Second Annual Conference on Advances in Cognitive Systems Poster Collection (2013) 251-266

Planning with Qualitative Models for Robotic Domains

Timothy Wiley TIMOTHYW@CSE.UNSW.EDU.AU
Claude Sammut CLAUDE@CSE.UNSW.EDU.AU
School of Computer Science and Engineering, University of New South Wales,
Sydney, NSW 2052, Australia

Ivan Bratko BRATKO@FRI.UNI-LJ.SI
Faculty of Computer and Information Science, University of Ljubljana,
Trzaska 25, 1000 Ljubljana, Slovenia

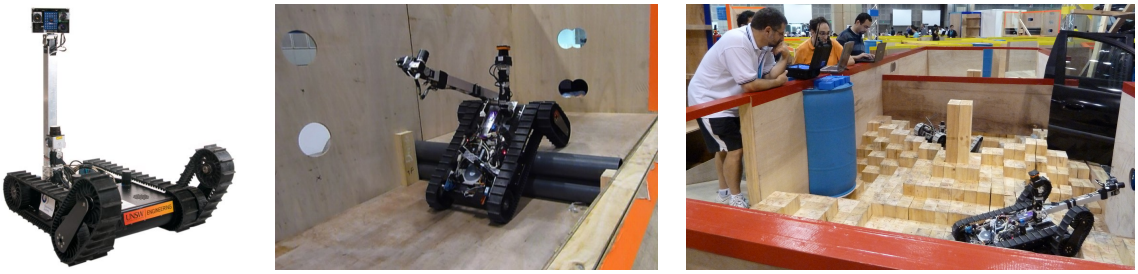
Abstract

We aim to develop an online method for learning robot behaviours that requires only a small number of trials, making it practical for use on a real robot. A hybrid method includes a symbolic planner, using qualitative reasoning, that constructs an approximate solution to a control problem. This approximate solution provides constraints for a numerical optimisation algorithm, which is used to refine the qualitative plan into an operational policy. The method is demonstrated on a multi-tracked robot intended for urban search and rescue.

1. Introduction

Typically, some form of reinforcement learning is used for online learning, that is, learning while an agent performs a given task. The system performs a succession of trials, which initially will fail frequently. As more experience is gained, the control policy is refined to improve its success rate. In its early formulations (Michie & Chambers, 1968; Watkins, 1989; Sutton & Barto, 1998), reinforcement learning worked well as long as the number of state variables and actions was small. Subsequently, many methods have been proposed to alleviate this problem. These include the use of function approximation, e.g. (Tesauro, 1995), relational reinforcement learning (Džeroski, De Raedt, & Driessens, 2001), hierarchical learning (Dietterich, 1998; Hengst, 2002) and hybrids of symbolic AI and reinforcement learning (Ryan, 2002). Model-based reinforcement learning includes an initial "system identification" step that create a characterisation of the system being controlled, so that a controller could then be derived. See, for example, Ng et al. (2006) and Buskey et al. (2002), where data collected from a human pilot flying a remotely controlled helicopter are used to learn a model of the helicopter. A reinforcement learning algorithm is then run against this model to produce a controller.

Building such models generally requires strong domain knowledge to be built in. Most trial-and-error learning systems are incapable of making use of general background knowledge, as humans do. For example, if we are learning to drive a car that has a manual gear shift, the instructor does not tell the student, "Here is the steering wheel. Here is the gear stick. Here are the pedals. Play



(a) Robotic Platform (b) Climbing a Ledge (c) Traversing Rubble

Figure 1: Track based rescue robot and simulated disaster environments

with them until you figure out how to drive". Rather, the instructor will be quite explicit about the sequence of actions to perform. To change gears, the instructor might tell the student that the clutch has to be depressed simultaneously as the accelerator is released, followed by the gear change, and so on. However, the instructor cannot impart the "feel" of the pedals and gear stick or control the student's muscles so that the hands and feet apply the right pressure and move at just the right speed. This can only be learned by trial-and-error. So despite the qualitative knowledge, the student will still make mistakes until the parameters of the control policy are tuned. However, with no prior knowledge, learning would take much longer since the student has no guidance about what actions to try, and in what order to try them. The qualitative constraints also give the learner what might be described as "common sense" in that it can reason about the actions it is performing. In the example above, the background knowledge was provided by a teacher but some or all of it could also be obtained from prior learning.

We describe a system ultimately intended to speed up trial-and-error learning by using qualitative background knowledge to constrain search. It builds on earlier work in hierarchical learning by Ryan (2002), in which a hybrid learning system incorporates a planner. The planner generates a sequence of actions whose implementation is not given but which can be learned by reinforcement learning. Ryan's method works well in discrete domains but does not scale well to continuous domains, whereas we are interested in domains with continuous states and actions. Following a similar approach Sammut and Yik (2010), developed a system that gave a bipedal robot the ability to learn to walk. Here, a planner produced a sequence of qualitative actions that contained parameter values not specified by the planner. Subsequent trial-and-error learning determined the values of those parameters so that the qualitative plan was made operational.

Another domain for learning locomotion applies to robots used in the field of Urban Search and Rescue. These robots must traverse rough terrain such as rubble and staircases (Figure 1). Such robots commonly have a main set of tracks to drive the robot and sub-tracks, or flippers, that can be used to lift the robot's body or configured to climb over obstacles. The robot must choose the best trajectory to avoid becoming stuck and, in the case of robots that can change their geometry, the robot must also decide how best to do that for the terrain ahead. In this paper, we focus on the problem of choosing flipper positions to enable the robot to climb over obstacles.

The main contribution of this work is in modifying the qualitative reasoning system, QSIM (Kuipers, 1986), to perform planning, rather than simulation. The actions in a qualitative plan are

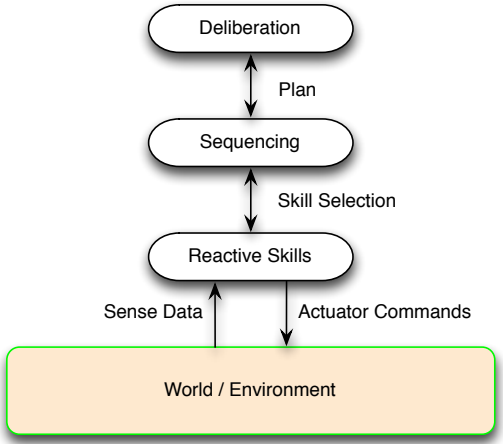


Figure 2: Modules in the ICARUS Architecture and their cascaded organization

not directly executable on the robot as they do not include any numerical parameters, such as the exact angle of the flipper. However, the advantage of using qualitative simulation for planning is that the process produces constraints on the range of possible of parameter settings. These parameters can be refined by reinforcement learning, now in a much smaller search space than if learning were applied in an unconstrained space.

2. A Planning and Learning Hierarchy

In a typical three level architecture for a robot (Bonasso et al., 1997; Gat, 1998), shown in Figure 2, the upper "deliberative" layer is responsible for long-term planning, generating actions to be performed. The intermediate, or "sequencing" layer selects the parameters required by the low-level controller to implement the actions. In previous work (Brown & Sammut, 2011), a STRIPS-like planner was used to generate actions. A constraint solver used information in the actions to limit the search of a motion planner, which provided the parameters for the controller. This scheme worked well because the robot in those experiments was wheeled and only need to drive over a flat surface. However, a more complex locomotion task, like driving over rough and unpredictable terrain in a disaster site, poses a challenge for a motion planner. In this case, we prefer to learn the robot's low-level behaviours. Like Sammut and Yik (2010), constraints derived from a plan are used to limit the search in trial-and-error learning to refine the parameters for the controller. In the case of both Brown and Sammut (2011) and Sammut and Yik (2010), a classical planner was used, followed by a constraint solver, in a separate phase. However, a qualitative simulation algorithm such as QSIM already incorporates a constraint solver. Additionally, combining the constraint solver in the planner allows for semi-numerical reasoning in the planning.

In its normal mode of operation, QSIM is given an initial state and performs a simulation to derive all possible states that may evolve from the initial conditions. QSIM has no concept of a goal state or actions that may be invoked to try to achieve the goal. To address this, Hogge (1987), Forbus (1989) and Aichernig et al. (2009) proposed planning architectures for qualitative process

theory. Like many qualitative reasoning frameworks, qualitative process theory uses a Qualitative Model to define the behaviour of a modelled system over time. The model consists of a set of qualitative constraints in the form of Qualitative Differential Equations (QDEs). The constraints defines relationships between variables in the system, and places restrictions on the value of variables. For planning, Hogge and Forbus used STRIPS-like actions, where each action added and removed constraints from the model. This in turn changes the behaviour of the system, allowing a planner to search through potential states and reach the desired goal. For example, in a water tank system where the tank has an out-flow pipe, in-flow pipe and a valves on the in-flow and out-flow pipes, an action might be to open or close one of the valves. This action adds and removes constraints that model the rate of change in the level of water in the tank.

Other frameworks for planning using qualitative reasoning include the reactive monitoring systems developed by DeJong (1994) and Drabble (1993). Qualitative reasoning is used to predict the next state of a system and adjust the quantitative controls of the system accordingly. However, as the name suggests, these were monitoring systems, not planners. These systems also used actions in the same form as Hogge and Forbus.

More recently, Troha and Bratko (2011) successfully performed planning using QSIM within the robotics domain. They trained a small two-wheeled robot to push objects into a given location. The robot first learnt the qualitative effects of pushing a given object, then planned a sequence of moves to place the object at a desired position with a desired orientation. However, their system was specialised to learning the effects of pushing objects.

A different approach to developing a framework for using qualitative reasoning in robotics was proposed by Mugan and Kuipers (2012) called QLAP (Qualitative Learner of Action and Perception). QLAP uses a different qualitative reasoning system to QSIM or qualitative process theory. Actions are defined as the occurrence of an event of the form $X \rightarrow x$, where variable X is set to the qualitative value x. For each event a small quantitative controller or Q-Table, $Q_i(s, a)$, is learnt. For each state s in the table action a is performed which corresponds to invoking another event $Y \rightarrow y$. In this way controllers are linked in a tree-like fashion. A task is completed by executing events that set all variables to their desired values, which causes a chain-reaction of actions to be triggered. In QLAP no planning is performed although it does allow for domain independent learning. Additionally, the tree representation means actions or events are highly linked such that if the configuration of one part of the robot is changed the entire tree of Q-tables must be re-learnt. We prefer to use a learning framework that allows for modularity as various aspects of the robotic system may change.

2.1 Framework for Learning System with Qualitative Planning

This work proposes a learning system (Figure 3) that is a modification of the design proposed by Sammut and Yik (2010). Whereas, the previous work used a STRIPS-like planner, this work proposes a qualitative planner where qualitative simulation is integrated with a heuristic state-action planning method. This includes a new formulation for qualitative actions that correspond to the physical actions of the robot. Actions in the resulting plan are parameterised and constrained. The constraints are used to restrict the search space of trial-and-error learner which finds the optimal values for the action's parameters.

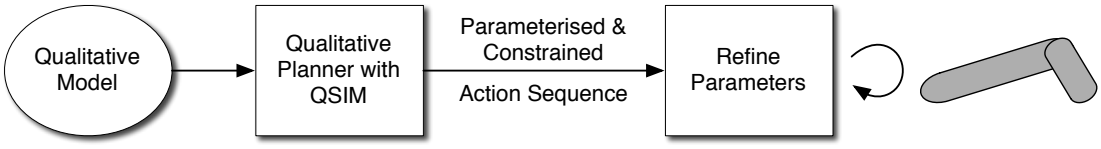


Figure 3: Architecture for learning robotic behaviours using a qualitative planner that incorporates the Qualitative Simulation algorithm, and a quantitative trial-and-error learner.

3. The Experimental Platform - Negotiator, a track-based robot

The robotic platform used in conducting experiments and shown in Figure 1a, is designed for Urban Search and Rescue. The robot consists of an iRobot Negotiator base that has been augmented with sensors and on-board computers. The vehicle contains two drive tracks and two sub-tracks, or "flippers". Each group of right and left tracks can be controlled independently, and the flippers can be rotated around a pivot point in parallel with one another. The vehicle is controlled using two actions that set the velocity of each track and the position of the flippers. Figure 4 shows the physical location various variables of the system and a basic task of climbing a ledge.

4. Qualitative Representation

The robot, and its interactions with the environment, are represented by *qualitative constraints* in the form of *qualitative differential equations (QDEs)*. These express relations between *qualitative variables*. The value of a variable is described by a qualitative magnitude relative to landmark values and a direction (either increasing, decreasing or steady) that describes how the magnitude changes over time. Table 1 lists a subset of the qualitative variables of the Negotiator system originally presented in Wiley et al. (2013).

The qualitative constraints place restrictions on the magnitude and direction of the qualitative variables. For example the relationship $\mathtt{M+}(x, y)$, enforces that as variable x increases, variable

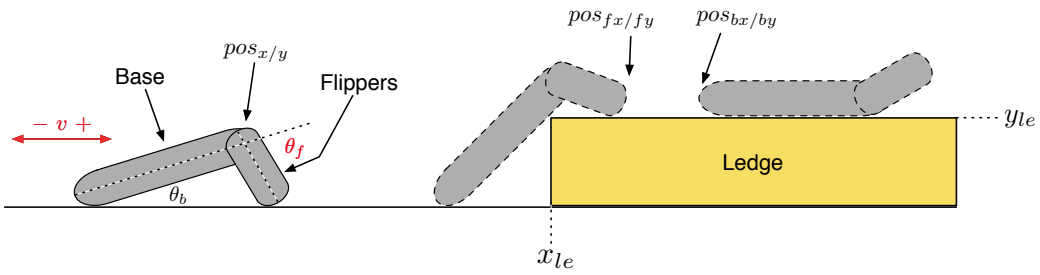


Figure 4: Representation of the Negotiator system and the task of climbing onto a ledge. The control variables are highlighted in red.

Table 1: Qualitative Variables in the Negotiator system. Landmarks such as θ_{fflat}, θ_{bflat} arise from properties of the robot and it's actions, while other landmarks such as max_{pxt} and x_{fall} arise from the robot interacting with the environment.

QVar	Landmarks	Category	Type	Description
pos_x	$[x_0, max_{pxt}, x_{le}, x_{fall}, x_{goal}]$	State		Robot x-position
pos_y	$[y_0, y_{le}]$	State		Robot y-position
pos_{fx}	$[x_0, x_{le}, x_{goal}]$	State		Flipper x-position
pos_{fy}	$[y_0, y_{le}]$	State		Flipper y-position
pos_{bx}	$[x_0, x_{le}, x_{goal}]$	State		Base x-position
pos_{by}	$[y_0, y_{le}]$	State		Base y-position
θ_f	$[-\pi, \theta_{crit}, 0, \theta_{fflat}, \pi]$	Control	Periodic	Flipper angle
θ_b	$[-\pi, \theta_{bflat}, \theta_{crit}, 0, \pi]$	State	Periodic	Base angle
v	$[v_{min}, 0, v_{max}]$	Control	Discrete	Velocity of the robot
θ_{fb}	$\left[-\pi, -\frac{\pi}{2}, 0, \frac{\pi}{2}, \pi\right]$	State	Periodic	Sum of θ_f and θ_b

y must also increase, and likewise as x decreases. The set of qualitative variables and qualitative constraints that describe a system form the *Qualitative Model* of the system. The model is influenced by both the physical properties of the robot and the operational environment of the robot.

The standard representation of qualitative models in QSIM is extended for the purposes of planning in the qualitative domain. Qualitative variables are extended by introducing *control variables*, in conjunction with standard QSIM qualitative variables, herein termed *state variables*. Control variables, when set, affect changes in the state of the system and may transition discontinuously. State variables are dependent and may not freely change value, but must follow the transition rules defined in the qualitative simulation algorithm. However, rather than being constrained by exterior action, state variables are dependent on control variables. For instance, the variables pos_{bx} and θ_b are state variables, while θ_f and v are control variables as these variables represent actions performed on the physical robot. *Discrete* and *periodic* qualitative variables are also introduced. The magnitude of a discrete qualitative variable may only reside at a landmark and the rate of change may only be steady. Discrete variables are further categorised as *nominal*, which may transition to any landmark in their domain, and *continuous* which may only transition between adjacent landmarks. Periodic variables, such as angles, allow additional transitions between landmarks at the extremities of their domains. For example, θ_f may transition between the landmarks $-\pi$ and π.

Qualitative constraints are extended by introducing *Preconditions*. These conditional constraints are termed *qualitative rules* and have the form:

$$Name : Precondition \rightarrow Constraints$$

where the constraints are a set QDE's. Qualitative rules allow constraints to be applied in a subset of the robot's state space, rather than applying globally as in typical applications of QSIM. Consequently, the qualitative simulation algorithm is modified such that constraints are only enforced in a state when the preconditions are satisfied. For example, the angle of the robot's body with respect

to the ground, θ_b depends on the angle of the flippers, θ_f. In some regions of the state space, both angles increase together, while in other regions θ_b decreases as θ_f increases.

Qualitative rules may, optionally, be grouped together by a common set of preconditions to form *operating regions*. For example, when the Negotiator robot is on flat ground, the robot's behaviour and the qualitative constraints representing this behaviour differ substantially compared to the behaviour of the robot as it is traversing the leading edge of the ledge. This defines two distinct operating regions, one for flat ground and one for traversing the ledge.

4.1 Qualitative Model for Negotiator climbing a ledge

The qualitative model for Negotiator and the task of climbing the ledge was originally presented in Wiley et al. (2013). Part of the model is presented again in Table 2, with some modifications. The model contains two operating regions: "flat" for where the robot is on flat ground, and "ledge" for where the robot is traversing the leading edge of the ledge[1].

The behaviour of the robot in the model is largely determined by the x/y-position of the robot (pos_x and pos_y), the x/y-position of the flippers (pos_{fx} and pos_{fy}), and the landmark values max_{pxt}, x_{le}, x_{fall} and y_{le}. As pos_x passes through landmark max_{pxt} the flipper must be in contact with the leading edge of the ledge, causing the robot to begin driving onto the ledge. As pos_x passes through x_{le} and x_{fall}, the base of the robot then comes into contact with the leading edge of the ledge. After x_{fall}, the centre of mass of the robot causes the robot to fall onto the top of the ledge. In addition to this, the angle of the flipper and the angle of the base may not exceed a critical threshold, θ_{crit}, as the robot does not have sufficient power to drive over the ledge if the angle of the robot is too steep. Finally, some states are impossible, indicated by $\perp$, the unsatisfiable constraint. Any transitions into states where $\perp$ applies are immediately discarded.

5. Planning Actions using Qualitative Simulation

Given a seed qualitative state q_0 occurring at time point t_0 a *single iteration* of the qualitative simulation algorithm (QSIM) produces a set of qualitative states $\{q_{0,1}\}$ that are immediately reachable from the seeding state. All the successor states occur during the *time interval* $t_{0,1}$. By repeated application using *multiple iterations* of QSIM, a set of *state sequences* $[(q_{0,1}, t_{0,1}), (q_1, t_1), (q_{1,2}, t_{1,2}), \ldots]$ is produced, that describe the set of all possible qualitative states that may be derived (eventually reached) from the seeding state and the qualitative trajectory taken to reach the state. State in a sequence (which also do not include the seed) are ordered by time and either occur at a time point or during a time interval. Time points represent instances in time while time intervals are non-zero lengths of time whose exact duration is unknown. Time points and intervals alternate such that a time point is followed by an interval and vice-versa. Finally, the length of each state sequence is variable, and some sequences may be subsets of each other.

For planning in the qualitative domain, a *Qualitative Action* is defined as executing one or more iterations of QSIM to produce a sequence state, where the value of all control variables in the

1. The common preconditions for each operating region are not listed in Table 2. The common preconditions for the "flat" operating region are $(pos_{fx} \leq x_{le} \wedge pos_x \leq x_{le}) \vee (pos_x \geq x_{fall})$. The common preconditions for the "ledge" operating region are $(pos_x \leq x_{fall}) \vee (pos_x \geq x_{le} \wedge pos_{fx} \geq x_{le})$.

Table 2: A subset of the qualitative model for Negotiator climbing a ledge. *Corresponding values* for various QDE's are provided by the `corr(..)` notation.

Name	Region	Preconditions	Constraints
`drive`			`deriv` (pos_x, v)
`pos_fx`			`sum` $(pos_x, \cos(\theta_{fb}), pos_{fx})$
`pos_fy`			`sum` $(pos_y, \sin(\theta_{fb}), pos_{fy})$
`sum_fb`			`sum` $(\theta_f, \theta_b, \theta_{fb})$
`flat_1`	flat	$-\pi \leq \theta_{fb} \leq 0$	`const` $(\theta_b, 0)$
`flat_2`	flat	$0 \leq \theta_{fb} \leq \frac{\pi}{2}$	`M-` (θ_b, θ_{fb}) , `corr` $(0, 0)$
			`M+` (θ_f, θ_{fb}) , `corr` $(0, 0)$
`flat_3`	flat	$\frac{\pi}{2} \leq \theta_{fb} \leq \pi$	`M+` (θ_b, θ_{fb}) , `corr` $(0, \pi)$
			`M+` (θ_f, θ_{fb}) , `corr` (π, π)
`in_ledge_1`	ledge	$pos_x > x_{le}, pos_y < y_{le}$	$\perp$
`in_ledge_2`	ledge	$pos_{fx} > x_{le}, pos_{fy} < y_{le}$	$\perp$
`in_ledge_3`	ledge	$pos_{bx} > x_{le}, pos_{by} < y_{le}$	$\perp$
`le_mpxt`	ledge		`M-` (pos_x, θ_{fb})
			`M-` (pos_x, θ_b) , `corr` $(max_{pxt}, 0)$
			`M+` (pos_x, pos_y) , `corr` $(max_{pxt}, 0)$,
`le_xle`	ledge	$x_{le} \leq pos_x \leq x_{fall}$	`M-` (pos_x, θ_b)
			`M-` (pos_y, θ_b)

sequence are constrained to fixed values. A Qualitative Action, a_i, is annotated as:

$$a_i := \{CQVar_1 = \texttt{Dom/Mag}, CQVar_2 = \texttt{Dom/Mag}, \ldots\}$$

where each $CQVar_i$ denotes the value that the control variable is constrained to for the entirety of the action. The finite set of all actions is thus the cross-product of all combinations of valid qualitative values for each control variable. It is important to note that a qualitative action does not include a specific state sequence produced by QSIM. Instead, performing an action may result in multiple successor states. At most, one successor state is produced for every state sequence produced by QSIM.

Planning with qualitative actions is performed using a classical state-action based approach. However, care must be taken since each action produces a set of state sequences of variable lengths, rather than a set of singular states. Also a distinction must be made between states used at the level of the planner, denoted by s_i, and states within the state sequences at the level of QSIM denoted by q_i (for time points) or $q_{i,i+1}$ (for time intervals). However, both the states at the level of the planner and at the level of QSIM are qualitative descriptions of the robot.

A plan is constructed from an initial state s_{init} to a goal s_{goal}[2]. Given a state in the planner s_i, the successor states $\{s_{i+1}\}$ are produced by the following algorithm:

2. In practice (see Section 7) the goal state is typically under specified. For example, the goal state may only specify the x/y-position of the robot. All other variables are unspecified. Thus, variables such as the angle of the flipper and thus the x/y-position of the end of the flipper may take any valid value in the goal. Therefore, multiple states may match the specified goal.

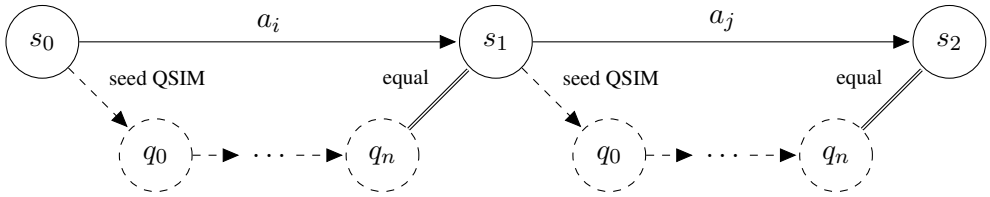


Figure 5: Representation of the qualitative planning process, from the planner state s_0 through s_1 to the planner state s_2. State s_0 and action a_0 produce state s_1 and likewise for $(s_1, a_1) \rightarrow s_2$. The QSIM state sequences $q_0, \ldots, q_n$ are the result of performing action a_i or a_j. Each q_0 is produced by QSIM, by using s_0 or s_1 as a seed state. Planner states s_1 and s_2 are equal to the last element q_n from each state sequence.

1. Choose an action a_i to perform.
2. Given, a_i, find all state sequences $[seq_0, \ldots, seq_n]$ where s_i is used as the state to seed QSIM.
3. The set of successor states $\{s_{i+1}\}$ is the last state in each sequence $[seq_0, \ldots, seq_n]$.

Figure 5 shows the outcome of this algorithm, namely the relationship between states at the level of the planner and states from the QSIM state sequences.

The *plan* from s_{init} to s_{goal}, shown below, is a sequence of states at the planner level, actions to perform, and QSIM state sequences. The plan is strictly ordered by time, where each action at any stage of the plan is executed immediately following the completion of the action from the previous stage, and no two actions may occur simultaneously.

$$\begin{aligned} init : &(s_{init}, a_i, [q_0, \ldots]) \\ &\rightarrow (s_1, a_j, [q_0, \ldots]) \\ &\rightarrow \ldots \\ &\rightarrow s_{goal} \end{aligned}$$

5.1 Heuristic Planning

In order to use heuristic planners, various methods are presented for defining the cost of an action and the heuristic estimate from a state at the planner level to the goal. The cost of performing an action is defined as the length of the state sequence produced by QSIM. The heuristic (or estimate) for any state to a goal state is defined in relation to the *Qualitative Magnitude Distance* for a qualitative variables in the state.

The qualitative magnitude distance (QMD) for a qualitative variable, denoted as $\texttt{QMD}\left(\texttt{Dom}_1/\texttt{Dir}_1, \texttt{Dom}_2/\texttt{Dir}_2\right)$ is the number of distinct values the variable must pass though in order to continuously, and according to the rules of QSIM, transition from the qualitative value

$\texttt{Dom}_1/\texttt{Dir}_1$ the qualitative value $\texttt{Dom}_2/\texttt{Dir}_2$. The QMD is defined piecewise as:

$$\begin{aligned}
\texttt{QMD}(L_i/\text{Dir}_i, L_j/\text{Dir}_j) &= 2 + 2 * (\text{\# of landmarks between } L_i \text{ and } L_j) \\
\texttt{QMD}(L_i..L_{i+1}/\text{Dir}_i, L_j/\text{Dir}_j) &= 1 + 2 * (\text{\# of landmarks between } L_i \text{ and } L_j) \\
\texttt{QMD}(L_i/\text{Dir}_i, L_j..L_{j+1}/\text{Dir}_j) &= 1 + 2 * (\text{\# of landmarks between } L_i \text{ and } L_{j+1}) \\
\texttt{QMD}(L_i..L_{i+1}/\text{Dir}_i, L_j..L_{j+1}/\text{Dir}_j) &= 2 * (\text{\# of landmarks between } L_i \text{ and } L_{j+1})
\end{aligned}$$

where each L_i is a landmark in the domain of the qualitative variable.

Two heuristics based on the qualitative magnitude distance, *MaxQMD* and the *TotalQMD*, are defined as:

$$\begin{aligned}
\text{MaxQMD} &= \max_{v \in \{QVar\}} \texttt{QMD}(s_i(v), s_{goal}(v)) \\
\text{TotalQMD} &= \sum_{v \in \{QVar\}} \texttt{QMD}(s_i(v), s_{goal}(v))
\end{aligned}$$

where $\{QVar\}$[3] is the set of qualitative variables, and $s_i(v)$ and $s_{goal}(v)$ are the qualitative values of variable v in the respective states. MaxQMD is the maximum of the qualitative magnitude distances for all variables in the given state, while TotalQMD is the sum of the distances.

Based on the definition of the cost of performing an action, MaxQMD does not over estimate the true minimum cost of planning from the given state to the goal, proving an admissible heuristic. This is because the rules of QSIM enforce that the qualitative variable with the maximum QMD must, at a minimum, transition through the number of states given by the QMD. The remaining qualitative variables may simultaneously transition to their required values for the goal with the maximal variable. In contrast, TotalQMD typically over estimates the true cost of reaching the goal. Thus, TotalQMD is inadmissible, and may lead to less than optimal plans. MaxQMD favours reducing the distance to the goal for the variable that is the furthest away from the goal. TotalQMD favours states where the combined set of variables is closer to the goal. The behaviour and performance of the heuristics is further discussed in Section 7.

5.2 Implementation considerations

A planner that naively implements the theoretical algorithm for calculating successor states at the planner level will be very inefficient. The inefficiency is caused by the finding all state sequences given the seeding state. The total number of state sequences may be large, and many sequences will be subsets of other sequences. Instead, for the purposes of efficiency a planner may restrict the state sequences produced by QSIM to length one. It is trivial to show that any plan produced by using state sequences of length one is equivalent to a plan produced in the theoretical manner, provided that sequences of a repeated action are merged into a single action. That is, the plan $(s_0, a_i) \rightarrow s(s_1, a_i) \rightarrow \ldots \rightarrow s_n$ is equivalent to the merged plan $(s_0, a_i) \rightarrow s_n$ where the QSIM state sequence for the action is $s_1, \ldots, s_n$.

3. For partially specified states, $\{QVar\}$ is limited to only those qualitative variables appearing the in the specification of the goal.

6. Trail-and-Error Learning

The plan generated by the qualitative planner is only a general guide for how to reach the goal or complete the desired task. In particular, actions are *parameterised* by the qualitative values of the control variables and their relative rates of change. Actions are further implicitly parameterised by time. Each action in the plan is executed sequentially. However due the nature of qualitative simulation, the length of time required to execute each action is unknown. For instance, the action $\{\theta_f = 0..\pi/\texttt{inc}\}$ specifies to increase the flipper angle between the values 0 and π for an unknown duration of time. The precise parameter values are found by trial-and-error learning.

The plan allows a trial-and-error learner to restrict its trials to a much narrower range of parameter values than it would if some form of Reinforcement Learning were applied naively. For example, the constraints from the QSIM plan can be used to bound the search space for a simple form of Markov chain Monte Carlo (MCMC) sampling of the parameter space (Andrieu et al., 2003). This performs what is, essentially, a hill-climbing search of the parameter space, selecting a point in the multi-dimensional space of parameter values, testing those values, then selecting a new point based on the results of the trial. In this case, the trial is the robot's attempt to climb the ledge. This method was used successfully by Sammut and Yik (2010) to learn the parameters for a bipedal gait, given a qualitative description of the phases of walking cycle. In that work, a 23 degree of freedom robot learned a stable gait in 46 trials, averaged over 15 attempts (Yik & Sammut, 2007). The work presented here is a generalisation of Sammut and Yik because in their experiments, the planner was highly specialised for that particular task, whereas using the qualitative planning approach is much more flexible in only requiring the specification of the domain model.

7. Preliminary Results

The qualitative planner has been implemented in Prolog using the A* heuristic search technique (Hart, Nilsson, & Raphael, 1968) and a modified version of the implementation of QSIM found in Bratko (2011). The planner additionally uses the efficiency improvement discussed in Section 5.2, where only state sequences of length one are produced by QSIM. The model used for planning is the full model that is partially presented in Section 4 and Wiley et al. (2013).

The task to be solved is climbing the ledge. This task is encoded, as shown in Table 3, with an initial state on the flat ground before the ledge, and a goal state that is on top of the ledge. Specifically the robot starts at $pos_x = x_0/\texttt{std}, \theta_f = 0/\texttt{std}$, with a velocity of zero ($v = 0/\texttt{std}$). The goal state is *partially specified*, with the robot finishing at $pos_x = x_{goal}/\texttt{std}$, also with a velocity of zero.

7.1 Experiments

Experiments for planning from the specified initial state to the goal were conducted. The MaxQMD and TotalQMD heuristics were compared for performance, in terms of both the time taken to find a plan, and the length of the found plan. It was also discovered that the specification of the goal state greatly impacted the performance of the planner, thus the specification of the goal was also varied. The results of the experiments are presented in Table 4. The sequence of states in the plan produced

Table 3: Initial state and partially specified goal state for the task of climbing the ledge. Marked variables (*) are optionally included in the goal state depending on the experiment.

Initial state:	pos_x	$= x_0/\texttt{std}$	Goal State:	pos_x	$= x_{le}/\texttt{std}$
	pos_y	$= zero/\texttt{std}$		v	$= 0/\texttt{std}$
	pos_{fx}	$= x_0..x_{le}/\texttt{std}$		pos_{fx}	$= x_{goal}..\infty/\texttt{std}^*$
	pos_{fy}	$= 0/\texttt{std}$		pos_{bx}	$= x_{le}..x_{goal}/\texttt{std}^*$
	pos_{bx}	$= -\infty..x_0/\texttt{std}$			
	pos_{by}	$= 0/\texttt{std}$			
	θ_f	$= 0/\texttt{std}$			
	θ_b	$= 0/\texttt{std}$			
	θ_{fb}	$= 0/\texttt{std}$			
	v	$= 0/\texttt{std}$			

Table 4: Results of experiments with the qualitative planner. For each test, the total execution time, and number of states in the plan are reported. The heuristics and inclusion is pos_{fx} and pos_{bx} in the goal are varied across the set of tests.

No.	Heuristic	pos_{fx}	pos_{bx}	Time (sec)	Plan length
1	MaxQMD	No	No	13	9
2	TotalQMD	No	No	209	9
3	MaxQMD	Yes	No	13	9
4	TotalQMD	Yes	No	35	9
5	MaxQMD	No	Yes	1,668	11
6	TotalQMD	No	Yes	1,428	13
7	MaxQMD	Yes	Yes	1,581	11
8	TotalQMD	Yes	Yes	13	13

during Experiment 1 is presented in Table 5 as an example. and the plan of actions, constructed as a result of merging repeated actions, is presented in Table 6. For clarity, less important variables have been omitted from the presented results. The experiments were conducted on a MacBook Pro 8,1 (2GHz Intel Core i7) laptop, using the SWI-Prolog (version 6.4.1) Prolog environment. SWI-Prolog was executed with one CPU core and 4GB of memory.

7.2 Discussion

Comparing the performance of the two heuristics, neither heuristic out performs the other in all experiments. Instead, the heuristic with the best performance is greatly dependent on the specification of the goal state. TotalQMD significantly out performs MaxQMD when all three position variables (pos_x, pos_{fx} and pos_{bx}) are specified in the goal. This performance is largely due to two factors. First, as the value of velocity the goal is zero, in any state where the velocity is not zero, the estimate with TotalQMD increases by one as $\texttt{QMD}\,(v = 0/\texttt{std}, v = v_{max}/\texttt{std}) = 1$. Thus TotalQMD prefers to first visit states where velocity is zero, even though the robot must have a

Table 5: Sequence of states in the plan for Experiment 1. The states have been ordered by time.

Time	State
t_0	$v = 0/\texttt{std}$, $\theta_f = 0/\texttt{std}$, $pos_x = x_0/\texttt{std}$, $pos_y = 0/\texttt{std}$, $pos_{fx} = x_0..x_{le}/\texttt{std}$, $pos_{bx} = -\infty..x_0/\texttt{std}$, $\theta_b = 0/\texttt{std}$, $\theta_{fb} = 0/\texttt{std}$
$t_{0,1}$	$v = v_{max}/\texttt{std}$, $\theta_f = \theta_{crit}..0/\texttt{dec}$, $pos_x = x_0..max_{pxt}/\texttt{inc}$, $pos_y = 0/\texttt{std}$, $pos_{fx} = x_0..x_{le}/\texttt{inc}$, $pos_{bx} = -\infty..x_0/\texttt{inc}$, $\theta_b = 0/\texttt{std}$, $\theta_{fb} = -\frac{\pi}{2}..0/\texttt{dec}$
t_1	$v = 0/\texttt{std}$, $\theta_f = \theta_{crit}/\texttt{std}$, $pos_x = max_{pxt}/\texttt{std}$, $pos_y = 0/\texttt{std}$, $pos_{fx} = x_{le}/\texttt{std}$, $pos_{bx} = x_0/\texttt{std}$, $\theta_b = 0/\texttt{std}$, $\theta_{fb} = -\frac{\pi}{2}..0/\texttt{std}$
$t_{1,2}$	$v = v_{max}/\texttt{std}$, $\theta_f = \theta_{crit}/\texttt{std}$, $pos_x = max_{pxt}..x_{le}/\texttt{inc}$, $pos_y = 0..y_{le}/\texttt{inc}$, $pos_{fx} = x_{le}..x_{goal}/\texttt{inc}$, $pos_{bx} = x_0..x_{le}/\texttt{inc}$, $\theta_b = \theta_{crit}..0/\texttt{dec}$, $\theta_{fb} = -\frac{\pi}{2}..0/\texttt{dec}$
t_2	$v = v_{max}/\texttt{std}$, $\theta_f = \theta_{crit}/\texttt{std}$, $pos_x = x_{le}/\texttt{inc}$, $pos_y = y_{le}/\texttt{inc}$, $pos_{fx} = x_{goal}/\texttt{inc}$, $pos_{bx} = x_0..x_{le}/\texttt{inc}$, $\theta_b = \theta_{crit}..0/\texttt{dec}$, $\theta_{fb} = -\frac{\pi}{2}..0/\texttt{dec}$
$t_{2,3}$	$v = v_{max}/\texttt{std}$, $\theta_f = \theta_{crit}/\texttt{std}$, $pos_x = x_{le}..x_{fall}/\texttt{inc}$, $pos_y = y_{le}..\infty/\texttt{inc}$, $pos_{fx} = x_{goal}..\infty/\texttt{inc}$, $pos_{bx} = x_0..x_{le}/\texttt{inc}$, $\theta_b = \theta_{crit}..0/\texttt{dec}$, $\theta_{fb} = -\frac{\pi}{2}..0/\texttt{dec}$
t_3	$v = 0/\texttt{std}$, $\theta_f = \theta_{crit}/\texttt{std}$, $pos_x = x_{fall}/\texttt{std}$, $pos_y = y_{le}..\infty/\texttt{std}$, $pos_{fx} = x_{goal}..\infty/\texttt{std}$, $pos_{bx} = x_0..x_{le}/\texttt{std}$, $\theta_b = \theta_{crit}/\texttt{std}$, $\theta_{fb} = -\frac{\pi}{2}..0/\texttt{std}$
$t_{3,4}$	$v = v_{max}/\texttt{std}$, $\theta_f = \theta_{crit}/\texttt{std}$, $pos_x = x_{fall}..x_{goal}/\texttt{inc}$, $pos_y = y_{le}..\infty/\texttt{dec}$, $pos_{fx} = x_{goal}..\infty/\texttt{inc}$, $pos_{bx} = x_0..x_{le}/\texttt{dec}$, $\theta_b = \theta_{crit}..0/\texttt{inc}$, $\theta_{fb} = -\frac{\pi}{2}..0/\texttt{inc}$
t_4	$v = 0/\texttt{std}$, $\theta_f = \theta_{crit}/\texttt{std}$, $pos_x = x_{goal}/\texttt{std}$, $pos_y = y_{le}/\texttt{std}$, $pos_{fx} = x_{goal}..\infty/\texttt{std}$, $pos_{bx} = x_0/\texttt{std}$, $\theta_b = 0/\texttt{std}$, $\theta_{fb} = -\frac{\pi}{2}..0/\texttt{std}$

Table 6: The plan of action for Experiment 1, as a result of merging repeated actions. Time point are provided for reference.

Plan Stage	Time	Action (Control variable values)
1	t_0	$v = v_{max}/\texttt{std}, \theta_f = \theta_{crit}..0/\texttt{dec}$
2	t_1	$v = v_{max}/\texttt{std}, \theta_f = \theta_{crit}/\texttt{std}$
3	t_4	$v = 0/\texttt{std}, \theta_f = \theta_{crit}/\texttt{std}$

positive velocity in order to reach the goal. MaxQMD does not suffer from this, as velocity rarely influences the heuristic. It is almost always the case that the QMD for another variable is at least one. Thus, MaxQMD can expand states more likely to lead to the goal earlier than TotalQMD, reducing the runtime. Secondly, the variables pos_x, pos_{fx} and pos_{bx} tend to reduce their QMD values in parallel, and at worst one decreases while the other remains the same. Thus specifying two out of the three variables only has a small impact on the performance of TotalQMD. It is not until the three variables are specified that TotalQMD performs significantly better than MaxQMD. Therefore, MaxQMD performs better when fewer variables are specified in the goal state, and TotalQMD performs better with more variables.

The heuristics further contribute to poor performances of the planers as large numbers of states have the same cost. For example, in Experiment 7 with MaxQMD, the planner visits 14,681 states during the search. Of these states, 10.3% have a cost of 8, 28.1% have a cost of 9 and 61.6% have a

cost of 10. The reason this occurs for MaxQMD, it that typically, the cost of performing an action is one, which is equal to the reduction in the value of the heuristic. Thus the total cost for the successor state is frequently the same as the cost of its parent. In in a similar way, in Experiment 2 with TotalQMD, the planner visits 410 states, where 16.3% have a cost of 8 and 83.4% have a cost of 9. This is again because the key variable contributing to the heuristic is pos_x, which reduces it's QMD value by at most 1 for each action.

A significant decrease in performance in the planner is observed in Experiments 5 to 7. The cause is the final state of the plan. In these experiments the value of pos_{bx}, remains unchanged from the initial state and finishes with the value $pos_{bx} = x_0/\texttt{std}$. In almost all practical situations for climbing a ledge, it is physically impossible for the x-position of the base of the robot to finish at this location. However, due to the nature of using qualitative reasoning, QSIM cannot determine these states are physically impossible and allows such a goal state to be reached. Thus when pos_{bx} is specified in the goal, the planner cannot terminate early when it reaches these physically impossible states. This contributes to the longer running time of the planner, as more states must be searched, and a longer plan to reach the goal is required.

Investigating techniques to address these performances issues is left for future work. However, another performance issue to investigate other than the planner is the efficiency of QSIM.

7.3 Improving the Efficiency of QSIM

The performance of QSIM also has a significant impact on the performance of the planner, namely in the time taken to calculate all successors to the seeding state. Consider a qualitative state, with n variables. Each variable can potentially transition to up to three different values, creating a maximum of 3^n possible successor states. However in practice, only a handful of these states are allowed by the qualitative model. For example, there are only 22 possible states that can be reached from the initial state in the above experiments, compared to the almost 60,000 potential successor states. Bandelj et al. (2002), investigated this. By implementing QSIM as a constraint logic program over a finite domain, they demonstrated, depending on the application, at least an order of magnitude improvement over standard implementations of QSIM. The disadvantage to Bandelj's approach is that the landmarks were encoded numerically. This causes problems, since a viable approach to improving the efficiency of the planner is introduce limited quantitative information similar to the work of Berleant and Kuipers (1997) which also requires numerical landmarks. The two systems are not compatible.

8. Conclusions and Future Work

A qualitative planner that produces a sequence of robotic actions required to completed a task has been presented. Qualitative actions are defined in relation to Control Variables, and the Qualitative Simulation algorithm is used to produce successor states. The planner has been implemented and tested using a qualitative a model of a robotic platform for Urban Search and Rescue, and the task of climbing a ledge. The results of the preliminary experiments have been presented.

Improving the efficiency of the planner also remains an open question. This can be achieved by developing heuristics that better discriminate between states. Additionally, using purely qualitative

information allows states which are physically impossible, bloating the state space that must be searched. This could be reduced by introducing into the planner quantitative restrictions using techniques from the work of Berleant and Kuipers (1997).

The motivation for this work is to use the constraints associated with each qualitative action to reduce the search space for a learning system that turns qualitative actions into quantitative commands that can be sent to actuators. Integration of the planner and learner is the next stage in this research leading to acquiring skills that include climbing a stair case and traversing uneven terrain.

References

Aichernig, B. K., Brandl, H., & Krenn, W. (2009). Qualitative Action Systems. In *Formal methods and software engineering in the series lecture notes in computer science*, 206–225. Springer Berlin / Heidelberg.

Andrieu, C., DeFreitas, N., Doucet, A., & Jordan, M. I. (2003). An introduction to MCMC for machine learning. *Machine Learning*, *50*, 5–43.

Bandelj, A., Bratko, I., & Šuc, D. (2002). Qualitative Simulation with CLP. *Qualitative Reasoning (QR), 16th International Workshop on*.

Berleant, D., & Kuipers, B. J. (1997). Qualitative and quantitative simulation: bridging the gap. *Artificial Intelligence*, *95*, 215–255.

Bonasso, R. P., Firby, R. J., Gat, E., Kortenkamp, D., Miller, P. D., & Slack, M. G. (1997). Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental & Theoretical Artificial Intelligence*, *9*, 237–256.

Bratko, I. (2011). *Prolog Programming for Artificial Intelligence*. Addison-Wesley.

Brown, S., & Sammut, C. (2011). Learning Tool Use in Robots. *Advances in Cognitive Systems*.

Buskey, G., Roberts, J., & Wyeth, G. (2002). Online learning of autonomous helicopter control. *Australasian Conference on Robotics and Automation (ACRA), Proceedings of the* (pp. 19–24).

DeJong, G. F. (1994). Learning to Plan in Continuous Domains. *Artificial Intelligence*, *65*, 71–141.

Dietterich, T. G. (1998). The MAXQ Method for Hierarchical Reinforcement Learning. *ICML*, 118–126.

Drabble, B. (1993). EXCALIBUR: A Program for Planning and Reasoning with Process. *Artificial Intelligence*, *62*, 1–50.

Džeroski, S., De Raedt, L., & Driessens, K. (2001). Relational reinforcement learning. *Machine Learning*, *43*, 7–52.

Forbus, K. D. (1989). Introducing Actions into Qualitative Simulation. *Artificial Intelligence (IJCAI), 11th International Joint Conference on*, 1273–1278.

Gat, E. (1998). On Three-Layer Architectures. *Artificial Intelligence and Mobile Robotics*, 195–210.

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *Systems Science and Cybernetics, IEEE Transactions on*, *4*.

Hengst, B. (2002). Discovering Hierarchy in Reinforcement Learning with HEXQ. *Machine Learning (ICML), 19th International Conference on* (pp. 243–250). Morgan Kaufmann.

Hogge, J. C. (1987). Compiling Plan Operators from Domains Expressed in Qualitative recess Theory. *Artificial Intelligence (AAAI), 6th National Conference on* (pp. 229–233).

Kuipers, B. J. (1986). Qualitative Simulation. *Artificial Intelligence*, *29*, 289–338.

Michie, D., & Chambers, R. A. (1968). BOXES: An Experiment in Adaptive Control. *Machine intelligence*, *2*, 137–152.

Mugan, J., & Kuipers, B. J. (2012). Autonomous Learning of High-Level States and Actions in Continuous Environments . *Autonomous Mental Development, IEEE Transactions on*, *4*, 70–86.

Ng, A. Y., Coates, A., Diel, M., Ganapathi, V., Schulte, J., Tse, B., Berger, E., & Liang, E. (2006). Autonomous inverted helicopter flight via reinforcement learning. *Experimental Robotics IX*, 363–372.

Ryan, M. R. K. (2002). Using Abstract Models of behaviours to automatically generate reinforcement learning hierarchies. *Machine Learning (ICML), 19th International Conference on* (pp. 522–529). Morgan Kaufmann.

Sammut, C., & Yik, T. F. (2010). Multistrategy Learning for Robot Behaviours. In J. Koronacki, Z. Ras, S. Wierzchon, & J. Kacprzyk (Eds.), *Advances in machine learning i*, 457–476. Springer Berlin / Heidelberg.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press. 1st edition.

Tesauro, G. (1995). Temporal difference learning and TD-Gammon. *Communications of the ACM*, *38*, 58–68.

Troha, M., & Bratko, I. (2011). Qualitative learning of object pushing by a robot. *Qualitative Reasoning (QR), 25th International Workshop on*, 175–180.

Watkins, C. J. C. H. (1989). *Learning with Delayed Rewards*. Doctoral dissertation.

Wiley, T., Sammut, C., & Bratko, I. (2013). Using Planning with Qualitative Simulation for Multistrategy Learning of Robotic Behaviours. *Qualitative Reasoning (QR), 27th International Workshop on* (pp. 24–31).

Yik, T. F., & Sammut, C. (2007). Trial-and-Error Learning of a Biped Gait Constrained by Qualitative Reasoning. *Robotics and Automation (ACRA), 2007 Australasian Conference on*.