Comparison of IP and CNF Models for Control of Automated Valet Parking Systems

Abdullah Makkeh and Dirk Oliver Theis*

Abstract In automated valet parking system, a central computer controls a number of robots which have the capability to move in two directions, under cars, lift a car up, carry it to another parking slot, and drop it.

We study the theoretical throughput limitations of these systems: Given a car park layout, an initial configuration of a car park (location of cars, robots), into a desired, terminal configuration, what is the optimal set of control instructions for the robots to reorganize the initial into the terminal configuration. We propose a discretization and compare an Integer Programming model and a CNF-model on real-world and random test data.

Keywords: Discrete optimization and control, emerging applications, logistics.

1 Introduction

In May 2015, the Miami Herald [7] printed an article about a fancy condominium tower in Miami. That building was equipped with a state-of-the-art automated valet parking system. In these systems, a central computer controls a small army of robots moving in the parking area. The robots have the capability to move in two directions, move under cars, lift a car up, carry it to another parking slot, and drop it. A human driver drops off his car at a *vehicle transfer station,* where the car is picked up by a robot. The robots store away the car in the parking lot until the driver requests to retrieve

Abdullah Makkeh, Dirk Oliver Theis
Institute of Computer Science of the University of Tartu. Ülikooli 17, 51014 Tartu, Estonia. e-mail: `abdullah.makkeh@ut.ee`

* Supported by the Estonian Research Council, ETAG (*Eesti Teadusagentuur*), through PUT Exploratory Grant #620, and by the European Regional Development Fund through the Estonian Center of Excellence in Computer Science, EXCS.

it. Systems like these are installed in parking lots of apartment complexes, airports, and malls in many countries.

According to the Miami Herald, in that apartment complex Miami, the following was happening. In the mornings, when the inhabitants of the apartments wanted to drive to work, the system was overloaded: Its algorithms were not able to steer the robots to satisfy a large number of requests within a small time window. Car owners experienced long waiting times (20–30 minutes) between the time when they requested their cars, and when they were delivered to the vehicle transfer stations. According to the Miami Herald, the car owners took Ubers to work, instead of waiting for the sluggish machines to drag out their cars. The operator promised improvement of the algorithms [7], but ultimately had to shut down the project [6], went into bankruptcy [8], and was sued by the residents [4].

This example illustrates nothing else than the need for optimization in the robot operated car parking systems: according to the New York Times, "the technology is there" [12].

In this paper, we study a fundamental optimization problem involved in the control of an automated car parking system. We study the theoretical throughput limitations of these systems: Given a car park layout, an initial configuration of a car park (location of cars, robots), into a desired, terminal configuration, what is the optimal set of control instructions for the robots to reorganize the initial configuration into the terminal configuration. By "optimal", we mean fastest, in terms of clock-on-the-wall waiting time until the robots have completed their tasks.

With the ultimate goal of heuristic algorithms for the control of the robots, in this paper, we study exact algorithms. We propose an Integer Programming (IP) model and a Constraint Programming model (Boolean variables with constraints in conjunctive normal form, CNF), and compare the two approaches by testing them on parking lot scenarios.

In an attempt to stay very close to the application, both approaches model very closely a realistic model of the way a particular type of robots operates, with specific velocities of loaded and empty robots along with acceleration times, speeds of car lift and drop, etc. The goal of the engineers who designed these robots was to turn existing, human-operated parking lots into robot operated ones, rather than building the whole parking lot from scratch, with the idea to increase the packing density of cars on parking lots in densely populated cities. In particular, in this paper we will consider the case study of parking lots in big condos in Tallinn, Estonia.

The rest of this paper is organized as follows. In the next section, we review literature relevant for our problem. In Section 3, we give an overview of the IP model and the CNF clauses. In Section 4 we describe the data, the computational experiments we ran, and we discuss the obtained results. In Section 5, we draw some conclusions regarding the problem.

We defer to the supplement [5] for full details of our models, in case the referees would like to verify their soundness.

2 Basic approach and related work

2.1 Problem data and basic modelling approach

The technical details of the problem were the result of a collaboration with a company which considered entering the market of automated valet parking installations. In the considered parking lots, all slots have the same width (3m) and length (6m), and all slots are parallel: the width is in East-West direction, the length in North-South direction. The parking lots have a rectangular bounding box: e.g., if the bounding box is 300m in wide and 600m long, the parking lot can contain up to 10×10=100 slots. Correspondingly, slots are identified by x (East-to-West) and y (South-to-North) coordinates. We allow for slots to be unusable, due to either obstacles (walls, pillars, broken down robots) or simply parked cars which, for some reason, we currently don't want to move. The robots can move either in North-South or in East-West direction (but not, e.g., diagonally); also they don't need to "rotate". A robot must come to a complete stand still before changing directions. The robots' maximum speed is 3m/s when empty, or 1.5m/s when carrying a car; they require 1s to accelerate to maximum speed or decelerate from maximum speed to standing still. (We don't allow for a robot to stand still between two slots.) The robot needs 6s to lift a car and 2s to drop it.

These data suggest to discretize time into intervals of .25s. E.g., Fig. 1 shows a complete sequence of a North-movement by a robot carrying a car from a slot (x, y) to a slot $(x, y+2)$ and then accelerate West. In time interval t_0, the robot picks up speed, and moves 1/8 of the length of a slot. In time intervals $t_0 + 1, \dots, t_0 + 7$, the robot moves at full speed. In time interval $t_0 + 8$, it slows down and comes to a stand 2 places north of where it started. The control instruction "ready" causes the robot to initiate stopping. In time interval $t_0 + 9$, the robot could be executing a new control instruction; in the figure, it is accelerating West.

The discrete optimal control problem we aim to solve is now the following: Given two configurations of locations of robots and cars on the parking lot, an "initial configuration" and a "terminal configuration", determine a feasible set of control instructions for the robots which transforms the initial configuration into the terminal configuration; minimize the time the reconfiguration takes. Clearly equivalent to a set of control instructions is a sequence of configurations, each one of which can be derived from the previous one by a feasible move of the robot.

In a practical application, there is usually no need to fix initial and terminal locations of each individual car. It is, e.g., not relevant whether car #47 goes to vehicle transfer station #23 and car number #54 goes to vehicle transfer station #22 — or vice versa. Hence, we work with "colored" cars, and the initial and terminal configurations specify only whether a slot is occupied by a car and if so, what the "color" of that car should be. In our computer

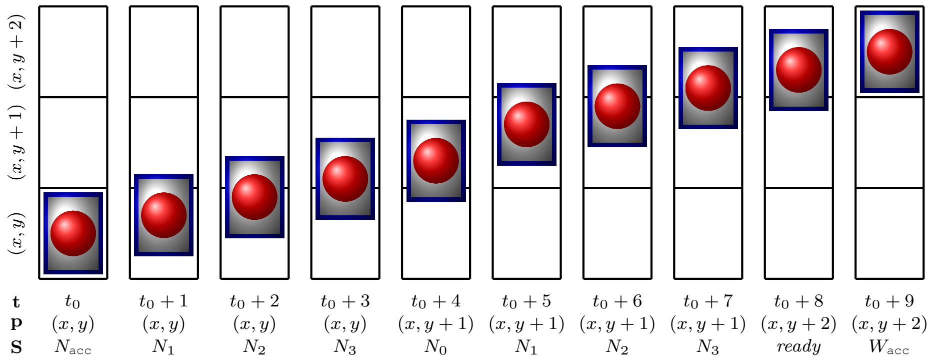

(x, y)
$(x, y+1)$
$(x, y+2)$
t t_0 t_0+1 t_0+2 t_0+3 t_0+4 t_0+5 t_0+6 t_0+7 t_0+8 t_0+9
p (x,y) (x,y) (x,y) (x,y) $(x,y+1)$ $(x,y+1)$ $(x,y+1)$ $(x,y+1)$ $(x,y+2)$ $(x,y+2)$
S N_{acc} N_1 N_2 N_3 N_0 N_1 N_2 N_3 *ready* W_{acc}

Fig. 1 *North* movement of a loaded robot. **t** = time interval; **p** = position associated and **S** = stage of the move. Robots is drawn in the pysical location which it occupies at the beginning of the time interval.

code, we formulate and solve IP and CNF models whose feasible solutions are sequences of configurations, for 3 colors.

2.2 Simplifications and literature review

Simplified versions of the problem have been studied theoretically. Most importantly, disregarding the role of robots (i.e., assuming that the cars move by themselves) and the speeds gives variants of pebble motion and reconfiguration problems in grids, or, more generally graphs: Vertices represent parking slots, and edges represent slots sharing an edge. We now review what is known.

The most famous problem in this group is the *15-Puzzle:* on a 4×4 grid, 15 vertices are occupied by labeled pebbles. ("Labeled" means that each pebble has a color of its own.) The decision version of the problem is whether a given initial configuration can be transformed into a given terminal configuration through a sequence of moving pebbles to adjacent vertices; the optimization version asks for a reconfiguration with the smallest number of total moves. The decision problem can be solved in polynomial time [14]. The optimization problem on grids is NP-hard [11], but constant factor approximation algorithms exist [10].

For graphs in general, Papadimitriou et al. [9] proved that with two labels, one pebble has a unique label, the problem is NP-hard. They also gave a polynomial time algorithm, using flow techniques, for the optimal solution on trees. The result was later improved to $O(n^5)$ by Auletta et al. [1]. Pach et al. [2] showed that the optimization problem on graphs is APX-hard. Moreover, when allowing more pebbles to move at the same time, J. Yu and S. laValle [15] proved that the optimization problem on graphs is NP-hard. It follows from [2] and [15] that there is no polynomial time (approximation) scheme for the simplified version of our problem, unless P = NP.

3 IP and CNF Models

Let $C := \{0, 1, \ldots, |C| - 1\}$ denote the set of colors of cars. The models require an upper bound on the number of time intervals: Each time interval is identified by an element of $T = \{0, 1, \ldots, t_{\max}\}$. The configuration at $t = 0$ is the initial configuration, the configuration at $t = t_{\max}$ is the terminal configuration. A car is called *stationary* if it is not carried by a robot.

The Variables.

For every time interval and every slot, four types of variables determine what is happening there and then: *slot occupation status* (whether there's a stationary car, and if so, of what color), *slot robot-status, SRS* (whether there's a robot, and if so, what it is carrying and doing), *robot vertical process* (if there is a robot lifting or dropping a car, which phase of that process is it executing), *robot horizontal movement* (if there is a moving robot, which phase of the movement is it executing). All variables are Boolean (0/1). At most one of the variables in each group is 1. Here they are.

Slot Occupation Status $x_{v,t,c}$, for every slot $v = (x, y)$, every time interval $t = 0, \ldots, t_{\max}$, and every $c \in \{\phi\} \cup \{C\}$. The symbol ϕ stands for "no car". Example: $x_{v,t,\phi} = 1$ iff during time interval t, there is no stationary car at the slot with coordinates v; $x_{v,t,2} = 1$ iff during time interval t, there is a stationary car of color 1 at the slot with coordinates v.

Slot Robot-Status (SRS) $s_{v,t,c,d}$, for every slot v, every time interval t, every $c \in \{\phi\} \cup \{C\}$, and every $d \in \{\epsilon, \rho, \mu, \zeta\}$. The symbol ϵ stands for "no robot", ρ stands for "ready", μ stands for "robot moving", ζ stands for "vertical process". Example: $s_{v,t,\phi,\rho} = 1$ iff at the slot with coordinates v, during time interval t, there is a robot doing executing the "ready" control instruction, i.e., it is doing one of the following: (1) nothing (being idle); (2) ending a movement (decelerating); (3) ending lifting or dropping a car. Another example: $s_{v,t,2,\zeta} = 1$, iff at the slot with coordinates v, during time interval t, there which is in the process of either lifting or dropping a car of type 2.

Robot Vertical Process (RVert) $z_{v,t,p}$, for every slot v, every time interval t, and every $p \in \{\lambda_0, \lambda_1, \lambda_2, \lambda_3, \lambda_4, \delta\}$. The symbols λ_i for $0 \le i \le 4$ stand for the phases of "lifting", δ stands for "dropping". Example: $z_{v,t,\lambda_2} = 1$ iff during time interval t, the robot is in the third phase of lifting at the slot with coordinates v. The following diagram shows the interconnection between x-,

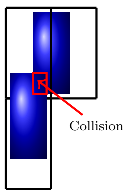
s-, and z-variables:

$t = t_0$	$x_{v,t,3} = 1$, $s_{v,t,\phi,\rho} = 1$, $z_{v,t,*} = 0$	a car of color 3, a robot
$t = t_0 + 1$	$x_{v,t,\phi} = 1$, $s_{v,t,3,\zeta} = 1$, $z_{v,t,\lambda_0} = 1$	no car; a robot lifting a car of color 3
$\vdots$	$\vdots$	$\vdots$
$t = t_0 + 5$	$x_{v,t,\phi} = 1$, $s_{v,t,3,\zeta} = 1$, $z_{v,t,\lambda_4} = 1$	no car; a robot lifting a car of color 3
$t = t_0 + 6$	$x_{v,t,\phi} = 1$, $s_{v,t,3,\rho} = 1$, $z_{v,t,*} = 0$	no car; robot executing "ready"

Robot Horizontal movements (RMove) $m_{v,t,c,e}$, for every slot v, every time interval t, every $c \in \{\phi\} \cup \{C\}$, and every $e \in \{N_{\text{acc}}\} \cup \{N_i | 0 \leq i \leq 3\} \cup \{S_{\text{acc}}\} \cup \{S_i | 0 \leq i \leq 3\} \cup \{E_a\} \cup \{E_i | 0 \leq i \leq 1\} \cup \{W_{\text{acc}}\} \cup \{W_i | 0 \leq i \leq 1\}$. The symbol N_{acc} stands for " acceleration in the north direction" and N_i for $0 \leq i \leq 3$ stands for the phases of "northern move". Similarly the other stand for the other directions. Example: $m_{v,t,2,E_1} = 1$ iff during time interval t, the robot moving E_1 at the slot with coordinates v towards its east. The interconnection between the x- and the m-variables is similar to the RVert case.

IP-Model Constraints

The setup of the variables allow to formulate both the IP- and the SAT-model by linking only consecutive time intervals $t, t+1$. Care has to be taken to allow for all feasible movements, while making sure that collisions (see the figure on the right) are ruled out. The following constraints describe a loaded robot on (x, y) moving *north*.



$$\sum_{i=0}^{|C|-1} m_{(x,y),t,i,N_0} \leq s_{(x,y+1),t,\phi,\epsilon} + \sum_{i=0}^{|C|-1} (m_{(x,y+1),t,i,N_0} + m_{(x,y+1),t,i,N_{\text{acc}}} + m_{(x,y+1),t,i,1} + m_{(x,y+1),t,i,N_2} + m_{(x,y+1),t,i,N_3}) + m_{(x,y+1),t,\phi,N_{\text{acc}}} + m_{(x,y+1),t,\phi,N_1} \tag{1}$$

$$\sum_{i=0}^{|C|-1} m_{(x,y),t,i,N_{\text{acc}}} \leq s_{(x,y+1),t,\phi,\epsilon} + \sum_{i=0}^{|C|-1} (m_{(x,y+1),t,i,N_{\text{acc}}} + m_{(x,y+1),t,i,N_1} + m_{(x,y+1),t,i,N_2} + m_{(x,y+1),t,i,N_3}) + m_{(x,y+1),t,\phi,N_{\text{acc}}} + m_{(x,y+1),t,\phi,N_1} \tag{2}$$

$$\sum_{i=0}^{|C|-1} m_{(x,y),t,i,N_1} \leq s_{(x,y+1),t,\phi,\epsilon} + \sum_{i=0}^{|C|-1} (m_{(x,y+1),t,i,N_1} + m_{(x,y+1),t,i,N_2} + m_{(x,y+1),t,i,N_3}) + m_{(x,y+1),t,\phi,N_1} \tag{3}$$

$$\sum_{i=0}^{|C|-1} m_{(x,y),t,i,N_2} \leq s_{(x,y+1),t,\phi,\epsilon} + \sum_{i=0}^{|C|-1} (m_{(x,y+1),t,i,N_2} + m_{(x,y+1),t,i,N_3}) \tag{4}$$

$$\sum_{i=0}^{|C|-1} m_{(x,y),t,i,N_3} \leq s_{(x,y+1),t,\phi,\epsilon} + \sum_{i=0}^{|C|-1} m_{(x,y+1),t,i,N_3} \tag{5}$$

$$\sum_{i=0}^{|C|-1} m_{(x,y),t,i,N_0} \leq s_{(x,y-1),t,\phi,\epsilon} + \sum_{i=0}^{|C|-1} m_{(x,y-1),t,i,N_0} \tag{6}$$

First (1) says that if a robot is moving in stage *N0* then *north* of it there can be nothing or robot accelerating in the same direction as it and so on. E.g., when the robot is moving *N1* from (3) there can't be a robot to the *north* of it accelerating since a collision will occur. Moreover, since *N0* is the stage of north movement when the robot arrives to (x, y) and is almost occupying all of this spot, then by (6) $(x, y-1)$ is unoccupied or there is a robot moving *N0*. We defer to the supplement [5] for the full details of the IP-model. Now we state the following result.

Proposition 1. *The IP-formulation using the x-, s-, z-, and m-variables on an* $n \times n$ *grid has* $O(t_{\max} n^2)$ *inequalities.*

The Objective Function. We made numerous experiments with different objective functions. The idea is to "nudge" the robots into movement, while trying to support reaching the terminal configuration as quickly as possible. For each time interval, $f(t) = \frac{t}{100 \cdot t_{max}}$ was the cost of the *SRS* excluding $s_{v,t,\phi,\rho}$ and $s_{v,t,\phi,\epsilon}$. The objective function is as follows,

$$\sum_{\forall v \in V, \forall t \in T, \forall i \in C} f(t) \cdot (s_{v,t,i,\rho} + s_{v,t,\phi,\mu} + s_{v,t,i,\mu} + s_{v,t,\phi,\zeta}) \tag{7}$$

The computational results in the next section use a linear dependence on the time for that. Details can be found in the supplement [5].

SAT-Model Clauses

The SAT-model follows the same basic approach as the IP-model. Indeed, many of the clauses have a direct counterpart in an inequality, and vice versa. Overall, deriving the CNF clauses from the control requirements and the variables is an exercise in logical thinking. The following clauses determine the situation of a slot v at time t after lifting or before dropping.

$$\forall i \in C \; s_{v,t+1,i,\rho} \vee_{\substack{j \in C \\ j \neq i}} s_{v,t+1,j,\rho} \vee x_{v,t,\phi} \vee z_{v,t,\lambda_4} \tag{8}$$

$$\forall i \in C \; s_{v,t,i,\rho} \vee_{\substack{j \in C \\ j \neq i}} s_{v,t,j,\rho} \vee x_{v,t,\phi} \vee z_{v,t+1,\delta} \tag{9}$$

$$\forall i, j \in C \; \neg s_{v,t+1,i,\rho} \vee \neg s_{v,t+1,j,\rho} \tag{10}$$

$$\forall i, j \in C \; \neg s_{v,t,i,\rho} \vee \neg s_{v,t,j,\rho} \tag{11}$$

By (8) and (10), in $t+1$, if a loaded robot (car of type i) will be ready on v, then, in t, v has no cars or the robot is done with lifting a car of type i. Similarly (9) and (11), if in t, the loaded robot is ready on v, then, in $t+1$, v will have no cars or the robot will drop a car of type i. We defer to the supplement [5] for the full details.

4 Computational Results

The data. We have two sets of parking lots. Five parking lots are derived from an existing parking area in Tallinn, Estonia. That parking area has been split up into parts, based on requirements such as having vehicle transfer stations accessible from elevators for pedestrians. (Humans and robots cannot use the same floor area for operational safety.) Then there is a set of 5 parking lots generated randomly: Starting from an $m \times n$ grid, we place "walls" between parking slots with a given probability.

Each parking lot gives rise to several instances to be solved, by choosing the initial and terminal configurations of the cars and robots. All configurations were chosen randomly (discarding those with no reconfiguration of initial to terminal configurations.) The total number of instances is 30. The largest instances were 13×10 (only 41 free spot). In those 7 cars of 3 different colors and 5 robots were used. Also we had small instances (4×5 and 9 free spots) with 6 cars of 3 different colors and 2 robots. Compared to real world problems, the instances are small in terms of number of cars, but the sizes and number of colors are compatible with real world ones. The full description of the all parking lots and instances can be found in the supplement [5].

Finally each instances is solved several times, for varying $t_{\max}$. (The difficulty is that no good upper bound on $t_{\max}$ is known.)

Hard- and Software We used the Gurobi optimizer [3] in version 7.0 for the IP-model, and CryptoMiniSat 5 [13] as SAT solver. The times were taken on a compute server with Intel(R) Core(TM) i7-4790K CPU (4 cores) and 16GB of RAM. (Both solvers were run with one thread.) Gurobi was run with the parameters `MIPFocus`=2, `Presolve`=2. CryptoMiniSat was run with `preproc`=1. Both solvers were given a time limit of 10000sec.

The time to read in the data from file and construct the models was negligible.

Results. CryptoMiniSat 5 very significantly outperformed Gurobi.

For only one of the resulting Integer Programs a solution was found within the time limit. For that one IP, in that case, the solve time was 90sec, whereas the solve time for the corresponding CNF model was 76sec.

The SAT-model, on the other hand, gave relatively satisfactory results. CryptoMinSat terminated within the given time limit for 85% the CNF-models (yielding either a feasible solution or the information that there was none). In 60% of instances, the running time was below 3600sec. In only 20% instances, the running time was larger than 6000sec.

The full computational results (tables) can be found in the supplement [5]. The code, instances and solutions are on GitHub (`Abzinger/crobots`).

5 Conclusion

Difficult discrete optimal control problems arise in the control of automated valet parking systems. For the study of exact methods for the problems, we devised a time-expanded model, and compared its solution via Integer Programming and CNF-based Constraint Programming. Using state of the art software for the two approaches, we found that Integer Programming was useless.

The CNF-model, however, proved to be usable. Now, the next goal is to (design and) compare the solution qualities (amount of time needed to reach a terminal configuration from a current configuration) of heuristic algorithms to optimal solutions, which can be found or at least bounded using our CNF-model.

References

1. Vincenzo Auletta, Angelo Monti, Mimmo Parente, and Pino Persiano. A linear-time algorithm for the feasibility of pebble motion on trees. *Algorithmica*, 23(3):223–245, 1999.
2. Gruia Călinescu, Adrian Dumitrescu, and János Pach. Reconfigurations in graphs and grids. *SIAM Journal on Discrete Mathematics*, 22(1):124–138, 2008.
3. Inc. Gurobi Optimization. Gurobi optimizer reference manual, 2016.
4. Debora Lima. Brickell condo residents sue developer over faulty robotic garage. Miami Herald, April 20 2016.
5. Abdullah Makkeh and Dirk Oliver Theis. Supplement to the paper "Comparison of IP and CNF models for control of automated valet parking systems". Supplementary material, University of Tartu, 2017. http://ac.cs.ut.ee/files/robot-supplement.pdf.
6. Nicholas Nehamas. Residents furious as robotic parking garage at Brickell condo shuts down. Miami Herald, November 6 2015.
7. Nicholas Nehamas. Robotic parking garage ruins commute, Brickell condo residents say. Miami Herald, May 15 2015.
8. Nicholas Nehamas. Operator pulls out of Brickell condos disastrous robotic parking garage. Miami Herald, January 11 2016.
9. Christos H Papadimitriou, Prabhakar Raghavan, Madhu Sudan, and Hisao Tamaki. Motion planning on a graph. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 511–520. IEEE, 1994.
10. Daniel Ratner and Manfred Warmuth. The (n2- 1)-puzzle and related relocation problems. *Journal of Symbolic Computation*, 10(2):111–137, 1990.
11. Daniel Ratner and Manfred K Warmuth. Finding a shortest solution for the n× n extension of the 15-puzzle is intractable. In *AAAI*, pages 168–172, 1986.
12. Frances Robles. Road to robotic parking is littered with faulty projects. The New York Times, November 27 2015.
13. Mate Soos. The cryptominisat 5 set of solvers at sat competition 2016. *SAT COMPETITION 2016*, page 28, 2016.
14. Richard M Wilson. Graph puzzles, homotopy, and the alternating group. *Journal of Combinatorial Theory, Series B*, 16(1):86 – 96, 1974.
15. Jingjin Yu and Steven M LaValle. Structure and intractability of optimal multi-robot path planning on graphs. In *AAAI*, 2013.