Multi-Lane Automated Intersection Management

Edward Lambert

December 11, 2021

1 Introduction

In previous chapters the throughput and delay of a number of promising Automated Intersection Management (AIM) algorithms have been compared on two perpendicular lane segments which meet in the middle. This might be called an elementary intersection. There are certain traffic phenomona which are difficult or impossible to observe in this environment, so it is difficult to draw firm conclusions about the superiority of one method. For instance, larger intersections will often permit non-conflicting flows such as opposing left turns (for those driving on the left-hand side).

The wider motion coordination problem faced by fleets of identical material transfer AGVs is described in Problem 1. A convenient representation of the workspace W is a connected graph of waypoint paths which avoid the static obstacles in the environment. This includes the starting position of every AGV $\tau_i(0)$, and the mission locations M_j.

Problem 1 Given a number N AGV operating in a workspace $W \in R^2$, what trajectory $\tau_i(t) \in W$ should each one follow to complete all missions in the minimum total AGV-Time. A mission takes place at position $M_j \in W$. A collision-free distance $||\tau_i(t) - \tau_k(t)|| > d_{min}$ must be maintained between each AGV at all times.

Numerous ways of dividing and conquering this problem have been developed. One of the most common is the conflict-free routing approach, where the path is discretized and segments are assigned to one vehicle at a time [1]. This is also known as prioritized planning and the priority order has a strong influence on the solution [2]. The global routing and crossing conflict resolution are solved simultaneously, using the same network representation for both.

An alternative where global routing is solved individually, and conflict resolution is performed at the intersection level is described in the breakdown in Chapter 3. In this scheme, Fleet Control assigns a mission j to an available AGV i and finds a suitable path through the network for only one AGV i using Dijkstra or similar graph shortest path technique. Now each AGV has a path π_i leading to its destination, which can be represented as a list of waypoints. An exact point in the Cartesian workspace any distance s along the path can be found by interpolation of this list $[\ x \quad y\ ] = \pi_i(s)$. The speed profile $s = v_i(t) \in R$ is not limited by the network representation. A set

of conflict avoiding speed profiles can be found by solving the local coordination sub-problem Problem 2.

Problem 2 Given a number $N < \bar{N}$ of AGV operating on a set of assigned paths $[\pi_0, ...\pi_N]$ which intersect between s_begin_i and s_end_i in path coordinates, at what speed profile $s_i(t)$ should each path be traversed so all paths are completed in minimum AGV-Time and the collision-free distance is maintained at the intersection points.

Digani et al [3] propose a 2-layer architecture to solve Problem 1 as a high level zone capacitated routing problem with local coordination within each zone to handle Sub-problem 2. A negotiation-based priority scheme had been previously used to complete the solution. However, the negotiation process could be time consuming so the system was improved by the addition of Autonomous Intersection Management (AIM) to avoid the need for negotiation in many cases. The original negotiation scheme remained active at all times as a backup. This led to higher average speeds, crucially reducing the average crossing time by more than the execution time of the optimization algorithm used to solve AIM. The backup system ensured correct behaviour in edge cases: preventing arrivals in the same lane until the intersection was clear. Having a backup gave sufficient certainty for a number of tests with hardware.

The collision avoidance constraints can be restated so they are convex, if the crossing order is fixed (FIFO order was tested). 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.

Preliminary work reported in Chapter 3**??** demonstrated the FIFO heuristic could produce equally good solutions with reduced execution time on an elementary intersection.

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 than $FIFO$. 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 intersect 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 because it is able to search over different orderings. Both approaches make the same simplifying assumptions about the constraints, maximize the same objective by varying the arrival time at 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 with interior point methods.

4 Method

4.1 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.

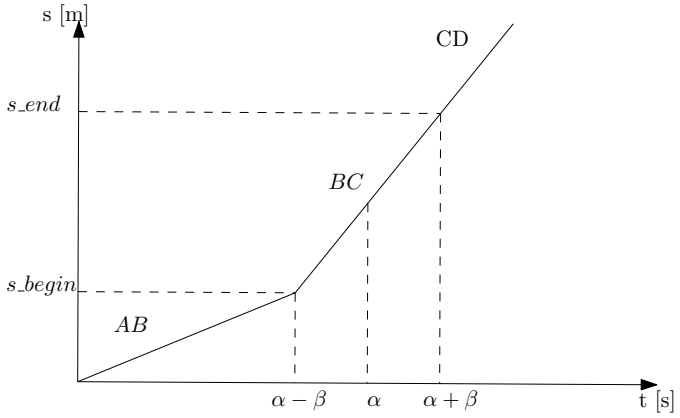


Figure 1: Diagram of one vehicle in path coordinates.

The path π_i can be evaluated at a longitudinal distance s to find the position $\xi_i(s) = \pi_i(s)$. Using the variable s it is helpful to divide the path into three parts as shown in Figure 1: approach, crossing and departing. The first part AB approaching the conflict where $s < s\_begin$, a part traversing the conflict zone BC where $s\_begin < s < s\_end$, and a part leaving the intersection CD where $s > s\_end$.

A1 No external obstacles can be found on the roadmap

A2 Followers on path π_i only begin to move after the lead AGV cleared the intersection

A3 Each AGV has a unique initial and final position

A4 The full set of paths through the intersection is known at initialization time

Assumption A1 ensures that collisions to be avoided by the motion co-ordination system are those between different AGVs. This is reasonable as most static obstacles can be avoided by construction of the roadmap. External obstacles comprise those which have unexpectedly moved since mapping time, and truly dynamic obstacles like pedestrians and human operated vehicles. For more on unexpected obstacles please refer to Chapter 2 **??**.

Similarly A3 rules out unavoidable collisions (or equivalently impossible to complete missions) caused by sending two vehicles to the same place at the same time. In the decomposition described in Chapter 3 **??** this function would be the responsibility of the task scheduling module. The AGV fleet scheduling problem is not a contribution of this thesis, other works in this area include the auction based method in [4].

Assumption A2 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 A4 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. If AGV are to plan adaptive paths across the intersection, they only need to limit their variation to within the original conflict zone. This is left as further work.

4.2 Problem Representation

In general there are a number of differential constraints on the motion of an AGV along a path. For any path and vehicle combination a maximum forward speed can be selected which keeps the lateral and angular acceleration within the vehicle constraints at the point of peak curvature and the point of peak sharpness. For simplicity of exposition in this work we take the lower for the two speed limits as the maximum $\bar{v}$ over the entire length so the lateral dynamics can be neglected without loss of generality. Continuous speed profiles which ensure path following subject to vehicle limitations as presented in [5] could offer higher speeds using the same waypoints in some cases. Methods for generating paths which are sufficiently smooth for a target traversal speed are discussed in Chapter 2 **??**. Longitudinal dynamics to second order lead to a maximum speed $\bar{v}$ and a maximum acceleration $\bar{a}$.

Additional assumptions B1, B2, B3 allow the optimal speed profile to be specified by the time of arrival at two waypoints. These are located at the beginning s_begin and end s_end of the conflict zone.

B1 Waypoint timing instructions are only sent with sufficient approach distance remaining to adjust speed before reaching the first waypoint $s\_begin - s > min\_conflict\_dist$

B2 The conflict zone must be long enough to reach the second waypoint at the right time without violating acceleration limits $s\_end - s\_begin > min\_conflict\_dist$

B3 All waypoints can be reached with an average speed $v < \bar{v}$

The maximum speed $\bar{v}$ is taken from the spec sheet. The $min\_conflict\_dist$ parameter is calculated based on the maximum acceleration parameter $\bar{a}$ of the AGV model. The acceleration limit is assumed to be symmetrical, so peak deceleration is $-\bar{a}$. The distance required to decelerate at $\bar{a}$ from any valid speed $v < \bar{v}$ is given by Equation 1.

$$min\_conflict\_dist = \frac{v^2}{2\hat{a}} \tag{1}$$

4.3 Individual Longitudinal Control

The individual controllers are simplified a great deal by using a first order dynamic model. This means that the controller can set the speed directly. The only source of error arises from the communication latency between the intersection manager and the vehicle, taken to be 200ms round trip. This is equal to two control cycles on the vehicle. The individual controller uses the latest position s and time t to meet the next Timed Waypoint $[\hat{s}, \hat{t}]$. The waypoint to target is chosen base on current position, the closest one with both $\hat{s} > s$ and $\hat{t} > t$ is selected. Then the target speed is calculated according to Equation 2.

$$u = \frac{\hat{s} - s}{\hat{t} - t} \tag{2}$$

This will ensure arrival at the right time, whatever the speed was over the communication delay. The actual speed of the simulated vehicle will be constant (assuming no safety sensor activation) but slightly different to that calculated by the intersection manager.

If the individual controller has not received a Timed waypoint yet, or it has passed the last one it will set the speed to the maximum allowed for this path.

4.4 Collision Avoidance

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 3 holds.

$$a_i > d_j \cup a_j > d_i \tag{3}$$

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 4. This contains a stack of pairs of reciprocal average speeds. One section AB and one across section BC.

$$\boldsymbol{\phi} = \begin{bmatrix} 1/v_{AB,0} \\ 1/v_{BC,0} \\ \vdots \\ 1/v_{AB,0} \\ 1/v_{BC,0} \end{bmatrix} \tag{4}$$

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 5.

$$\begin{bmatrix} s_0 - s\_begin_0 & 0 & \ddots & \mathbf{0} \\ \vdots & & \ddots & \vdots \\ \mathbf{0} & \ddots & s_N - s\_begin_N & 0 \end{bmatrix} \boldsymbol{\phi} \leq \boldsymbol{t_p} \tag{5}$$

4.5 FIFO

FIFO is simple heuristic for ordering based on the one described by [6]. In a world where all the vehicles have the same mass and acceleration and two arrive at full speed on different approach lanes, the farther vehicle will have to slow down less to allow the closer one to pass. To use a different ordering in this simple case always wastes time, and our objective is to minimize travel time. For this method the pairs in $\boldsymbol{\phi}$ must be sorted by the remaining distance along path AB 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 3 can be expressed as

$$\begin{bmatrix} s_0 - s\_begin_0 & s_0 - s\_end_0 & \ddots & \mathbf{0} \\ \vdots & & \ddots & \vdots \\ \mathbf{0} & \ddots & s_N - s\_begin_N & s_N - s\_end_N \end{bmatrix} \boldsymbol{\phi} \leq \begin{bmatrix} a_0 \\ d_0 \\ \vdots \\ a_N \\ d_N \end{bmatrix} \tag{6}$$

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.

4.6 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 7 for the constraint between any pair of AGVs (i, j).

$$|\alpha_i - \alpha_j| > \beta_i + \beta_j \tag{7}$$

And equivalently Equation 8.

$$(\alpha_i - \alpha_j)^2 - (\beta_i + \beta_j)^2 > 0 \tag{8}$$

In terms of the parameter vector $\boldsymbol{\phi_i}$ containing the only the reciprocal speeds relevant to AGV i, this can be captured in a block diagonal Hessian H_{ij}, containing a block Λ_{ij} for each pair as shown in Equation 9.

$$\Lambda_{ij} = \begin{bmatrix} A_i \mathbf{1}_i \mathbf{1}_i^T A_i & -A_i \mathbf{1}_i \mathbf{1}_j^T A_j \\ A_j \mathbf{1}_j \mathbf{1}_i^T A_i & A_j \mathbf{1}_j \mathbf{1}_j^T A_j \end{bmatrix} \tag{9}$$

$$\boldsymbol{H}_{ij} = \begin{bmatrix} \boldsymbol{\Lambda}_{ii} & \boldsymbol{\Lambda}_{ij} \\ \boldsymbol{\Lambda}_{ji} & \boldsymbol{\Lambda}_{jj} \end{bmatrix} \tag{10}$$

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

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. Timed waypoints are expressed as a $[s, t] \in \mathbb{R}^2$ tuple. The discrete waypoint list which specifies the path is used to convert the path distance s into a workspace position $[\ x \quad y\ ]$.

The timed waypoint 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. As there are two constant speed sections, two timed waypoints are calculated. Section AB is represented by $[s\_begin, \phi_{AB}(s\_begin - s)]$ and section BC by $[s_end, \phi_{AB}(s\_begin - s) + \phi_{BC}(s\_end - s\_begin)]$. As long as vehicles reach point s at time t, their actual speed profile can take any form as long as it remains positive. This was tested in Chapter 3 **??** where a second order model of the motor dynamics followed the instructions. In this test the simulated AGVs follow a simple first order model, the controller used is specified in Section **??**.

The first test is inspired by the published results in [7], 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. As both the tested methods in the present study are optimal in that sense, it makes sense to use the delay based definition given in Equation 11 to compare them. The delay based Twait will averaged over different AGV in each run, and the worst run average is recorded for that layout.

$$Twait = crossing\_time - \frac{PathLength}{max\_speed} \tag{11}$$

In the table below $Twait$ is calculated according to Equation 11 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.

6 Numerical Results

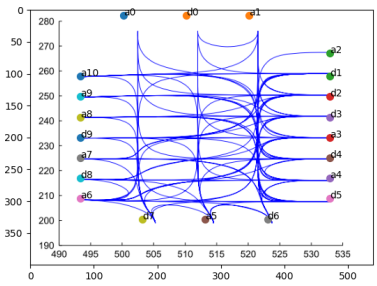


Figure 2: Intersection 4 reproduced from [7] with approximate arrival a and departure d nodes overlaid.

To examine the success of the FIFO heuristic on a complex intersection we begin with with the results for Intersection 4 with 11 entrances. The waypoints were reproduced by estimating the origin and destination points and finding new Reeds-Shepp paths between them. The effect can be seen by comparing Figure 2 and Figure 3 where the conflict locations are similar in size and number but there are some diagonal straight sections.

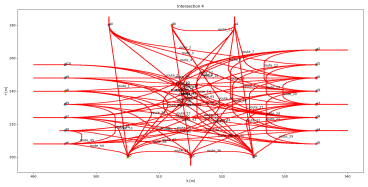


Figure 3: Intersection 4 detailed waypoints using Reeds-Shepp paths [8] to join the 66 $a-d$ pairs.

In figure 4 the total travel time produced by $FIFO$ ordering is similar to the solution to the quadratic constraints for 2 - 5 AGV. For 6 - 11 AGV the quadratic constraints permit much lower travel times. The greater the number of AGVs the greater the advantage, with travel times for 11 vehicles reduced from over 350 AGV-Seconds to less than 150 AGV-Seconds. This corresponds to a doubling of average crossing speed.

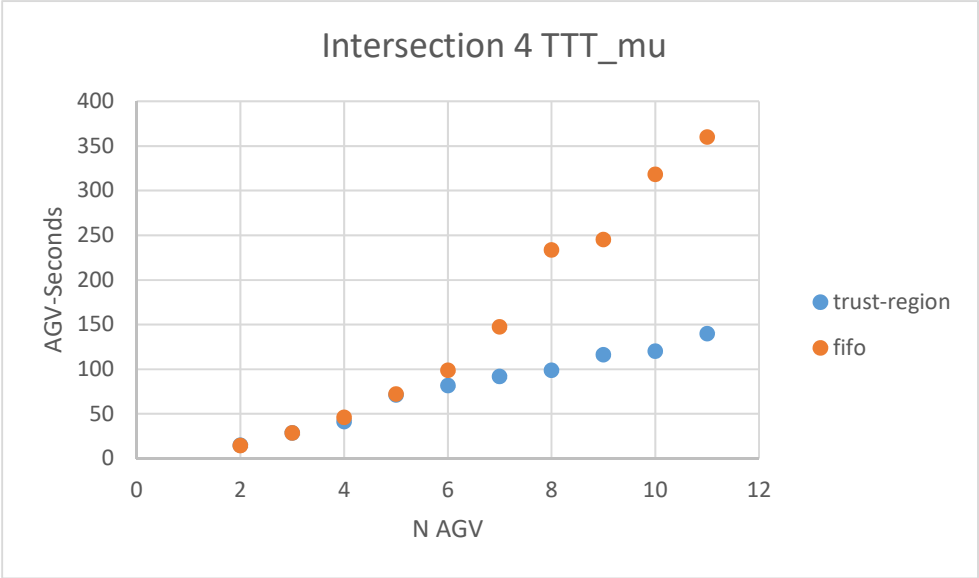


Figure 4: Total Travel Time for 2-11 Vehicles on Intersection 4. Trust Region Algorithm.

Execution time results in Figure 5 are for the $trust-region$ algorithm, running on a 1.6Ghz Intel(R) Core(TM) i5-8250U Quad Core with 16 GB main memory. The FIFO method is much faster to compute as expected. The computation time increases roughly

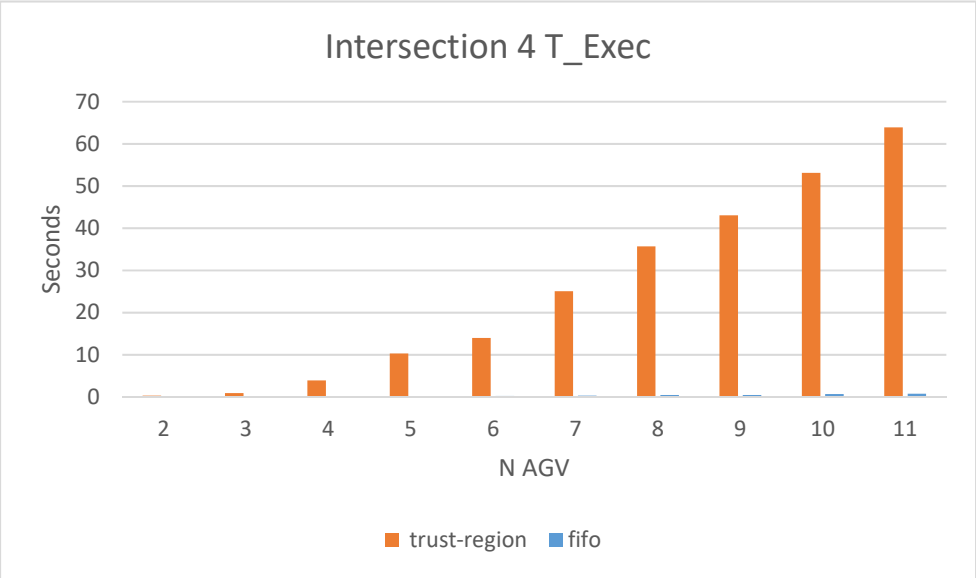


Figure 5: Execution time for 2-11 vehicles compared on Intersection 4. Trust region Algorithm.

linearly with the number of vehicles, reaching 65 seconds for 11 AGV. This may be too slow for a responsive real time controller, but the search can be terminated earlier and the constraints will be satisfied, so no risk of collision only slightly reduced crossing speed.

6.1 Intersection 3

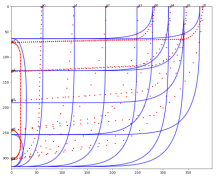


Figure 6: Intersection 3 Layout including eight arrival nodes. Reproduced waypoints as tested shown as red dots.

To establish how performance varies with intersection geometry, Intersection 3 in Figure 6 has up to 8 entrances, and Intersection 4 shown in Figure 2 has up to 12. Lower numbers of vehicles can be tested on large intersection by repeated runs with the vehicles starting at random locations.

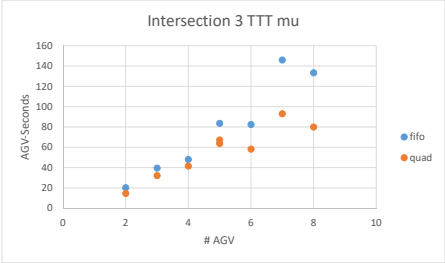


Figure 7: Intersection 3 Total Travel Time. Quadratic constraints solved with SLSQP algorithm.

The total travel time TTT in Figure 7 is consistently improved by the *Quad_constr*

method, and the degree of improvement is greater the busier the intersection becomes.

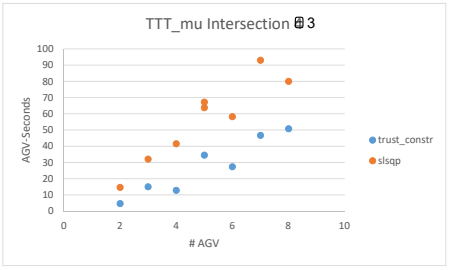


Figure 8: Solution quality with the two algorithms can be compared based on the TTT objective.

The travel time is improved further by switching to the trust region algorithm, as shown in Figure 8. On the same layout, with the same arrivals the trust region algorithm takes much longer to converge but find a superior ordering which almost halves the travel time objective.

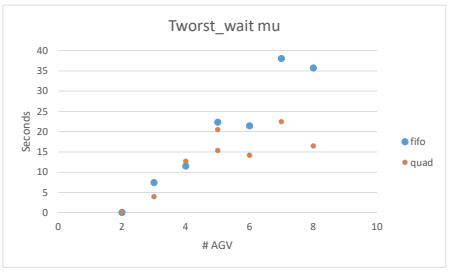


Figure 9: Worst waiting time experienced by a single AGV, averaged over 10 runs across Intersection 3.

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.

6.2 Intersection 1 Results in Detail

Intersection 1 only has three arrival nodes as shown in Figure 10, so only two and three arrivals can be tested while keeping one vehicle per lane and no car following (Assumption A3).

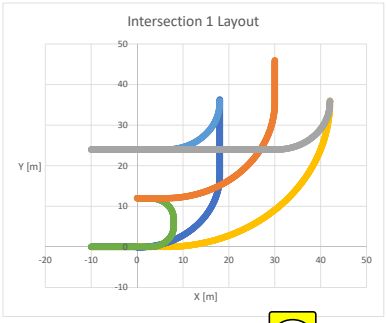


Figure 10: Waypoints with 10cm spacing for each the six paths through Intersection 1.

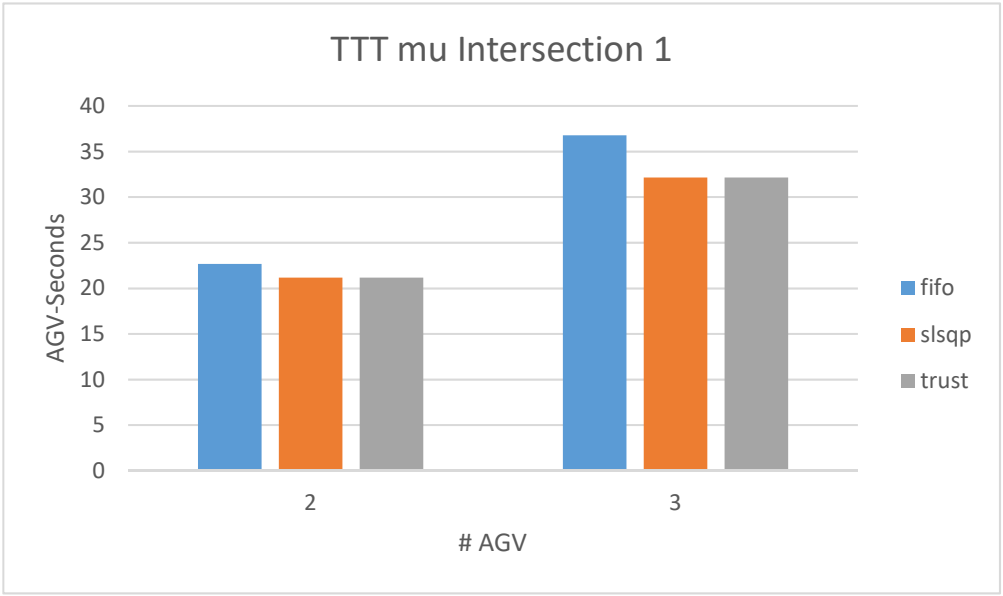


Figure 11: Total Simulated Travel Time in AGV-Time units

Results for two and three vehicles in Figure 11 suggest *Quad_constr* is able to find a better solution at the cost of a higher execution time. With few vehicles interacting the difference is minimal, suggesting only one or two of the 10 runs could be improved with a change in ordering and most were the same. The execution time variability shown in

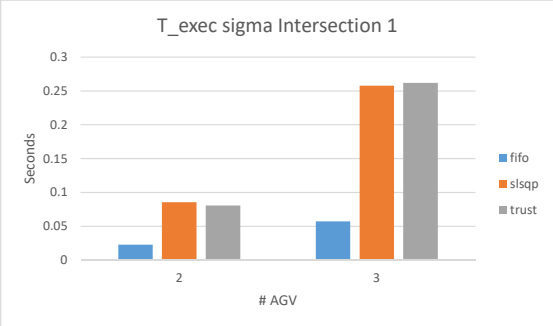


Figure 12: Standard Deviation of the Execution Time over 10 runs with random paths.

Figure 12 increases markedly with the switch to quadratic constraints. The choice of algorithm makes little difference in this case.

6.2.1 Detail of Improved Ordering with 2 AGV

To understand better the mechanism for improving travel time by reordering, it is instructive to examine two vehicles in Intersection 1, where a reordering benefit was detectable, although it became more significant for higher numbers of crossings on larger intersections.

The distance-time trace for each vehicle can be plotted together, where distance is measured as chainage along the path, starting at the arrival point. This means the conflict start distance is different for each AGV. The location is marked by a circle plotted at the start and the star plotted at the end in Figure 13. The AGV position at each time step is represented by a cross one colour for each AGV. The timing instruction from the intersection manager is indicated by the position of the circle and start on the

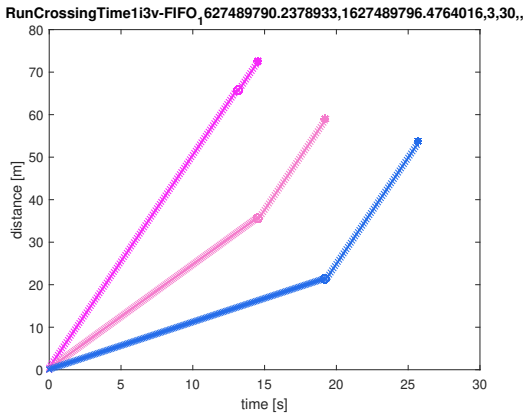


Figure 13: Distance along path over time for each AGV on Intersection 1, Run 3 with FIFO method.

time axis. The use of latency compensated first order control with no disturbances ensures the deadlines are m et exactly in this simulation.

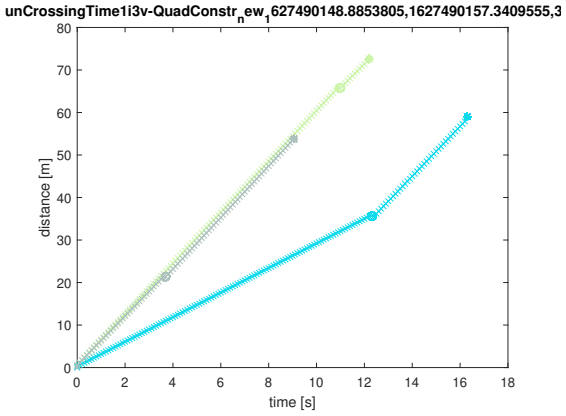
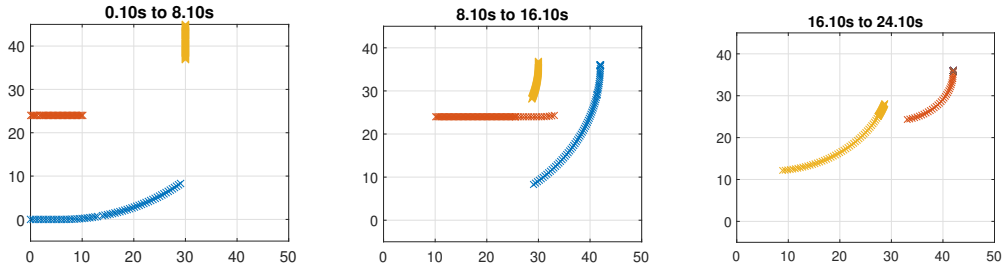


Figure 14: Distance along path over time for each AGV on Intersection 1, Run 3 with Quad (trust-constr) method.

The distance time plot of the same scenario with three vehicles under quad control in Figure 14 looks very different. The fact that two routes are not in direct conflict is exploited to allow two vehicles to proceed at full speed. The third vehicle needs to slow down a little less as less time is used for the second one to get clear.

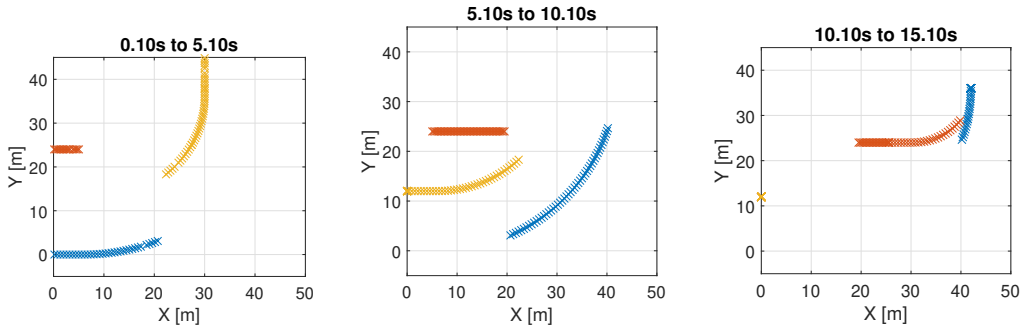
Shown on a plan view in Figure 15 the red and yellow traces are seen to travel bery slowly to give the blue trace (closest to the decision point) time to get clear.

By contrast, the traces in Figure 16 both blue and yellow proceed at full speed as they don not conflict. Once they are out of the way the red trace can proceed, saving 10 seconds overall.



(a) (b) (c)

Figure 15: Positions of three AGV over the specified time window under FIFO ordering.



(a) (b) (c)

Figure 16: Positions of three AGV over the specified time window under Quad management with flexible ordering.

6.3 Alternative Algorithm for Minimizing Linear Objective with Quadratic Constraints

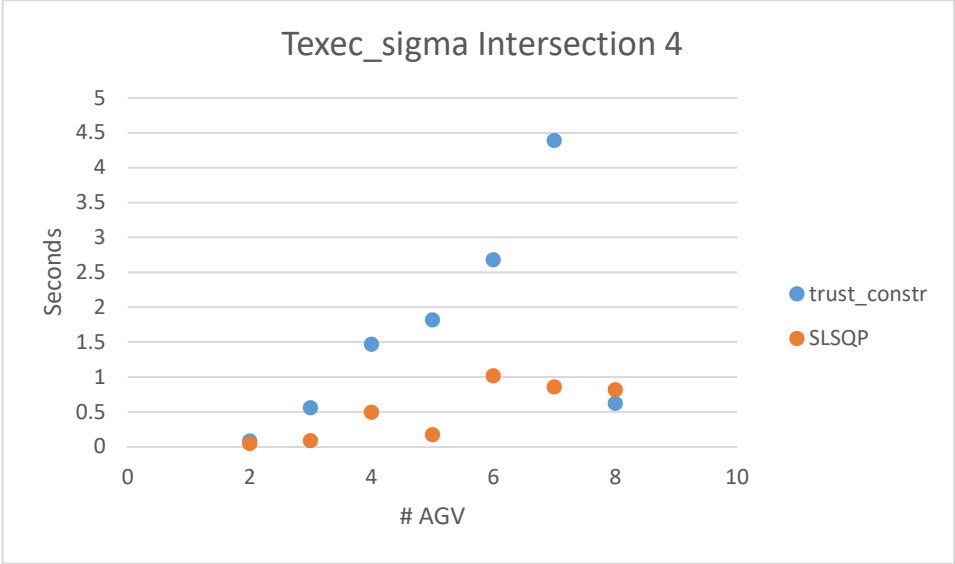


Figure 17: Standard Deviation of execution time for the Quadratic constraints.

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 **??**. The value of *gtol* is the threshold 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 17. 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.

7 Conclusion

On all three intersection layouts tested, the quadratic constraints formulation consistently found a crossing solution with higher average speeds. The non-convex constraints likely contributed to the long execution time, depending on the minimization algorithm

and termination criteria. These can be tuned depending on the available computing power and real-time execution deadlines. Crucially, with either trust region or slsqp method a solution which satisfies the hard constraints will be available, and more search time finds a solution closer to the global minimum.

References

[1] Jur P. Van Den Berg and Mark H. Overmars. Prioritized motion planning for multiple robots. *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pages 2217–2222, 2005.

[2] Maren Bennewitz, Wolfram Burgard, and Sebastian Thrun. Constraint-based optimization of priority schemes for decoupled path planning techniques. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2174(July):78–93, 2001.

[3] Valerio Digani, M. Ani Hsieh, Lorenzo Sabattini, and Cristian Secchi. A Quadratic Programming approach for coordinating multi-AGV systems. *IEEE International Conference on Automation Science and Engineering*, 2015-Octob:600–605, 2015.

[4] F. Basile, P. Chiacchio, and E. Di Marino. An auction-based approach to control automated warehouses using smart vehicles. *Control Engineering Practice*, 90(July):285–300, 2019.

[5] Pedro F. Lima, Marco Trincavelli, Jonas Martensson, and Bo Wahlberg. Clothoid-Based Speed Profiler and Control for Autonomous Driving. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2015-Octob:2194–2199, 2015.

[6] Zhiyuan Du, Baisravan HomChaudhuri, and Pierluigi Pisu. Hierarchical distributed coordination strategy of connected and automated vehicles at multiple intersections. *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, 22(2):144–158, 2018.

[7] 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.

[8] 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.