Qualitative Planning with Quantitative Constraints for Online Learning of Robotic Behaviours

Timothy Wiley and **Claude Sammut**
School of Computer Science and Engineering
University of New South Wales
Sydney, NSW 2052, Australia
{timothyw,claude}@cse.unsw.edu.au

Ivan Bratko
Faculty of Computer and Information Science
University of Ljubljana
Trzaska 25, 1000 Ljubljana, Slovenia
bratko@fri.uni-lj.si

Abstract

This paper resolves previous problems in the Multi-Strategy architecture for online learning of robotic behaviours. The hybrid method includes a symbolic qualitative planner that constructs an approximate solution to a control problem. The approximate solution provides constraints for a numerical optimisation algorithm, which is used to refine the qualitative plan into an operational policy. Introducing quantitative constraints into the planner gives previously unachievable domain independent reasoning. The method is demonstrated on a multi-tracked robot intended for urban search and rescue.

1 Introduction

For many complex robotic tasks, such as locomotion behaviours, it is preferable to learn the low-level controller actions. We focus on online approaches, that is, learning while an agent performs a given task. A *Multi-Strategy Architecture* (Figure 1) may be used for online learning of robotic behaviours (Wiley, Sammut, and Bratko 2013a). A planner uses a qualitative model of a robotic system to produce a parametrised sequence of actions that complete a given task, where the exact values of the parameters are found by a trial-and-error learner. However, such planners required task specific knowledge to produce correct plans. The work in this paper resolves the problem by introducing quantitative constraints into the planner. This addition, however, causes a significant performance reduction.

Online learning has typically been tackled with some form of reinforcement learning. The system performs a succession of trials, which initially fail frequently. As more experience is gained, the control policy is refined to improve its success rate. In its early formulations (Michie and Chambers 1968; Watkins 1989; Sutton and Barto 1998), reinforcement learning worked well as long as the number of state variables and actions was small. This is not the case for robotic systems that have large, continuous and highly noisy state spaces. Various approaches to alleviate this problem include building a high-level model of the system to restrict the search of a learner. Hand-crafted computer simulations of robotic systems allow a large number of trials to be run. With enough trials, robotic behaviours may be learnt such as teaching a dog-like robot to jump (Theodorou, Buchli, and Schaal 2010). Temporal data allows a learner to think multiple steps ahead reducing the risk of immediate failure and has been applied to office navigation tasks (Quintia et al. 2010). A second stage of real-world learning may further refine policies learnt in simulation and have been applied to visual AUV Navigation (El-Fakdi and Carreras 2013).

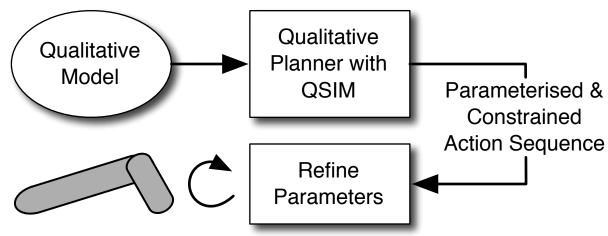


Figure 1: Three stage architecture for learning robotic behaviours using a qualitative planner based on the QSIM algorithm, and a quantitative trial-and-error learner.

Behavioural cloning observes the actions of expert humans using a robotic system to build a model that trains the learner (Michie, Bain, and Hayes-Michie 1990). The model may also be further refined in a second stage before training the learner (Isaac and Sammut 2003), which has been applied complex non-linear tasks such as to learning to fly aircraft (Šuc, Bratko, and Sammut 2004) and controlling complex non-linear container crane (Šuc and Bratko 1999). System identification removes the expert human and uses machine learning to create a characterisation of the system being controlled. Reinforcement learning then produces a controller and has applied to learn autonomous helicopter flight (Buskey, Roberts, and Wyeth 2002; Ng et al. 2006). The reinforcement learner's policy may also be modelled (or approximated) by using a combination of neural networks and a database of previously visited states. This has been applied to train a robot with articulated fins to swim (Lin et al. 2010), but does not scale well as the database grows.

1.1 Multi-Strategy Learning

Building the above types of models generally requires strong domain knowledge. However, 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, the gear stick, and the pedals. Play with them until you figure out how to drive". Rather, the instructor give an explicit 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 the right speed. This can only be learned by trial-and-error. So despite having a complete qualitative plan of actions, the student will still make mistakes until the parameters of their 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. Additionally, common physics constraints can give a learner what might be described as "common sense", in that it can reason about the actions it is performing. The background knowledge in the above example was provided by a teacher, but some or all of it could also be obtained from prior learning.

Hybrid learning frameworks have been proposed that use qualitative background knowledge with multiple stages of learning. Ryan (2002), used a symbolic qualitative planner to generate a sequence of parameterised actions, and a second stage of reinforcement learning to refine the parameters. However, the system was limited to discrete domains. Brown and Sammut (2011) used a STRIPS-like planner to generate actions for a high-level motion planner. A constraint solver was added that used the actions to limit the search of a motion planner that provided the parameters for a low-level controller. This worked well because the robot in the experiments was wheeled and only needed to drive over a flat surface, but is not appropriate for more complex tasks. Sammut and Yik (2010) proposed a Multi-Strategy Learning Architecture for complex low-level control actions, using a classical planner to produce a sequence of parameterised qualitative actions. The parameter values were bounded by a constraint solver and refined by a trial-and-error learner. Given a qualitative description of the phases of walking cycle, a stable gait for a 23 degree of freedom bipedal robot was learnt in 46 trials, averaged over 15 experiments (Yik and Sammut 2007). However, the planner was highly specialised for that particular task. We generalised this architecture (Wiley, Sammut, and Bratko 2013a) (Figure 1) to address the need for a highly specialised planner. The STRIPS-like action model was replaced with a qualitative model that specified the robot and its interaction with the environment with qualitative constraints. The planner and constraint solver were combined by using Qualitative Simulation (QSIM) (Kuipers 1986) to reason about the effect of performing actions, and automatically constrain the action parameters. The revised approach made progress toward removing domain specific knowledge from the planner, however, the purely qualitative information was not sufficient in all cases. The planner could produce sequences of actions that were physically impossible to execute.

We extend our previous work by introducing quantitative constraints into the planning process, to prevent the planner from producing impossible solutions. However, the use of quantitative constraints have significant performance impacts on the planner. This performance impact is analysed.

1.2 Related work in Qualitative Planning

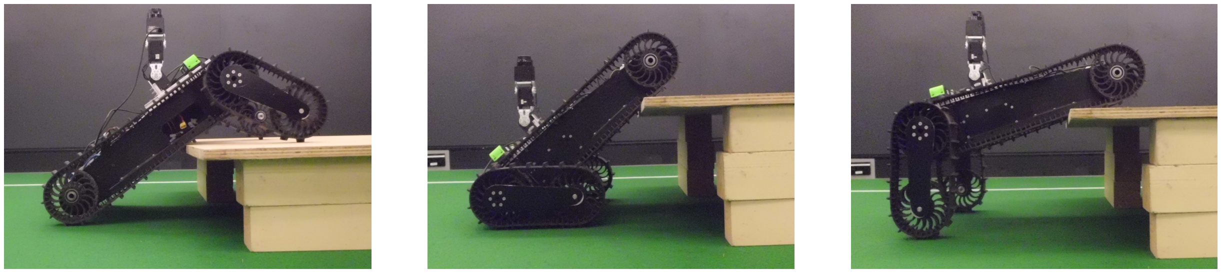
Planning with Qualitative Reasoning has been previously investigated through planning architectures for Qualitative Process Theory (QPT) (Aichernig, Brandl, and Krenn 2009; Forbus 1989; Hogge 1987). The dynamics of a system are modelled by qualitative constraints in the form of Qualitative Differential Equations (QDEs), and the planner uses STRIPS-like actions to add and remove QDEs from the model to search through potential states until the desired goal is found. For example, a Water Tank may be modelled with an out-flow pipe, in-flow pipe and a valves on each pipe. Actions open or close the valves which add and remove constraints that govern the rate of change in the level of water in the tank. This technique has been adapted for live monitoring tools to predict state of dynamic system and allow a reactive controller to maintain the system (DeJong 1994; Drabble 1993). However, these actions do not contain parameters sufficient for the trail-and-error learner.

In the robotics domain, Troha and Bratko (2011) used planning with QSIM to train 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 actions to place the object at a desired position and orientation. However, their system was specialised to learning the effects of pushing objects. Mugan and Kuipers (2012) developed the QLAP (Qualitative Learner of Action and Perception) framework. Each action sets the value of a single variable. A small quantitative controller is learnt for each action that triggers further actions, creating a hierarchy of linked actions. Tasks are performed by triggering appropriate high-level actions. The hierarchy lacks modularity as actions are strongly linked. If the configuration of the robot or environment changes the entire tree must be re-learnt.

2 Application Domain

The task of traversing rough terrain typically found in the field of Urban Search and Rescue, such as steps, staircases and loose rubble, is a complex control task. The iRobot manufactured Negotiator (Figure 2), typical of those used in the field, contains a main set of tracks to drive the robot and sub-tracks, or flippers, that can re-configure the geometry of the robot to climb over obstacles. The planner must choose the best sequence of actions to overcome terrain obstacles without becoming stuck.

Solving autonomous terrain traversal by reinforcement learning is an active field of research. Tseng et al. (2013) reduced the search space for learning to climb a staircase by using modes (such as"align" and "climb") that restrict the choice of action. Gonzalez et al. (2010) specifically modelled slip to improve the accuracy of an active drive controller. Finally, Vincent and Sun (2012) tackled the problem of climbing a step (or box) by first training a reinforcement



(a) Approach 1 (b) Approach 2, Part 1 (c) Approach 2, Part 2

Figure 2: Climbing a Step using one of two approaches. Approach 1 (a) driving forward. Approach 2 (b, c) driving in reverse.

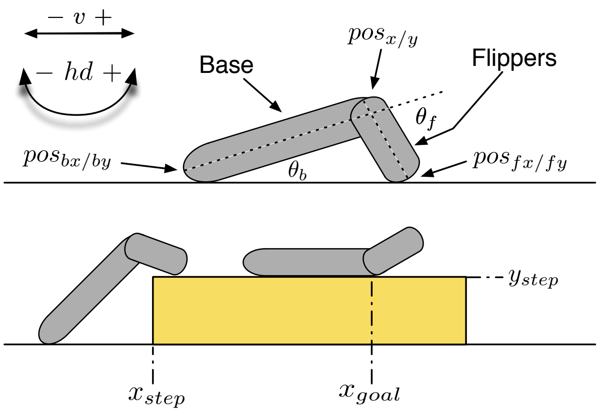


Figure 3: Negotiator and the step climbing task.

Variable	Landmarks	Description
pos_x	$[.., x_0, x_{step}, x_{goal,..}]$	Robot x-position
pos_y	$[y_{min}, y_0, y_{step}, y_{max}]$	Robot y-position
pos_{fx}	$[x_{min}, x_0, x_{step}, x_{max}]$	Flipper x-position
pos_{fy}	$[y_{min}, y_0, y_{step}, y_{max}]$	Flipper y-position
pos_{bx}	$[x_{min}, x_0, x_{step}, x_{max}]$	Base x-position
pos_{by}	$[y_{min}, y_0, y_{step}, y_{max}]$	Base y-position
θ_f	$\left[-\pi, -\frac{\pi}{2}, 0, \frac{\pi}{2}, \pi\right]$	Flipper angle
θ_b	$\left[-\pi, -\frac{\pi}{2}, 0, \frac{\pi}{2}, \pi\right]$	Base angle
hd	$[-\pi, 0, \pi]$	Heading
v	$[v_{min}, 0, v_{max}]$	Velocity

Table 1: Qualitative Variables for the Negotiator. Special landmarks such as x_0 and x_{step} are shown in Figure 3.

learning using a simulator. However, these approaches do not easily generalise if the task or robot is changed.

Following these approaches, we apply our planner to the step climbing task. This task is more complex than it may appear and requires skills essential to traversing other terrain such as stairs and rubble. The Negotiator must configure itself to climb an obstacle that is higher than its main tracks using two broad approaches (Figure 2), sometimes requiring a lengthy sequence of actions. In Approach 1 the flippers are raised above the step before the robot drives forward. Approach 2 is significantly more complex. The robot reverses up to the step then, by supporting the it's weight on the flippers, the base is raised and placed on the step. The flippers are reconfigured and the robot reverses onto the step. Tacking Approach 1 is favourable to Approach 2 as the process of supporting the robot's weight on the flippers is very unstable. However, Approach 1 cannot be used if the step is too high. Thus, a planning system should prefer Approach 1 when possible, but also be able to determine when Approach 2 is the only way to climb the step. Further, in Approach 1 it is undesirable for the Negotiator to "fall" from a height onto the step as this may damage the robot. Additionally, the robot may not have enough traction to push itself onto the step unless the flippers provide additional grip. Both problems require the flippers to be lowered onto the step before it is traversed. This reasoning was not possible when only qualitative information was used in the planner.

2.1 Qualitative Model for Climbing the Step

The Qualitative Model for the Negotiator and its interactions with the environment for the step climbing task are described by *qualitative constraints* which express relations between *qualitative variables*. Each variable is described by a magnitude relative to landmark values in the variable's domain, and a direction (increasing, decreasing or steady) of the change in the variable's magnitude over time. Special *control variables* affect changes in the qualitative state during simulation, and map directly to operations executable on the robot. Table 1 summarises the variables and their landmarks, including state variables. The control variables for Negotiator are the angle of the flippers (θ_f), the velocity (v) and heading (hd). Figure 3 shows all of the variables in relation to the Negotiator and the step climbing task. The robot starts at $pos_x = x_0$, and must climb the step to arrive at $pos_x = x_{goal}$.

Qualitative constraints are expressed as *Qualitative Differential Equations (QDEs)* which place restrictions on the magnitude and direction of change of variables. For example, the monotonicity constraint $\mathtt{M+}(x, y)$ requires that direction of change for x and y are equal, for instance, if x is increasing y also increases. Qualitative constraints are extended to *Qualitative rules* which define preconditions that indicate conditions under which each constraint in the model applies. That is, the constraints that govern the behaviour of the robot change depending on the robot's configuration or the region of environment. 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, the an-

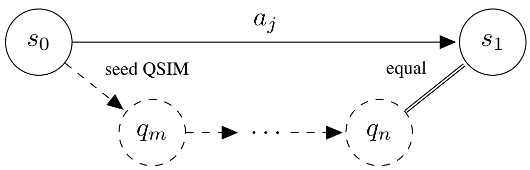


Figure 4: An instance of Qualitative Planning, calculating the transition function $(s_0, a_j) \rightarrow \{s_1\}$ for one sequence of states produced by QSIM.

gles are monotonically constrained ($\texttt{M+}(\theta_f, \theta_b)$), inversely constrained ($\texttt{M-}(\theta_f, \theta_b)$), or are unrelated ($\theta_b = 0$ as θ_f changes). The full qualitative model for the Negotiator and step climbing task is provided in our previous work (Wiley, Sammut, and Bratko 2013b).

3 Qualitative Planning

The Qualitative Planner uses the classical state-action transition function $(s_i, a_j) \rightarrow \{s_{i+1}\}$ where performing action a_j in state s_i leads to a set of successor states $\{s_{i+1}\}$. Each state s_i is a qualitative description of the robot's configuration and location with the environment. An action is defined in relation to operations executable on the robot. An action a_j is a qualitative value (magnitude and direction) for *every* control variable ($CVQar$), defined below. The result of performing a action a_j in state s_i is calculated by executing the Qualitative Simulation algorithm (QSIM).

$$a_j := \{CQVar_1 = \texttt{Mag/Dir}, \ldots\}$$

Given an initial qualitative state, the execution of QSIM produces sequences of qualitative states $[(q_0, t_0), \ldots, (q_n, t_n)]$ that the system may evolve through over a variable length of time (where state q_i occurs at time t_i). The execution may produce multiple sequences, each of varying length. QSIM uses the qualitative rules to determine which sequences are possible, that is, allowable under the qualitative model. Typically, control variables which govern how the state of a system evolves over time, may freely change value during simulation. The remaining variables may only take values allowable under the model and the operational restrictions of QSIM.

A single instance of qualitative planning is depicted in Figure 4. Given a chosen state s_i and action a_j, an execution of QSIM is seeded with s_i. For the execution, the value of each control variable is fixed to the value defined in a_j. Each successor state s_{i+1} for the action is the terminal state q_n of each sequence produced by QSIM. Therefore, a plan is the sequence of actions and states $(s_0, a_0) \rightarrow \ldots \rightarrow s_{goal}$ from the initial state s_0 to the goal state s_{goal}. The sequences produced by QSIM in calculating the effect of each action is discarded, as this precise sequence of states is not required by the Multi-Strategy Architecture. The trial-and-error learner, when optimising the parameters of the actions in a plan, will find the optimal sequence of states necessary to solve the task. In fact, the sequence of states produced by QSIM may be suboptimal.

3.1 Quantitative Constraint Propagation

As landmarks in the model are purely qualitative, the planner cannot deduce, for instance, that Approach 1 to climbing the step cannot be used if the height of the step is too great. In fact, the planner would think it was possible to climb a one kilometre high wall! This deficiency of qualitative reasoning is addressed by assigning, in the model, known quantitative values to landmarks and applying quantitative constraints (Berleant and Kuipers 1997) to the magnitude's of variables as successor states are calculated.

For monotonic constraints, such as $\texttt{M+}(\theta_f, \theta_b)$, interval arithmetic (Hyvönen 1992) places quantitative upper and lower limits on the possible value of variables. For instance, if the qualitative state of the above variables is $\theta_f = -\frac{\pi}{2}..0/\texttt{inc}$, $\theta_b = \frac{\pi}{2}..\pi/\texttt{inc}$, and the variables have quantitative lower bounds (in degrees) of -45 and 135 respectively, then in the next successor state the variables are quantitatively bound by $\theta_f = -45..0$, $\theta_b = 135..180$.

Bounds from interval arithmetic are combined with ensuring the arithmetic consistency of `sum`, `mult` and `deriv` constraints. For example, the x-position of the flipper pos_{fx} is determined by the constraint $\texttt{sum}(pos_x, scale_{fx}, pos_{fx})$ where $scale_{fx} := [-f_{length}, 0, f_{length}]$ is a scaling factor bounded by the length of the flipper (f_{length}). Qualitatively, the instantiation $\texttt{sum}(x_0, f_{length}, x_{goal})$ is legal. That is, the qualitative model allows the robot to be at the initial position before the step, and the flipper to be at the goal, which in most real-world problem is impossible. With quantitative constraints such instantiations are removed.

To demonstrate the effects of using quantitative constraints on the sequences of actions produced by the planner, experiments were conducted that changed three factors: quantitative constraints were optionally enabled, the height of the step was gradually increased when quantitative constraints were enabled, and heading of the Negotiator in the initial state was either forward (facing the step with $hd = 0$) or reverse (facing away from step with $hd = \pi$). For reference, other quantitative landmarks used in the experiments include the start location $x_0 = 0$ (the origin), step x-location $x_{step} = 100$ cm, the goal $x_{goal} = 200$ cm, and Negotiator's dimensions of the flipper (30 cm) and base (60 cm) length. The sequence of actions produced by the planner for each experiment is listed in Table 2. Without quantitative constraints, the planner uses Approach 1 and does not prevent the robot falling onto the step. With quantitative constraints, and a low step height, Approach 1 is used, but with additional actions to prevent falling. With a higher step height, Approach 2 is used as the planner deduces Approach 1 cannot be used, and with a step height that is impossible to high to climb, no plan is found.

Therefore, quantitative constraints are essential to producing the correct plan. Importantly, the quantitative constraints allow the *same* qualitative model regardless of the dimensions of the step or starting location of the robot.

3.2 Efficient Planning

The qualitative model of the Negotiator and step climbing task has an upper bound of 2×10^8 distinct qualitative states.

Quant. Enabled	Step height	Start Direction	Plan
No	-	forward	3*
No	-	reverse	3*†
Yes	20 cm	forward	3
Yes	20 cm	reverse	3†
Yes	50 cm	forward	4
Yes	100 cm	forward	-

Table 2: The calculated plan with varying configurations of the model and planner. Plan numbers refer to the Table number showing the plan. (*) plans use the qualitative-only version of the plan in the table, and (†) plans include an additional action to first turn the robot to face the step.

Action	Action (Control variable values)		
No.	v	hd	θ_f
1	$v_{max}/\mathtt{std}$	$0/\mathtt{std}$	$-\frac{\pi}{2}..0/\mathtt{dec}$
2	$v_{max}/\mathtt{std}$	$0/\mathtt{std}$	$-\frac{\pi}{2}..0/\mathtt{std}$
3	$v_{max}/\mathtt{std}$	$0/\mathtt{std}$	$-\frac{\pi}{2}..0/\mathtt{inc}$
4	$0/\mathtt{std}$	$0/\mathtt{std}$	$-\frac{\pi}{2}..0/\mathtt{std}$

Table 3: Sequence of actions from the planner following Approach 1 with a step height of 20 cm. The plan produced with the qualitative-only planner omits Action 3.

While not every state is valid, a brute force search of the state space is not sufficient, especially since using QSIM's slow generation of successor states for a model of this scale creates a bottleneck in performance.

To allow a more intelligent cost-based search, Wiley, Sammut, and Bratko (2013a) proposed calculating the cost of each state in the planner s_i using the typical cost function (as in the A* Algorithm (Hart, Nilsson, and Raphael 1968)) $c(x) = g(x)+h(x)$ with an immediate cost $g(x)$ for each action and an estimate $h(x)$ of the cost to reach the goal. The immediate cost is the number of states in the intermediate sequence for the action. The estimate is provided by *Qualitative Magnitude Distance (QMD)*. The QMD estimates the minimum number of qualitative states that are required for a *single* variable to transition from one value ($\mathtt{Mag}_1/\mathtt{Dir}_1$) to another ($\mathtt{Mag}_2/\mathtt{Dir}_2$). The QMD is defined by:

$$\begin{aligned}\mathtt{QMD}(L_i, L_j) &= 2 + 2 * lands(L_i, L_j)\\ \mathtt{QMD}(L_i..L_{i+1}, L_j) &= 1 + 2 * lands(L_i, L_j)\\ \mathtt{QMD}(L_i, L_j..L_{j+1}) &= 1 + 2 * lands(L_i, L_{j+1})\\ \mathtt{QMD}(L_i..L_{i+1}, L_j..L_{j+1}) &= 2 * lands(L_i, L_{j+1})\end{aligned}$$

where $lands(L_i, L_j)$ is the number of landmarks in the domain of the variable between landmarks L_i and L_j. For example, for θ_f, QMD$(0/\mathtt{std}, \pi/\mathtt{std}) = 4$ as θ_f transitions through the states $0..\frac{\pi}{2}/\mathtt{inc}$, $\frac{\pi}{2}/\mathtt{inc}$ and $\frac{\pi}{2}..\pi/\mathtt{inc}$.

Therefore, the estimate for a state to the goal must be accumulated across all variables in the state. Two heuristics are MaxQMD (the maximum QMD of all variables) and TotalQMD (the sum of the QMD's for all variables). MaxQMD always underestimates the cost, as trivially, the minimum number of states required to reach the goal can be no less than the number of states required for any variable to reach its desired value. TotalQMD typically overestimates the cost, but allows the planner to favour states where a greater number of variables are closer to their goal.

Action	Action (Control variable values)		
No.	v	hd	θ_f
1	$0/\mathtt{std}$	$0..\frac{\pi}{2}/\mathtt{inc}$	$0/\mathtt{dec}$
3	$0/\mathtt{std}$	$\pi/\mathtt{std}$	$-\frac{\pi}{2}..0/\mathtt{dec}$
8	$v_{max}/\mathtt{std}$	$\pi/\mathtt{std}$	$-\frac{3\pi}{2}..-\pi/\mathtt{dec}$
9	$v_{max}/\mathtt{std}$	$\pi/\mathtt{std}$	$-\frac{3\pi}{2}..-\pi/\mathtt{std}$
10	$0/\mathtt{std}$	$\pi/\mathtt{std}$	$-\frac{3\pi}{2}..-\pi/\mathtt{inc}$
14	$v_{max}/\mathtt{std}$	$\pi/\mathtt{std}$	$0..\frac{\pi}{2}/\mathtt{inc}$
15	$v_{max}/\mathtt{std}$	$\pi/\mathtt{std}$	$0..\frac{\pi}{2}/\mathtt{dec}$
17	$0/\mathtt{std}$	$\pi/\mathtt{std}$	$0/\mathtt{std}$

Table 4: Sequence of actions from the planner with quantitative constraints enabled and a step height of 50 cm, following Approach 2. Actions where control variables transition through a sequence of landmarks have been removed. Actions 1-2 rotate the robot, 3-7 raise the base, 8-9 place the base on the step, and 10-14 reconfigure the robot to reverse onto the step in 15-17.

It was previously found that the choice of heuristic is based on amount of domain specific knowledge known about the goal state (Wiley, Sammut, and Bratko 2013a). If little is known, MaxQMD should be used, and TotalQMD should be used if the value of most variables is known. This conclusion is re-evaluated under the addition of quantitative constraints.

4 Performance

To evaluate the impact of quantitative constraints, experiments were conducted that compared the performance of the existing qualitative-only planner to the planner extended with quantitative constraints. The experiments were conducted using the step climbing task. As both the choice of heuristic and specification of the goal state greatly impacted the performance of the qualitative-only planner (Wiley, Sammut, and Bratko 2013a), the experiments considered different combinations of the heuristic and goal. The results of the experiments are summarised in Table 5.

The experiments were conducted using a Prolog implementation of the qualitative planner, QSIM (Bratko 2011), and A* (Hart, Nilsson, and Raphael 1968) to provide a basic search. For each experiment, the robot was initially stationary, on the ground before the step, with the flippers directed toward the step, ($pos_x = 0$, $hd = 0$, $v = 0$). The experiments are grouped by the variables specified in the goal state. For each goal state, the choice of heuristic and the type of planner (with quantitative constraints optionally enabled) was varied. In the first set of experiments, the step height was set to 10 cm to ensure that a plan following Approach 1 was found. This was necessary to ensure comparisons could be made as the qualitative-only planner ran out of memory when attempting to find plans following Approach 2. Finally, for reference a set of experiments was conducted with a step height of 30 cm. The performance of the planner is compared by the number of inferences Prolog evaluates to find a plan as it is independent of variations in a specific CPU. However, the number of inferences increases proportionally to total execution speed, and the number of operations performed in calculating successor states and the

Goal Variables					Approach	Qualitative		Quantitative	
v	pos_x	pos_{fx}	pos_{bx}	θ_f		MaxQMD	TotalQMD	MaxQMD	TotalQMD
x	x				1	4.17×10^7	1.48×10^8	1.28×10^9	1.47×10^9
x	x	x			1	4.18×10^7	1.51×10^8	1.15×10^9	3.17×10^8
x	x		x		1	9.25×10^8	1.99×10^8	3.80×10^8	2.03×10^8
x	x			x	1	4.83×10^7	9.78×10^7	1.70×10^9	1.35×10^9
x	x	x	x	x	1	9.26×10^8	1.37×10^8	3.71×10^8	1.05×10^8
x	x				2	-	-	1.80×10^9	2.19×10^9

Table 5: Experimental Results. The number of inferences evaluated by the Prolog implementation of each combination of the planner type and heuristic for each specification of the goal state.

cost-based search.

The results show three key trends. First, the planner extended with quantitative constraints performs, at best, on par with the qualitative-only planner when all variables are specified in the goal state. Although the quantitative constraints reduces the total number of possible states to search through, in practice specifying most variables in the goal state causes both planner to evaluate a similar sequence of states during the search. The difference is that with quantitative constraints, the plans are physically possible to execute. the reduction is not overly significant, and the planner still evaluates a similar set of states. However, the use of quantitative constraints introduces a degradation in performance if few variables are specified in the goal state. The reason for this performance is that although the the qualitative-only planner "cheats". Consider the step climbing task. The variables pos_{bx} and pos_{fx} are largely unbound and appear in few of the qualitative rules in model. Without quantitative constraints the planner "cheats", for example, by leaving pos_{bx} unchanged from its value in the initial state. This allows the planner find a solution significantly faster, as changing pos_{bx} activates different rules placing additional restrictions on other variables, limiting the number of valid states. To a lesser extent this is that case for pos_{fx}, however, the variable is forced change in value for a successful plan.

Secondly, the comparative performance between the MaxQMD and TotalQMD heuristics is similar for the two types of planners. That is, as more variables are introduced MaxQMD performs comparatively worse than TotalQMD. However, certain variables such as θ_f, pos_{bx} and pos_{fx} have greater influences on the comparative differences, indicating these are "critical variables" to successfully solving the task. Consider the impact of changing θ_f which influences a number of other variables including pos_{fx}. Frequently, changing θ_f makes no progress toward the goal. Without θ_f in the goal, MaxQMD focuses on the variables changes that lead to the goal, which TotalQMD is unable to exploit. However with θ_f in the goal MaxQMD gets hung up on changes in that variable, which TotalQMD is better able to ignore.

Finally, quantitative constraints enables the planner to find a result with a step height of 30 cm. This further demonstrates the necessity for quantitative constraints. The same trends for the performance of the heuristic noted above have been observed with plans following Approach 2.

Therefore it would be ideal to use quantitative constraints, the TotalQMD heuristic and have all "critical variables" included in the goal state. However, to ensure domain specific reasoning is not introduced into planning, this may not be possible. Consider specifying pos_{bx} in goal. If x_{goal} is far enough from x_{wall}, the goal should be $pos_{bx} = x_{step}..x_{goal}$. However, if the step is short, the base will overhang the step with the goal $pos_{bx} = x_0..x_{goal}$. This reasoning is domain specific and is disallowed. More generally, without detailed domain specified reasoning, the variables which have the greatest impact cannot be known, thus it cannot be ensured those variables are specified in the goal. Therefore, it can only be concluded that quantitative constraints ensure the plan can be executed on the robot but cause significant performance issues.

5 Future Work

The Prolog implementation is still inefficient, and in some experiments could not find a solution. ASPQSIM, is a more efficient implementation of QSIM and the qualitative planner using Answer Set Programming (Wiley, Sammut, and Bratko 2014). However, efficiently introducing quantitative constraints into ASPQSIM is an unsolved problem.

The sequence of actions generated by the planner is only a general guide to how to climb the step. The actions are *parameterised*, which only specify the magnitude for control variables and non-uniform time intervals during which each action is executed. The precise parameter values will be found by trial-and-error learning, such a simple form of Markov-Chain Monte Carlo (MCMC) sampling of the parameter space (Andrieu et al. 2003). The plan restricts the learner's trials to a much narrower range of parameter values than it would if some form of Reinforcement Learning were applied naively. This method was successfully used to learn the parameters for a bipedal gait (Sammut and Yik 2010).

6 Conclusion

This work resolves previous problems within the three stage Multi-Strategy architecture for learning robotic behaviours, specifically by removing the need to introduce domain specific knowledge into the planner. Quantitative constraints have been added to a qualitative planner that enables the system to reason about the effect of performing actions and determine when a sequence of actions cannot be executed due to particular features of the robot or the environment. However, the benefit of using quantitative constraints comes at a significant cost, especially in keeping domain specific information out of the planner.

References

Aichernig, B. K.; Brandl, H.; and Krenn, W. 2009. Qualitative Action Systems. In *Formal Methods and Software Engineering in the series Lecture Notes in Computer Science*. Springer Berlin / Heidelberg. 206–225.

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

Berleant, D., and Kuipers, B. J. 1997. Qualitative and quantitative simulation: bridging the gap. *Artificial Intelligence* 95(2):215–255.

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

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

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

DeJong, G. F. 1994. Learning to Plan in Continuous Domains. *Artificial Intelligence* 65(1):71–141.

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

El-Fakdi, A., and Carreras, M. 2013. Two-step gradient-based reinforcement learning for underwater robotics behavior learning. *Robotics and Autonomous Systems* 61(3):271–282.

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

Gonzalez, R.; Fiacchini, M.; Alamo, T.; Luis Guzman, J.; and Rodriguez, F. 2010. Adaptive Control for a Mobile Robot Under Slip Conditions Using an LMI-Based Approach. *European Journal of Control* 16(2):144–155.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *Systems Science and Cybernetics, IEEE Transactions on* 4(2).

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

Hyvönen, E. 1992. Constraint reasoning based on interval arithmetic: the tolerance propagation approach. *Artificial Intelligence* 58(1-3):71–112.

Isaac, A., and Sammut, C. 2003. Goal-directed learning to fly. In *Machine Learning (ICML), 20th International Conference on*, 258–265.

Kuipers, B. J. 1986. Qualitative Simulation. *Artificial Intelligence* 29(3):289–338.

Lin, L.; Xie, H.; Zhang, D.; and Shen, L. 2010. Supervised Neural Q-learning based Motion Control for Bionic Underwater Robots. *Journal of Bionic Engineering* 7:S177–S184.

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

Michie, D.; Bain, M.; and Hayes-Michie, J. 1990. Cognitive models from subcognitive skills. *Knowledge-base Systems in Industrial Control*.

Mugan, J., and Kuipers, B. J. 2012. Autonomous Learning of High-Level States and Actions in Continuous Environments . *Autonomous Mental Development, IEEE Transactions on* 4(1):70–86.

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

Quintia, P.; Iglesias, R.; Rodriguez, M. A.; and Regueiro, C. V. 2010. Simultaneous learning of perception and action in mobile robots. *Robotics and Autonomous Systems* 58(12):1306–1315.

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

Sammut, C., and Yik, T. F. 2010. Multistrategy Learning for Robot Behaviours. In Koronacki, J.; Ras, Z.; Wierzchon, S.; and Kacprzyk, J., eds., *Advances in Machine Learning I*. Springer Berlin / Heidelberg. 457–476.

Šuc, D., and Bratko, I. 1999. Symbolic and qualitative reconstruction of control skill. *Electronic Transactions on Artificial Intelligence* 3:1–22.

Šuc, D.; Bratko, I.; and Sammut, C. 2004. Learning to fly simple and robust. In *Machine Learning (ECML), European Conference on*, 407–418.

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

Theodorou, E. A.; Buchli, J.; and Schaal, S. 2010. A Generalized Path Integral Control Approach to Reinforcement Learning. *Journal of Machine Learning Research* 11:3137–3181.

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

Tseng, C.-K.; Li, I.-H.; Chien, Y.-H.; Chen, M.-C.; and Wang, W.-Y. 2013. Autonomous Stair Detection and Climbing Systems for a Tracked Robot. In *System Science and Engineering (ICSSE), IEEEE International Conference on*, 201–204.

Vincent, I., and Sun, Q. 2012. A combined reactive and reinforcement learning controller for an autonomous tracked vehicle. *Robotics and Autonomous Systems* 60(4):599–608.

Watkins, C. J. C. H. 1989. *Learning with Delayed Rewards*. Ph.D. Dissertation.

Wiley, T.; Sammut, C.; and Bratko, I. 2013a. Planning with Qualitative Models for Robotic Domains. In *Advances in Cognitive Systems (Poster Collection), Second Annual Conference on*, 251–266.

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

Wiley, T.; Sammut, C.; and Bratko, I. 2014. Qualitative Simulation with Answer Set Programming. Technical report, The School of Computer Science and Engineering, University of New South Wales.

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