Increasing the Numeric Expressiveness of the Planning Domain Definition Language

Simon Parkinson and **Andrew P. Longstaff**
Centre for Precision Technologies
School of Computing and Engineering
University of Huddersfield, UK
s.parkinson@hud.ac.uk
a.p.longstaff@hud.ac.uk

Abstract

The technology of artificial intelligence (AI) planning is being adopted across many different disciplines. This has resulted in the wider use of the Planning Domain Definition Language (PDDL), where it is being used to model planning problems of different natures. One such area where AI planning is particularly attractive is engineering, where the optimisation problems are mathematically rich. The example used throughout this paper is the optimisation (minimisation) of machine tool measurement uncertainty. This planning problem highlights the limits of PDDL's numerical expressiveness in the absence of the square root function. A workaround method using the Babylonian algorithm is then evaluated before the extension of PDDL to include more mathematics functions is discussed.

Introduction

Currently, PDDL provides four arithmetic operators ($+$, $-$, $/$, $*$) that can be used in preconditions, effects, durative-action, continuous effects, and conditional effects (Fox & Long 2003). Combining these operators will often provide the level of mathematical expressiveness that is required for abstract modelling of planning problems. However, when modelling planning problems that are mathematically rich, it is found that the expressive power of PDDL is not enough to allow for a complete, real-world representation. In some planning problems, abstractions can be made that can remove some of the calculations and still provide an accurate and useful representation that is sufficient for finding optimal and complete plans.

Previous work on creating a PDDL 2.2 encoding of the temporal problem of minimising machine tool down-time (Parkinson *et al.* 2012b) showed that automated planning could produce complete and optimal calibration plans. In addition to the PDDL encoding, a Hierarchical Task Network (HTN) encoding was created (Parkinson *et al.* 2011a) and a preliminary tool was developed to aid with the construction of calibration plans in PROLOG (Parkinson *et al.* 2011b). This work was then verified by conducting industrial tests where the calibration plans produced by an industrial expert were compared against the automatically constructed plan (Parkinson *et al.* 2012a). Although optimising calibration plans to reduce machine tool down-time is highly beneficial to the engineering community, optimising calibration plans to reduce the estimated measurement uncertainty is very useful when working on high precision machines that are manufacturing to small tolerances. Recently, the authors have been working on modelling the measurement uncertainty minimisation problem by extending the temporal PDDL 2.2 encoding. In PDDL it is possible to model the continuous and conditional effects that are required in the measurement uncertainty estimation formula. However, one of the equations is to calculate the standard deviation, and therefore requires the use of the square root function (ISO230-2 2006), which is not currently not part of regular PDDL.

The numerical expressive power of PDDL is sufficient for modelling many problems that are challenging for many planners (Fox, Long, & Magazzeni 2011; Ruml *et al.* 2011). Currently, planner development in continuous time-dependent costs has resulted in the implementation of linear programming to solve mixed integer problems (Coles *et al.* 2012). Planning problems of this nature are common and very difficult to solve, hence why planner development has been focused in this area. One problem that has been successfully solved is the liner shipping fleet repositions problem that has temporal, continuous time-dependent costs conditional effects (Tierney *et al.* 2012). This implementation of PDDL domains across many different disciplines is a success of its current expressiveness. The authors believes that increasing the numerical expressiveness will increase the versatilely and become more attractable for many more, real-world applications. Fox et. al (Fox & Long 2002) describe PDDL's expressive power as being strongly driven by potential applications. This paper continues with this philosophy and presents the idea of increasing the set of mathematical functions available when modelling problems in PDDL.

The first part of this paper provides a brief background to PDDL arithmetic operators, highlighting the limitations. Next, the measurement uncertainty optimisation problem is shown as a mathematically rich example. This planning problem requires the use of the square root function which is not currently available in PDDL. A workaround solution is presented, where a square root algorithm is encoded in the form of an PDDL action using the Babylonian method. The

```
(:durative-action transfer-money
    :parameters (?acc1 - Account ?acc2 - Account)
    :duration(= ?duration (+(trns-out-time ?a1)
          (trns-in-time ?a2)))
    :condition
       (and (at start (>=(trns-amt a1)(bal ?a1)))
       (at start (>=(+(bal a2)(trns-amt ?a1))
          (acc-limit ?a2))))
    :effect
       (and (at end (decrease(bal a1)
          (trns-amt ?a1)))
       (at end (increase(bal a2)
          (trns-amt ?a1)))
       (at end (increase(trns-amt a1)
          (#t *(interest-rate ?a1)))))
)
```

Figure 1: PDDL action for moving money between two bank accounts held with the same bank.

idea of increasing the numerical expressiveness of PDDL is then discussed. A brief conclusion is then provides, highlighting the direction of future work.

PDDL Arithmetic Operators

Current set

PDDL provides access to four arithmetic operators (`+, -, /, *`) (Fox & Long 2003). These operators can be used in many aspects of an PDDL action. For example, consider the following PDDL code seen in Figure 1 for transferring money between two bank accounts held with the same bank. The action uses the addition arithmetic operator (+) in the precondition statement checking that the amount to transfer does not take the destination account over its limit. This precondition will be required if the destination account is governed by the tax free amount a person can save per year. The duration of the action also uses the addition arithmetic operator to sum the transaction times of the source and destination account. The durative action uses the multiplication ($\star$) arithmetic operator in the time-dependent effect to apply the correct amount of interest to the source account. Other arithmetic operators can be used in the same way to implement numeric pre, post and durative conditions.

Expressive limitations

The expressive power of PDDL has allowed for the modelling of many complex real-world problems that have significantly motivated planner development. However, the following planning problem highlights that PDDL is not numerically expressive enough for all applications. The rest of this section will briefly present the measurement uncertainty optimisation (minimisation) planning problem, identifying the useful features that PDDL provides, but also highlighting the limitations with the numerical expressiveness. In the section we also consider a possible solution of implementing the square root function in regular PDDL by using the Babylonian method.

Measurement uncertainty

Measurement uncertainty can be regarded as the doubt that exists about the result of any measurement and a confidence level of the assessment (Bell 2001). When estimating measurement uncertainty there are lots of factors that must be considered that involve; (1) the measuring device, (2) the object to be measured, and (3) the environment (Birch 2003). One method to estimate uncertainty then involves combining the individual uncertainties using the root sum of squares to produce a combined uncertainty u_c. It is possible to optimise (minimise) the measurement uncertainty by planning the measurement procedure carefully. Two of the influencing factors and how they can be implemented in PDDL are described below.

1. The temperature of the environment might influence the measurement object and the measurement device. It is possible for the temperature to change considerably throughout the twenty four hour cycle in an industrial environment. It is, therefore, necessary to encode the temperature profile into the model so that the temperature over the duration of the measurement is known. The temperature can then be used when estimating u_c to help provide the best estimation as possible.

2. Instrument drift can be described as a gradual change in respect to an instrument's reference value with which measurements are made. This is mainly a concern for electronic measurement equipment, such as temperature sensors and laser interferometers. It is important to consider the instruments state when calculating u_c, and in PDDL this can easily be modelled using durative actions with time-dependent effects.

Implementation

Here we consider the estimation of measurement uncertainty for measuring the positioning accuracy of a machine tool using a laser interferometer as seen in (ISO230-2 2006).

Formula

$$U(M) = \sqrt{U_D^2 + U_M^2 + U_{M,M}^2 + U_{M,D}^2 + U_{E,M}^2 + U_{E,D}^2 + \tfrac{1}{2}.U_{EVE}^2}$$

where $U(M)$ is the measurement uncertainty of mean positional deviation, and the following uncertainties are expressed:
U_D is the uncertainty due to the measurement device. U_M is due to misalignment. $U_{M,M}$ is due to temperature measurement of the machine tool and $U_{M,D}$ for the measurement device. $U_{E,M}$ is due possible error in the thermal expansion coefficient of the machine tool and $U_{E,D}$for the measuring device. $U_{EVE}$is the uncertainty due to environmental variation. In this paper the author has omitted how the above uncertainties are calculated because the lengthy process is not relevant to the undertaken work as they can be achieved using the combination of the standard PDDL arithmetic operators. However, it is necessary to note that both the environmental temperature and instrument drift are used in their calculation.

```
(at end (increase(u_m)
        (+(+(+(+(+(*(u_D)(u_D))
            (*(u_M)(u_M)))
            (*(u_M-M)(u_M-M)))
            (*(u_M-D)(u_M-D)))
            (*(u_E-M)(u_E-M)))
            (*(u_E-D)(u_E-D)))
            (*(/(1)(2))(*(u_EVE)(u_EVE))))))
```

Figure 2: PDDL encoding of the uncertainty equation

```
(:durative-action calculate-sqrt
     :duration(= ?duration 1)
     :condition
     (and
         (at start (<=(current-step)(number)))
         (at start (start))
     )
     :effect
     (and
         (at start(increase(calculated-sqrt)
         (*(+(calculated-sqrt)(/(number)
            (calculated-sqrt)))0.5)))
            (at end(increase(current-step)1))
     )
)
```

Figure 3: Partial PDDL encoding to calculate the square root using the Babylonian method

PDDL

Figure 2 shows the implementation of the formula to calculate $U(M)$. However, the equation is incomplete because it does not implement the square root function. It would still be useful to create a model that minimises a sequence of measurement tasks and aims to reduce the combines uncertainties without using the square root function. Creating a model in this way would require post-processing the produced plan to complete the calculation. Additionally, many planners will not accept this PDDL encoding because it is a non-linear equation.

$$\begin{aligned} x_0 &\approx \sqrt{S} \\ x_{n+1} &= \frac{1}{2}(x_n + \frac{S}{x_n}) \\ \sqrt{S} &= \lim_{n\to\infty} x_n \end{aligned} \tag{1}$$

Encoding workaround

In the absence of the square root function, it is possible to encode a method which can enumerate the square root for a given value. In Figure 3 a PDDL encoding is shown that uses the Babylonian method that is shown in Equation 1 to calculate the square root. This method will calculate the square root for a given number S, however, the number of iterations i required is equal S. It is often the case that a very close approximation will be produced within only a few iterations. However, imposing an iteration limit will depend upon the application and the desired level of accuracy.

Iteration(i)	Result (x_n)	Difference ($x_{n-1} - x_n$)
1	5.50000000	4.50000000
2	3.65909091	1.84090909
3	3.19600508	0.46308583
4	3.16245562	0.03354946
5	3.16227767	0.00017796
6	**3.16227766**	0.00000001
7	3.16227766	0.00000000
8	3.16227766	0.00000000
9	3.16227766	0.00000000
10	3.16227766	0.00000000

Table 1: Results of using the Babylonian method in PDDL 2.1 to calculate the square root of 10. The correct result to eight decimal places is 3.16227766. In the first iterations $x_n = 10$.

Experimental Analysis

An example can be seen in Table 1 where the square root for the number 10 is calculated. In the table it is noticeable that after six iterations, the Babylonian method correctly calculates the square root to eight decimal places. It is also noticeable that the difference between the current calculation x_{n+1} and the result with the previous calculation x_n converges to zero to eight decimal places after seven iterations, meaning that the correct. This shows that the correct square root was calculated in the seventh iteration. However, if accuracy to only two decimal places was required, it would be possibly to stop after six iterations.

Figure 4 shows the difference between x_{n+1} and x_n per iteration i when calculating the square root for the value 1000 using the Babylonian method. The experiment was performed using the PDDL encoding as seen in Figure 3. Although it is noticeable that it only took eleven iterations to converge, the LPG-td planner (Gerevini, Saetti, & Serina 2003) performed 1000 iterations, which are not all required. Solving this problem alone took 4.66 sections, but could be significantly reduced if not all iterations were executed.

Combining this implementation with other models would most likely have an adverse affect on plan generation and quality. This shows that even if it is always possible to encode an algorithm that could perform the desired mathematical function in PDDL, it could be regarded as excessive modelling effort.

Extended set

As a potential solution we propose extending the set of mathematical functions in PDDL. The author does realise that this would not come without any complications. The following section briefly discusses some of the complications and provides future challenges for planner development.

Firstly, deciding upon the set of functions could be con-

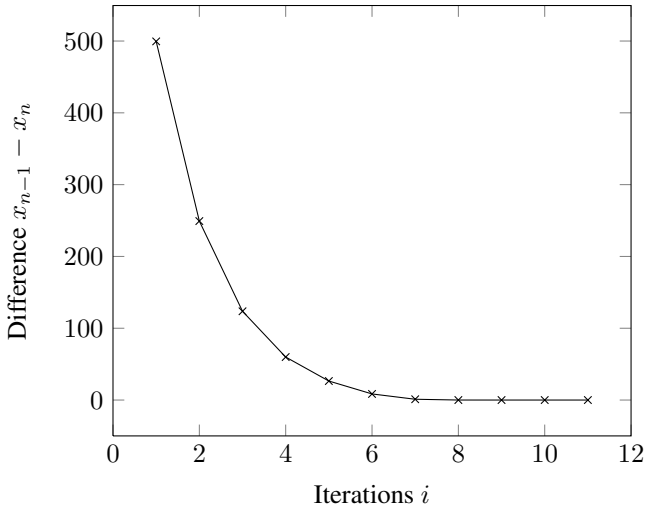

Figure 4: Results from calculating the square root for the number 1000 using the Babylonian method. The graph only shows the first eleven iterations because this is where the method converges. The remaining iterations do not produce an x_{n+1} that is different from x_n to eight decimal places.

tentious. As different communities continue to adopt the use of PDDL, different mathematical requirements will appear. The author suggests that providing access to the functions found in the cmath (math.h) class would be highly beneficial and comprehensive enough for most applications. This includes trigonometric, hyperbolic, exponential and logarithmic, power, rounding, absolute value and remainder functions.

Secondly, modifying currently available planners to cater for a richer set of mathematics functions could result in significant effort, so the author does not suggest making this extension a requirement. For example, OPTIC(Tierney *et al.* 2012) and COLIN(Coles *et al.* 2012) do not support non-linear equations, and with the availability of these functions, there would be more instances of people modelling non-linear equations. Carrying out major modification work to make the planner accept non-linear equations might not be feasible. Additionally, further research will also be required to examine the effect that these mathematical functions might have on a planner's heuristic functions, as they might impede performance and plan quality. The author suggests that increasing the mathematics function set should be taken into consideration for future planner developments to increase their applicability in other disciplines, not just artificial intelligence planning research.

The implementation of the additional functions into the PDDL language would require additional thought into how they are going to be represented. The mathematical functions that have an ASCII symbol, such as the square root and power function should be recognisable by a PDDL text parser and probably would not be used in any other part of the PDDL model. However, trigonometric functions such as `cos` and `sin` do not have a symbolic representation and would require string representation, which could cause significant interference with functions, actions and variable names in PDDL. A possible solution would be to have a special, unused character that could be used to denote the start of a maths function. However, in an effort to simplify things it would be better to prevent the names of mathematical functions being used as a string anywhere else in the PDDL model other than to express formula.

Conclusions

This paper presents the limitations of the current PDDL arithmetic operators. The measurement uncertainty minimsation problem is shown as an example that requires the square root function if the PDDL encoding is to model the real-world process without any abstractions. A possible workaround is presented in the form of a PDDL encoding to calculate the square root using the Babylonian method. Although this method works, it is infeasible to over complicate the model to implement mathematical functions. The author then suggested the extension of the available mathematical functions to provide access to the cmath (math.h) class.

Further work is being carried out by the author to adapt a current PDDL parser and planner to accept more mathematics functions. This will then allow for a complete representation of the measurement uncertainty domain, which can subsequently be used to provide a benchmark.

Alongside this future work, the authors will also continue to consider the implementation of solutions that do not require planner modification. One area that will be explored is the modification of the Babylonian PDDL encoding to reduce the iteration size. A method to estimate the iterations required to converge will be explored. One possible solution could be to use of a preprocessor to estimate the number of iterations required which could subsequently be included in the PDDL domain to restrict the number of iterations.

Acknowledgement

The authors gratefully acknowledge the UK's Engineering and Physical Sciences Research Council (EPSRC) funding of the Centre for Advanced Metrology under its innovative manufacturing program.

References

Bell, S. 2001. A beginner's guide to uncertainty of measurement. *Measurement Good Practice Guide* (11).

Birch, K. 2003. Estimating uncertainties in testing. *Measurement Good Practice Guide* (36).

Coles, A. J.; Coles, A. I.; Fox, M.; and Long, D. 2012. COLIN: Planning with Continuous Linear Numeric Change. *Journal of Artificial Intelligence Research* 44:1–96.

Fox, M., and Long, D. 2002. Pddl+: Modelling continuous time-dependent effects.

Fox, M., and Long, D. 2003. Pddl2.1: An extension to pddl for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20:2003.

Fox, M.; Long, D.; and Magazzeni, D. 2011. Automatic Construction of Efficient Multiple Battery Usage Policies. In *ICAPS*, 74–81.

Gerevini, A.; Saetti, A.; and Serina, I. 2003. Planning through stochastic local search and temporal action graphs in lpg. *J. Artif. Intell. Res. (JAIR)* 20:239–290.

ISO230-2. 2006. Part 2: Determination of accuracy and repeatability of positioning numerically controlled axes.

Parkinson, S.; Longstaff, A. P.; Crampton, A.; Andrew, P.; Fletcher, S.; Allen, G.; and Myers, A. 2011a. Hierarchical Task Based Process Planning For Machine Tool Calibration. In *Proceedings of The 29th Workshop of the UK Planning and Scheduling Special Interest Group*, 53–60. PlanSIG2011.

Parkinson, S.; Longstaff, A. P.; Crampton, A.; Andrew, P.; Fletcher, S.; Allen, G.; and Myers, A. 2011b. Representing the Process of Machine Tool Calibration in First-order Logic. In *Proceedings of the 17th International Conference on Automation & Computing*. Chinese Automation and Computing Society.

Parkinson, S.; Longstaff, A.; Fletcher, S.; Crampton, A.; and Gregory, P. 2012a. Automatic planning for machine tool calibration: A case study. *Expert Systems with Applications* 39(13):11367 – 11377.

Parkinson, S.; Longstaff, A. P.; Crampton, A.; and Gregory, P. 2012b. The application of automated planning to machine tool calibration. In McCluskey, T.; Williams, B.; Silva, J. R.; and Bonet, B., eds., *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling*, ICAPS 2012. California, USA: AAAI Press. 216–224.

Ruml, W.; Do, M. B.; Zhou, R.; and Fromhertz, M. P. J. 2011. On-line Planning and Scheduling : An Application to Controlling Modular Printers. *JAIR* 40:415–468.

Tierney, K.; Coles, A. J.; Coles, A. I.; Kroer, C.; Britt, A.; and Jensen., R. M. 2012. Automated planning for liner shipping fleet repositioning. In *Proceedings of the Twenty Second International Conference on Automated Planning and Scheduling (ICAPS-12)*.