as the size of the map.

PSO for Traffic Lights Particle swarm optimization pairs nicely with traffic light optimization problem. The main decision is how to represent a traffic light’s behavior as a vector of real numbers. An intuitive formulation we explore is to fix the order of the n light phases in the cycle, and then represent each traffic light as a vector where each component represents the length of the i-th phase. Furthermore, in practice, traffic lights stay on a phase at most roughly a few minutes, depending on the intersection, and at least a few seconds. Thus, for our simulations, we impose a restriction that each component must be within some minimum time tmin=1. This fits naturally into the PSO algorithm as the algorithm allows us to bound the search space we want to explore. Another possible representation of a traffic light includes the initial phase in the cycle of light phases. The difficulty here is that intuitively this would be represented as a discrete value, but PSO assumes continuous values. To accommodate for this, we let the initial phase be a continuous value in some range, and then round to the nearest. This is a general method that allows us to use PSO for discrete valued inputs. It is not immediately clear what the interpretation of this discrete-to-continuous relaxation is, but one possible explanation is that the continuous value is a measure of how certain the algorithm is of assigning that discrete value.

**Algorithm:**

Initialize vehicles

For each vehicle:

Create particles with random velocities and same target

While target is not reached:

Find pbest for each particle

Update particle position and velocity w.r.t. pbest, gbest and previous velocity(momentum)

Find the gbest

Save path of best particle

Using path of all vehicles update timing control

**Simulator and Traffic Map Details**

The map consists of a 2-d list, where each intersection consists of 4 lanes (East, west, north and south).

The movement of a vehicle is simulated by selecting an active lane (i.e. a lane with green light) and selecting the first vehicle present in that lane. This vehicle chooses the next lane it wants to go towards. This vehicle is then removed from the list and added to the next intersection. At each iteration the counter for the lane lowers by 1 and after it reaches 0, next lane is selected in next iteration. The vehicle is removed from the simulation when it reaches its destination.

**Some common variables used:**

veh : contains the list of vehicles with random initial and final points

veh

|  |  |  |
| --- | --- | --- |
| Initial x, y coordinates | Destination x, y coordinates | Vehicle no. |

vehpos: contains the current position of vehicle inside the simulation

vehpos

|  |  |  |
| --- | --- | --- |
| Current x, y coordinates | Destination x, y coordinates | Vehicle no. |

boid\_paths: contains the path of best particle for a vehicle

lanes: a list of intersections. Each intersection contains four lanes which are lists

each intersection has 4 lists for each direction

|  |  |  |  |
| --- | --- | --- | --- |
| North | East | South | West |

signal: control timer for each intersection

signal

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Time remaining | Selected lane for an intersection | Time range for North lane | Time range for East lane | Time range for South lane | Time range for West lane |

cost= total cost of the simulation

**Result**

The simulation upon implementing PSO is more than 5% faster on an average than in a map where all lanes in an intersection has same time interval.

Link :