Multi-Lane Automated Intersection Management

Edward Lambert

July 23, 2021

1 Introduction

The wider motion coordination problem faced by fleets of identical material transfer AGVs is described in Problem 1. AGVs are used to complete missions at different locations in a network of connected paths. Using the 2-level decomposition of Digani et al [1], Problem 1 can be broken down into a high level zone capacitated routing problem and a local coordination sub problem within each zone. Autonomous Intersection Management (AIM) is one promising approach to solving the local coordination sub problem.

Problem 1. Given a number $N < \bar{N}$ of AGV operating on a fixed network of paths $[\pi_0, ...\pi_N]$, what trajectory should each one follow to allow all the missions to be completed in the minimum AGV-Time while a safe distance is maintained between each one at all times.

Previous studies have shown the use of AIM can lead to reduced travel time compared to negotiation based conflict resolution, while being computationally tractable for larger numbers of vehicles than centralized control.

Questions remain about the importance of searching alternative crossing orders, compared to the additional computational cost. Better solution time guarantees due to a fixed order could enable AIM to operate without a backup system in more situations.

2 Aim

To compare the performance of an intersection manager with fixed First-In-First-Out (FIFO) crossing order with a published method based on a non-convex optimization with Linear objective and Quadratic constraints (Quad_constr), in which the crossing order can be varied to improve the objective, which is to minimize the sum of crossing time for a set number of participants. The numerical performance of the intersection manager is also important for its use as a real-time safety critical system including the scaling with increasing numbers of approaching vehicles.

3 Hypothesis

The *Quad_constr* approach is expected to find solutions with a lower total travel time. Freedom to vary the crossing order has the potential to significantly reduce total travel time in a multi-lane intersection by allowing sets of vehicles whose path do not cross to cross the intersection together.

The *Quad_constr* approach should match $FIFO$ performance if that is the best ordering, or in other cases improve upon it. Both approaches make the same simplifying assumptions about the constraints, maximize the same objective by varying the same control waypoints.

The *Quad_constr* approach must search a non-convex space, so it is expected to take longer to solve. It is also expected to have an unpredictable solution time, which may limit the scale at which is can be applied, and possibly rule out its use as a standalone collision avoidance method at any scale. The execution time is expected to be shorter and more consistent for the $FIFO$ approach, as the linear program can be solved efficiently.

4 Method

4.1 Problem Representation and Assumptions

The objective is to minimize the total travel time for all AGV to complete their missions. At this stage the missions just consist of reaching the end of their assigned path π_i.

The path π_i can be divided based on longitudinal distance s into three parts. The first part γ_i approaching conflict where $s < s\_begin$, a part traversing the conflict zone β_i where $s\_begin < s < s\_end$, and a part leaving the intersection β_i where $s > s_{end}$.

A1 No external obstacles can be found on the roadmap

A2 Waypoint instructions are only sent with sufficient approach distance remaining to adjust speed before reaching the first waypoint $s\_begin - s > min\_conflict\_dist$

A3 The conflict zone must be long enough to reach the second waypoint at the right time without violating acceleration limits $s\_end - s\_begin > min\_conflict\_dist$

A4 Followers on path π_i only begin to move after the lead AGV has cleared the intersection

A5 Each AGV has a unique initial and final position

A6 The full set of paths through the intersection is known at initialization time

A7 All waypoints can be reached with an average speed $v < \hat{v}$

Assumption A4 ensures that car-following behaviour does not factor in the following analysis. In the test system arrivals were limited at the source, which counted AGVs

already present on any of the associated lane alternatives, and if the lane capacity was exceeded further arrivals were stacked in a vertical queue.

Based on A6 we can compute the shape of the conflict zone. This is defined in path coordinates by the earliest intersection point with any other path *s_begin* and the latest intersection point *s_end*. As AGVs have some bodywork extending around their control point, the conflict must be expanded by the diameter of the bounding circle. In this way, an AGV waiting outside the conflict zone can never collide with one inside.

Assumption A2 means the optimization problem can be solved for a bounded average speed in each region. the upper bound is set based on the maximum acceleration parameter of the AGV model.

The collision avoidance constraints can be expressed in terms of the arrival time a_i of one and the departure time d_j of the other of a pair of approaching AGV in conflict. For safe crossing between AGV i and j we require the condition in Equation 1 holds.

$$a_i > d_j \vee a_j > d_i \tag{1}$$

The only difference between the two AIM approaches is the way they transform this condition into constraints on a standard form optimization which can be solved with convex methods. One uses a fixed order leading to to linear constraints and the other permits any ordering through quadratic constraints.

Both optimize over the same parameter vector $\boldsymbol{\phi}$ as defined in Equation 2. This contains a stack of pairs of reciprocal average speeds. One for section α and one across section β.

$$\boldsymbol{\phi} = \begin{bmatrix} 1/v_{\alpha,0} \\ 1/v_{\beta,0} \\ \vdots \\ 1/v_{\alpha,0} \\ 1/v_{\beta,0} \end{bmatrix} \tag{2}$$

The constraint on approaching vehicles from those which are already past their own decision point is the same for both approaches. Every approaching AGV must have an arrival time $(s_i - s\_begin_i)\phi_i$ greater than the latest departure time of any crossing vehicle d_j. Setting $t_p = \max_j d_j$ and $\boldsymbol{t_p} = t_p \times \mathbf{1} \in R^{(1\times N)}$ to create N constraints from the single value of t_p this can be expressed as in Equation 3.

$$\begin{bmatrix} s_0 - s\_begin_0 & 0 & \ddots & \mathbf{0} \\ \vdots & & \ddots & \vdots \\ \mathbf{0} & \ddots & s_N - s\_begin_N & 0 \end{bmatrix} \boldsymbol{\phi} \leq \boldsymbol{t_p} \tag{3}$$

4.2 FIFO

For this method the pairs in $\boldsymbol{\phi}$ must be sorted by the remaining distance along path α before the start of the conflict $s - s\_begin$. As the conflict zone extent is a fixed property of the intersection this distance will not change when other vehicles arrive on new paths,

so the ordering is stable. After sorting, index $i+1$ will be further from the decision point than index i. Then Equation 1 can be expressed as

$$\begin{bmatrix} s_0 - s\_begin_0 & s_0 - s\_end_0 & \ddots & \mathbf{0} \\ \vdots & & \ddots & \vdots \\ \mathbf{0} & \ddots & s_N - s\_begin_N & s_N - s\_end_N \end{bmatrix} \boldsymbol{\phi} \leq \begin{bmatrix} a_0 \\ d_0 \\ \vdots \\ a_N \\ d_N \end{bmatrix} \tag{4}$$

Care should be taken to exclude from $\boldsymbol{\phi}$ those AGV which have already passed the decision point defined as $s\_begin - min\_conflict\_length$. The motion of these vehicles can no longer be altered because the on-board longitudinal controller would likely be unable to meet the new waypoint times due to physical limits on acceleration.

4.3 Quad_constr

The constraints can be expressed without imposing a fixed order by transforming the reserved time blocks into centroid α and half width β representation. This leads to Equation 5 for the constraint between any pair of AGVs (i, j).

$$|\alpha_i - \alpha_j| > \beta_i + \beta_j \tag{5}$$

And equivalently Equation 6.

$$(\alpha_i - \alpha_j)^2 - (\beta_i + \beta_j) > 0 \tag{6}$$

In terms of the parameter vector $\boldsymbol{\phi}$, this can be captured in a block diagonal Hessian H_{ij}, containing a block Λ_{ij} for each pair as shown in Equation 7.

$$\Lambda_{ij} = \begin{bmatrix} A_i \mathbf{1}_i \mathbf{1}_i^T A_i & -A_i \mathbf{1}_i \mathbf{1}_j^T A_j \\ A_j \mathbf{1}_j \mathbf{1}_i^T A_i & A_j \mathbf{1}_j \mathbf{1}_j^T A_j \end{bmatrix} \tag{7}$$

$$\boldsymbol{H}_{ij} = \begin{bmatrix} \boldsymbol{\Lambda}_{ii} & \boldsymbol{\Lambda}_{ij} \\ \boldsymbol{\Lambda}_{ji} & \boldsymbol{\Lambda}_{jj} \end{bmatrix} \tag{8}$$

The quadratic constraints are then expressed as $\boldsymbol{H}\boldsymbol{\phi} > \mathbf{0}$.

5 Numerical Experiment

The two intersection managers were implemented in Python with an identical messaging interface to communicate with a collection of AGV controllers with simplified dynamics.

The FIFO speeds were found with scipy.optimize.linprog and the vehicles ordered with the Python 3.7.6 function sorted

The Quad_constr speeds were found with scipy.optimize.minimize given the analytical Jacobian and Hessian. The speeds were converted into timed waypoints which the AGV

controllers must meet as closely as possible. This conversion takes the constant speed solution for the intersection problem and retains only the safety critical part at the entry and exit of the conflict zone to pass to the AGV controllers. Vehicles with second order dynamics were also simulated.

The first test is inspired by the published results in [2], showing the improvement of intersection management similar to Quad_constr over decentralized negotiation. The tests took place on a close duplicate of three realistic intersection layouts with fixed direction lanes.

Over ten runs, one AGV was initialized at each entrance, and assigned a random exit. The clearing time $Tclearing$ for the last vehicle to leave the intersection is recorded in each run. The execution time to calculate the optimal waypoint times for every vehicle was calculated at the start $Texec$. The worst delay for a single AGV compared to its free flow time to cross the intersection $Twait$ is also recorded. $Twait$ is defined somewhat differently to Digani 2019. In that paper the metric captures the time spent stationary so the optimal method has zero waiting time in every case. The waiting time is averaged over different AGV in each run, and the worst run average is recorded for that layout.

$$Twait = crossing\_time - \frac{PathLength}{max\_speed} \tag{9}$$

In the table below $Twait$ is calculated according to Equation 9 so it is likely to be non-zero, even for optimal methods. It represents the delay caused by the intersection compared to free flow speed. In some cases it is slightly negative, because vehicles can exceed the maximum speed to meet the waypoints given by the intersection manager.

5.1 Real World Intersection Designs 1,3,4 from Literature

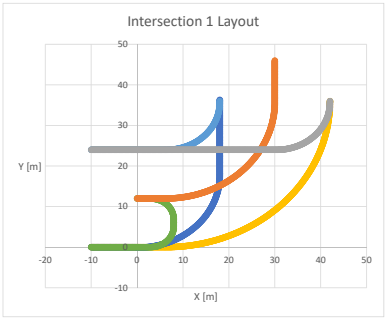

Figure 1: Waypoints with 10cm spacing for each the six paths through Intersection 1.

6 Numerical Results

Execution time results in Figure 6 are for the $SLSQP$ algorithm, running on a 1.6Ghz Intel(R) Core(TM) i5-8250U Quad Core with 16 GB main memory.

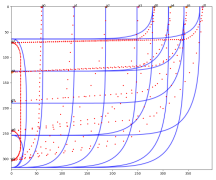

Figure 2: Intersection 3 Layout including eight arrival nodes. Reproduced waypoints as tested shown as red dots.

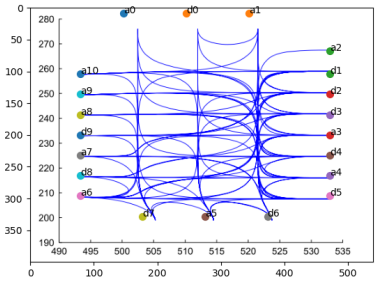

Figure 3: Intersection 4 reproduced from [2] with approximate arrival a and departure d nodes overlaid.

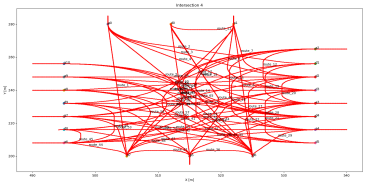

Figure 4: Intersection 4 detailed waypoints using Reeds-Shepp paths [3] to join the 66 $a - d$ pairs.

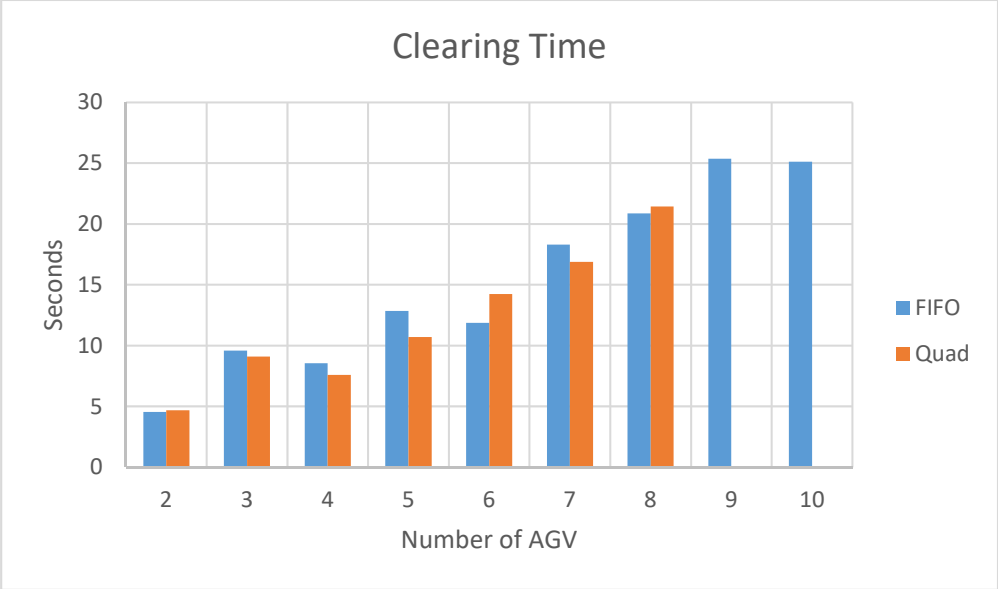

Figure 5: Tclearing is the time for the last vehicle to cross Intersection 4.

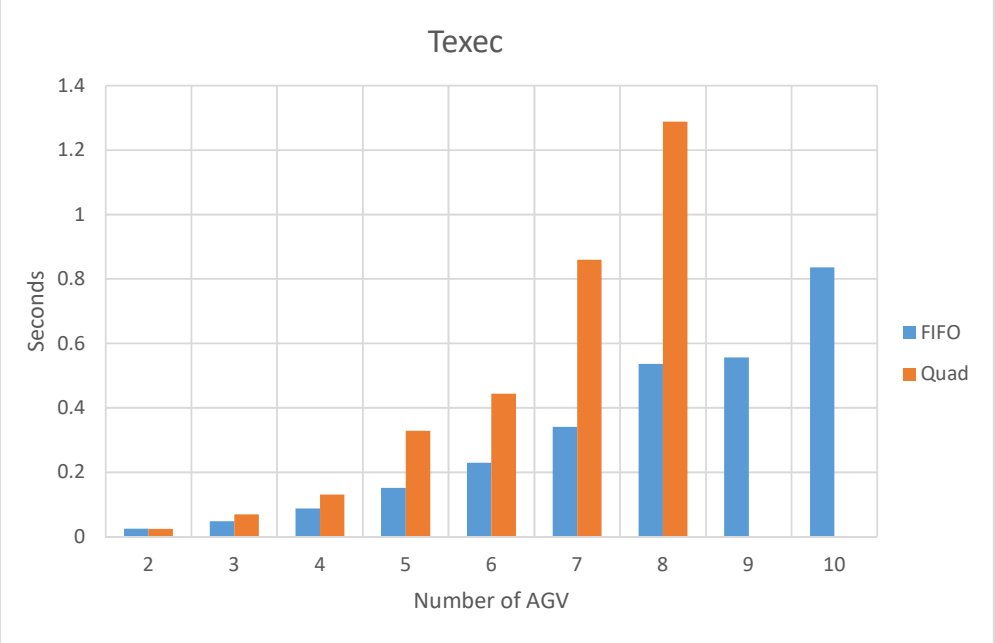

Figure 6: Computation time for Intersection Manager to calculate safe speeds for all participants on Intersection 4. SLSQP algorithm.

Figure 7: Average Clearing Time over 10 runs with random paths drawn from the six options.

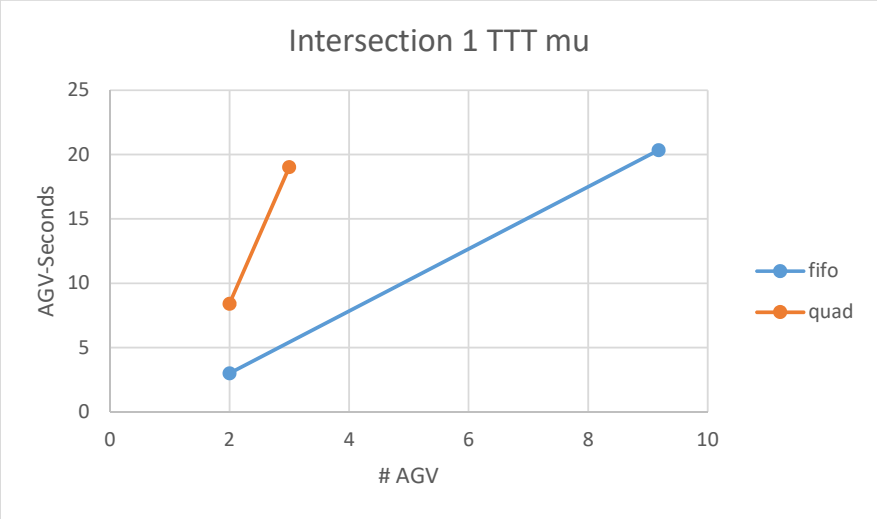

Figure 8: Total Simulated Travel Time in AGV-Time units

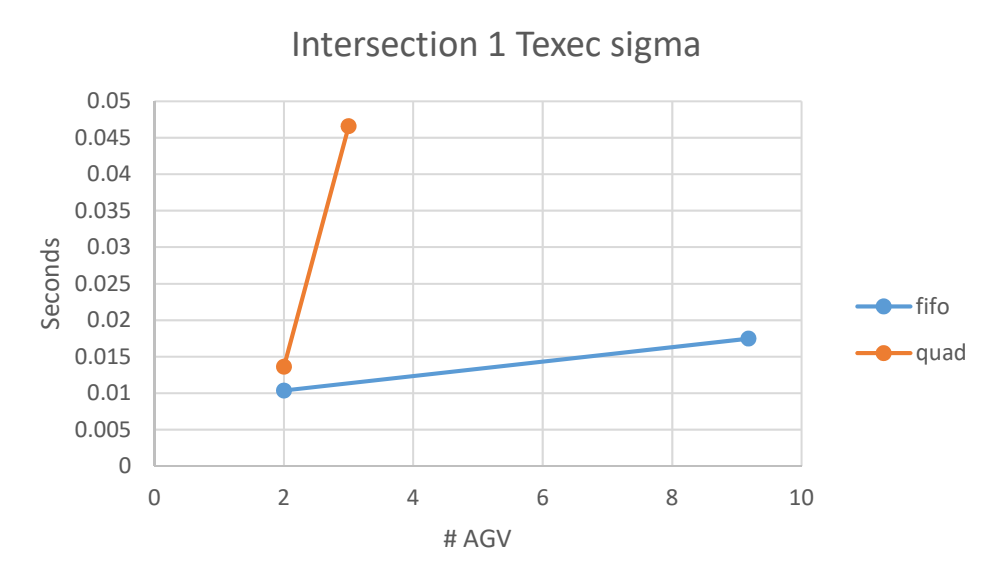

Figure 9: Standard Deviation of the Execution Time over 10 runs with random paths.

Results for two and three vehicles suggest *Quad_constr* is able to find a better solution at the cost of a higher execution time. It is difficult to establish a trend as this intersection only has three arrival nodes as shown in Figure 7.

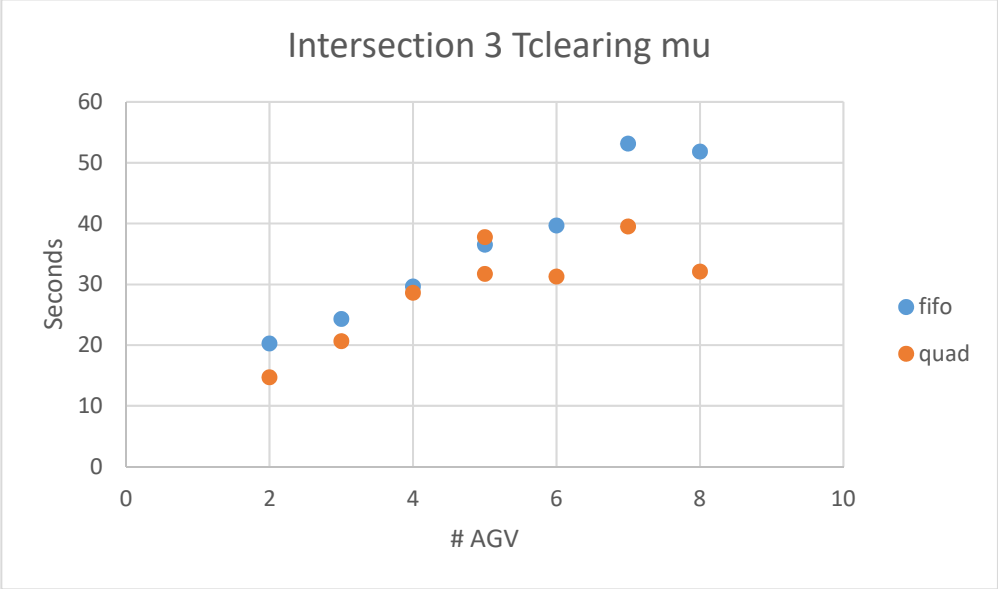

Figure 10: Simulation clock time for all AGV to clear Intersection 3.

The total travel time TTT is consistently improved by the $Quad_constr$ method, and the degree of improvement is greater the busier the intersection becomes. The clearing time shows a similar trend, but there is one test where the $FIFO$ solution leads to an improved $Tclearing$. As the objective function is based on Total Travel Time, the $Quad_contr$ solver improves the objective with a order which leads to a worse $Tclearing$.

Comparing the $Tworst\_wait$ should reveal the degree of individual sacrifice required to reach social optimum. As before, *Quad_constr* is consistently able to improve on the $FIFO$ solution by choosing a different order with the benefits increasing as the intersection gets busier. This shows that the wrong ordering leads to individual delay in higher traffic, even with a completely fair policy. Missed opportunities for non-conflicting flows to progress at the same time, lead to a longer wait for whoever crosses last.

6.1 Alternative Algorithm for Minimizing Linear Objective with Quadratic Constraints

In this test $SLSQP$ stands for the Sequential Least Squares Quadratic Programming algorithm. With the default settings this algorithm terminated much faster, but the solution were lower quality as shown by the consistently worse travel time than the *trust_constr* method with $gtol = 1e^{-6}$ in Figure 13. The value of $gtol$ is the threshold

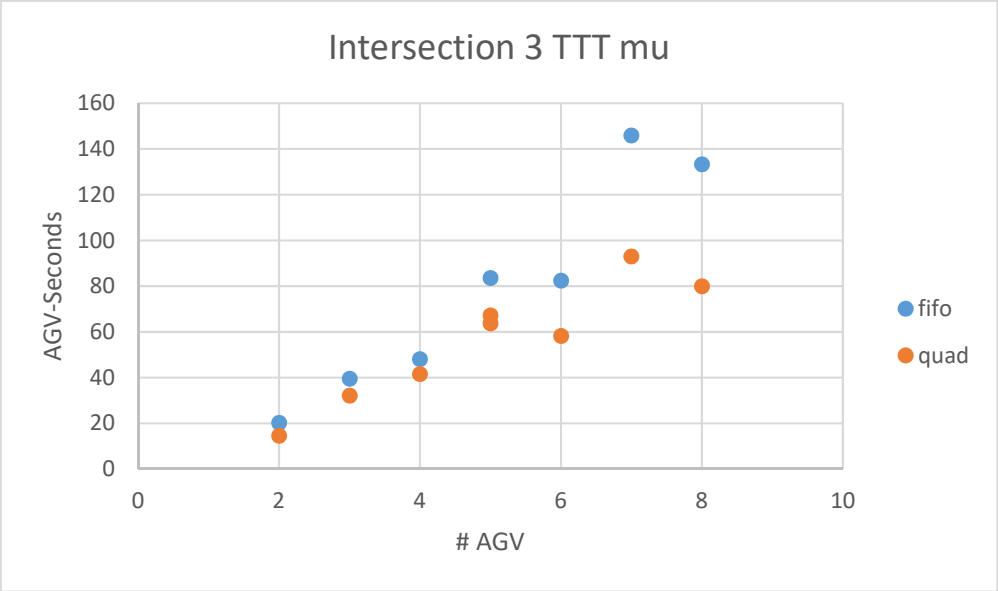

Figure 11: Intersection 3 Total Travel Time.

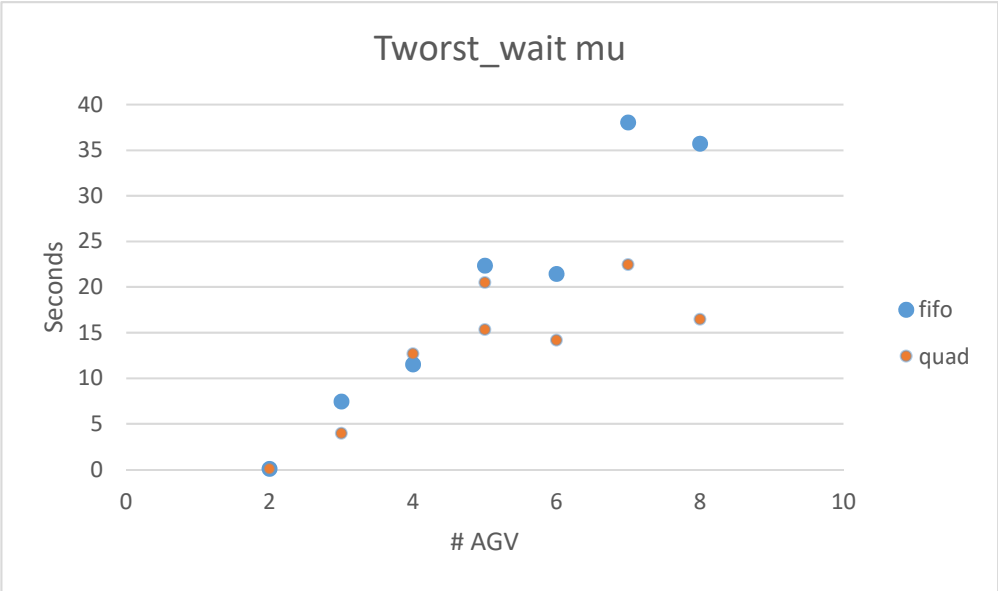

Figure 12: Worst waiting time experienced by a single AGV, averaged over 10 runs across Intersection 3.

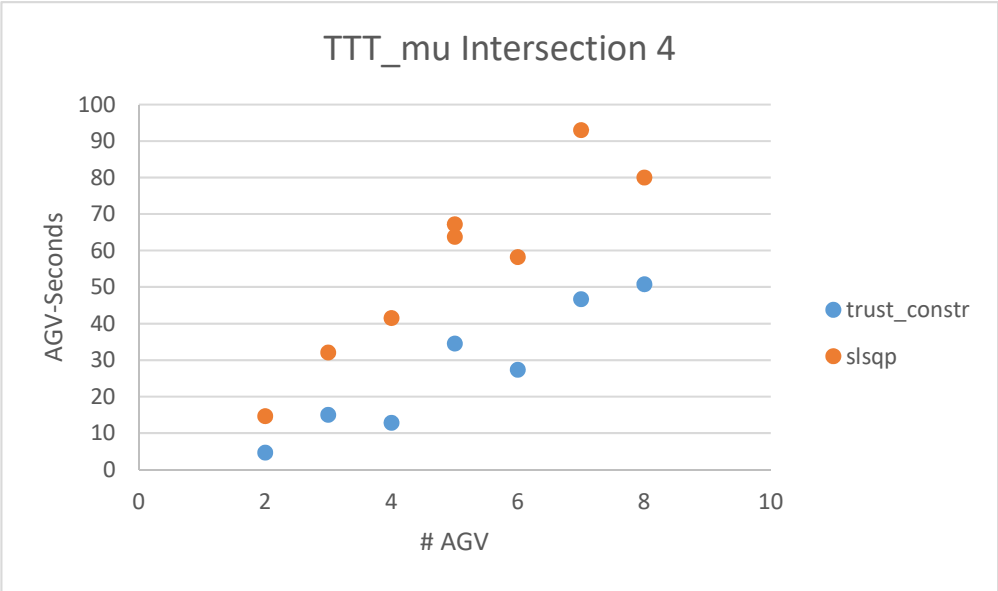

Figure 13: Solution quality with the two algorithms can be compared based on the TTT objective.

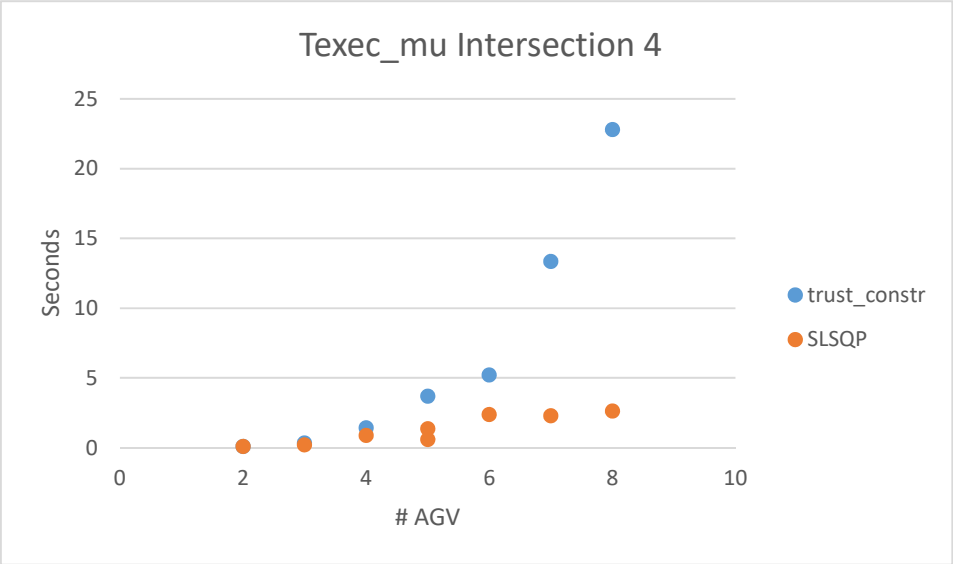

Figure 14: Mean execution time solving the Quadratic constraints with two algorithms.

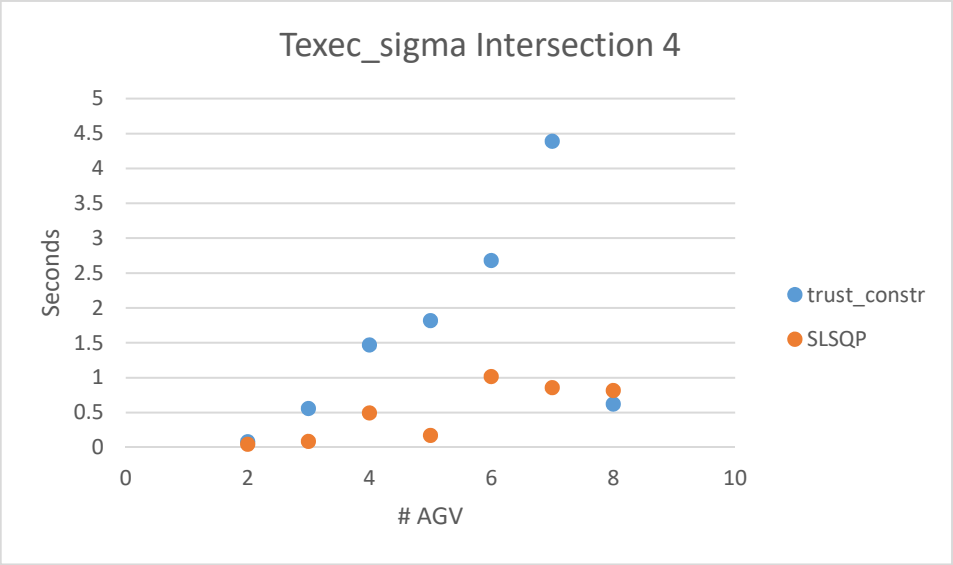

Figure 15: Standard Deviation of execution time for the Quadratic constraints.

the gradient of the objective must be below for termination, and has already been reduced from its default of $gtol = 1e^{-8}$ to try and speed things up.

There is a problem when n_agv=8, which leads to the number of iterations exceeding the limit, which was set to n_iter=600. This explains the very low standard deviation in Figure 15. The time taken to complete n_iter iterations is very consistent. Despite early termination he solution appears to meet the constraints.

The results including n_agv=9,10,11 show a different pattern. Suggesting *slsqp* fails to find a good solution here. The execution time levels off as the solutions become much worse, suggesting the search terminated early. The iteration limit was not reached.

It is surprising the execution time for FIFo method which involves solving a linear program is so high, approaching a second for only 11 AGV. Closer inspection reveals that 99% of this time is spent sorting the list before the linear program is executed which seems to indicate a sorting bug.

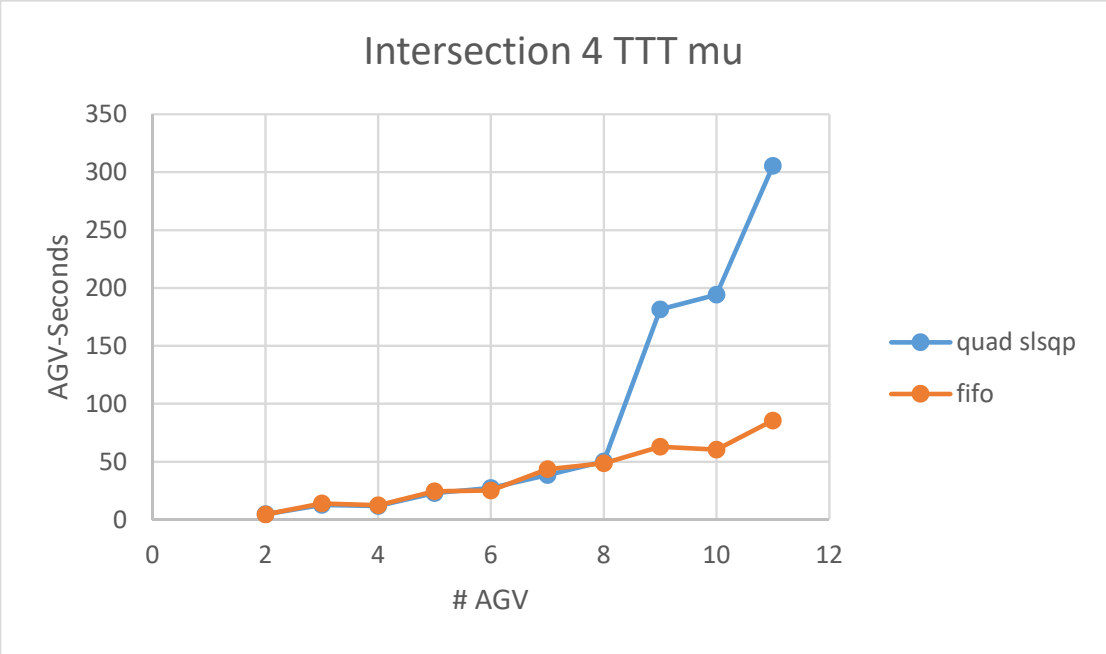

Figure 16: Total Travel Time for 2-11 Vehicles on Intersection 4.

References

[1] Valerio Digani, Lorenzo Sabattini, Cristian Secchi, and Cesare Fantuzzi. Ensemble Coordination Approach in Multi-AGV Systems Applied to Industrial Warehouses.

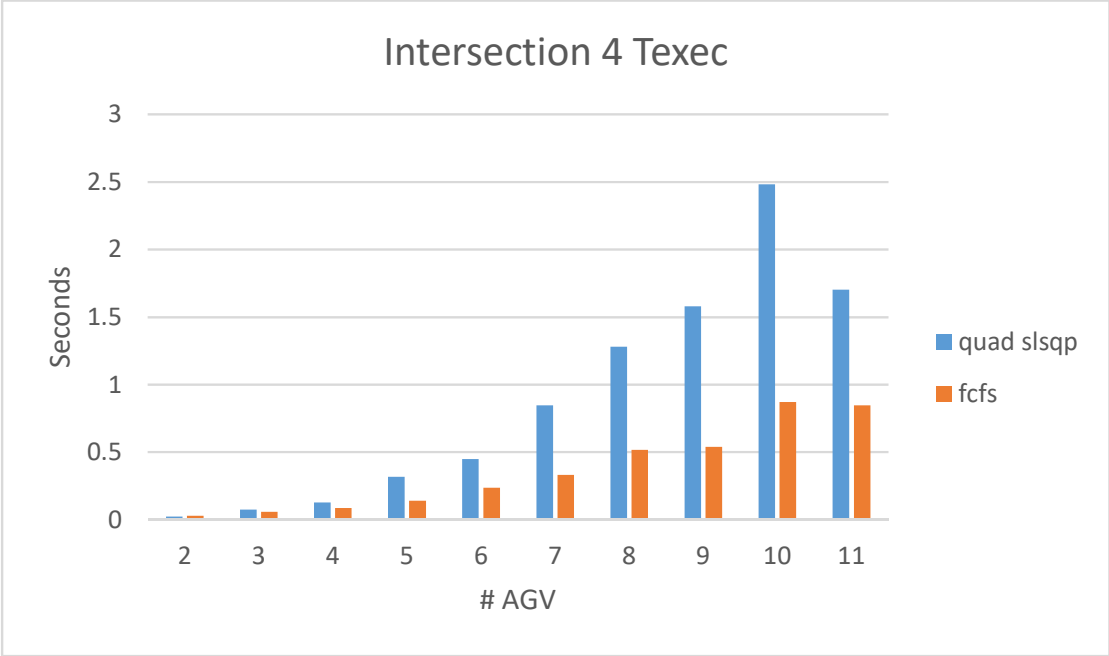

Figure 17: Execution time for 2-11 vehicles compared on Intersection 4.

IEEE Transactions on Automation Science and Engineering, 12(3):922–934, 2015.

[2] Valerio Digani, M. Ani Hsieh, Lorenzo Sabattini, and Cristian Secchi. Coordination of multiple AGVs: a quadratic optimization method. *Autonomous Robots*, 43(3):539–555, 2019.

[3] J A Reeds and L A Shepp. Optimal paths for a car that goes forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–393, 1990.