A Hybrid Relaxed Planning Graph–LP Heuristic for Numeric Planning Domains

Andrew Coles, Maria Fox, Derek Long and Amanda Smith
Department of Computer and Information Sciences,
University of Strathclyde, Glasgow, UK
email: firstname.lastname@cis.strath.ac.uk

Abstract

Effective search control for numeric planning domains, in which appropriate numeric resource usage is critical to solving the problem, remains an open challenge in domain-independent planning. Most real-world problems rely on metric resources such as energy, money, fuel or materials. Despite the importance of numbers, few heuristics have been proposed to guide search in such domains. Hoffmann's extended relaxation, implemented in Metric-FF, is one of the best general heuristics. We examine the behaviour of the Relaxed Planning Graph (RPG) heuristic, used by Metric-FF, in numeric problems. While effective in problems with simple numeric interactions, it has two weaknesses when numeric reasoning is a fundamental part of solving the problem. We present a new heuristic for use in strongly numeric domains, using a Linear Program to capture numeric constraints as an adjunct to a relaxed planning graph. We demonstrate that an intelligent combination of these two techniques offers greatly improved heuristic guidance.

1 Introduction

Domain-independent planning has mainly focussed on propositional models. Although there is important work on planning in models with numbers, since the proposal for a standard formulation of such problems in PDDL2.1 (Fox & Long 2003) there have been relatively few significant developments in heuristics to guide search for plans in such domains. The most influential domain-independent heuristic for managing numbers is Hoffmann's extension of the relaxation approach from logical to metric fluents (Hoffmann 2003). This approach has been widely adopted, forming the basis of treatment of numbers in SGPLAN (Chen, Wah, & Hsu 2006) and MIPS (Edelkamp 2003) for example. Other approaches have been explored, although with more limited success (Do & Kambhampati 2001; Koehler 1998).

Numbers play a central role in many real planning problems, since they are the natural tool for capturing quantities of resources, including fuel, money and materials. Most real problems are greatly influenced by the use of such resources and the quality of plans is typically measured in terms of the consumption of one or more of these resources. However, planners are not very effective at solving problems that involve complex numeric interactions and are typically insensitive to quality metrics.

In contrast, in optimisation research, solving problems with numbers is the core. The most important techniques for solving constraint problems with numbers are based on linear programming, for which powerful solvers now exist. Mixed-Integer Programs (MIPs), in which variables may be constrained to be integer-valued, are solved by a combination of search and the use of relaxations as linear programs. Although there have been efforts to exploit linear programming in planning (van den Briel *et al.* 2007; Long & Fox 2003a), relatively little work has considered the exploitation of linear programming techniques to improve the behaviour of *numeric* domain-independent planners (Kautz & Walser 2000; Shin & Davis 2005; Wolfman & Weld 2000; Benton, van den Briel, & Kambhampati 2007).

In this paper, we re-examine planning with numeric resources, with particular focus on the behaviour of the Metric Relaxed Planning Graph (RPG) heuristic used in Metric-FF (Hoffmann 2003). We explore the behaviour of this heuristic, demonstrating how very typical patterns of interactions in numeric planning domains can lead to uninformative heuristic guidance, particularly when domains allow exchanges between metric variables. To address this, we introduce a novel heuristic based on a linear program (LP), used alongside the RPG, to capture the numeric behaviour. Having described the LP, and how it can be used to complement the RPG, we present static analysis techniques and an LP-relaxation reformulation to reduce LP overheads whilst still providing useful guidance. Finally, we evaluate the hybrid LP–RPG heuristic by comparing the search guidance it provides with the baseline metric RPG heuristic.

2 Background

PDDL2.1 extends PDDL to include both temporal and numeric action constraints. Numeric values are maintained by *functions* which map combinations of entities onto numeric values. Such numeric values can be used in describing *numeric preconditions* and *effects* of actions. These can be used to model, for example, the space available in a vehicle or how much energy an action uses. Hoffmann (Hoffmann 2003) proposed a way to plan in numeric domains using forward-chaining state-space search. As the numeric preconditions of actions are specified in terms of function values in a

state and their effects by changes to these, forward-chaining search is a convenient approach to numeric planning. Values are known in the initial state and can be updated alongside the logical state as actions are applied.

2.1 Ignoring Delete Effects

Hoffmann extended the relaxation from the propositional part of a domain (ignoring delete effects) to numeric elements. To achieve this, certain restrictions must be made to the language: only linear pre- and postcondition expressions (Linear Numeric Expressions or LNEs) can be handled. This is a powerful subset of the language, forming the basis of a huge body of work in optimisation research. LNE preconditions must be of the form $\mathbf{w}.\mathbf{x}$, $\{\geq, >\}$, c and effects of the form x, $\{increase, decrease, =\}$, $\mathbf{w}.\mathbf{x}$, where $\mathbf{x}$ is the vector of numeric variables and $\mathbf{w}$ a vector of constants. The translation of more general linear expressions into this form is automated: equality preconditions are replaced with a pair of preconditions, one $\leq$ and one $\geq$ and algebraic transformations turn $<, \leq$ comparisons into $>, \geq$.

Once all preconditions and effects are in this form, it is possible, by analogy with ignoring delete effects, to ignore negative numeric effects. With respect to a precondition $x > c$, effects that increase x are positive and effects that decrease it are negative. To handle preconditions of the form $-x > c$, *negative variables* are created as a counterpart for each metric variable. Within the LNE formulæ, any terms with negative coefficients are rewritten as positive terms using the negative variable counterparts. During heuristic computation, upper bounds on the values of variables are maintained at each fact layer (with an upper-bound on $-x$ corresponding to the negation of a lower bound on x). These optimistic bounds can be used to determine action applicability through substitution into numeric precondition formulæ at each fact layer. Actions are considered applicable if all their propositional and numeric preconditions hold.

3 The Metric RPG Heuristic

The metric RPG heuristic provides good search guidance in a range of domains and Metric-FF has performed well in several international planning competitions, either directly, or as the sub-solver within SGPlan (Chen, Wah, & Hsu 2006). However, it has two interesting weaknesses which we now observe: *helpful action distortion* and *cyclical resource transfer*. To illustrate these we will use the Settlers domain (described below), which has complex numeric properties and is representative of a larger class of interesting numeric problems featuring resource production and exchange.

The metric RPG heuristic provides two sources of search guidance: a heuristic goal-distance estimate and helpful actions. The helpful actions filter is a powerful way to restrict search, by considering only those actions in the relaxed plan that appear in the first layer of the RPG (along with any other actions that achieve the same effects). The numeric structure of problems interacts with the metric RPG to harmfully affect both of these techniques. For this analysis, we consider only the subset of LNEs in which numeric variables are increased by, decreased by or assigned a constant value. This restriction still leaves a very expressive subset of the language (Helmert 2002).

Action	Timber	Stone	Ore	Wood	Coal	Iron
Move cart						
Move train					-1	
Move ship					-2	
Fell timber	+1					
Quarry stone		+1				
Mine ore			+1			
Saw wood	-1			+1		
Make coal	-1				+1	
Smelt iron			-1		-2	+1
Build cabin						
Build quarry						
Build mine				-2		
Build saw-mill	-2					
Build iron-works		-2		-2		
Build coal-stack	-1					
Build dock		-2		-2		
Build wharf		-2				-2
Build house		-1		-1		
Build cart	-1					
Build train						-2
Build ship						-4
Build rail				-1		-1

Table 1: Production/Consumption in the Settlers Domain

3.1 The Settlers Domain

The Settlers domain (Long & Fox 2003b) is a complex numeric planning domain, in which the aim of the problem is to build up an infrastructure of buildings and transportation.
1) There are six numeric resources: shown with the production/consumption effects of actions in Table 1[1].
2) Not all resources are directly producible. Timber, Stone, and Ore can all be produced directly, in particular locations, but Wood, Coal and Iron require *refinement* of one resource into another.
3) Transferrable Resources: resources can be moved between vehicles and locations using load/unload actions. Functions are used to track the quantities of each resource at each location and in each vehicle. Vehicles have finite carrying capacity. No resource is produced or consumed during loading and unloading: it is only moved. However, the model requires that the resource level in the source is increased and the level in the destination is decreased.

3.2 Definitions

Numeric planning domains exhibit common patterns. In particular, two kinds of actions frequently appear: resource producers, which generate quantities of a metric resource and consumers that can only be executed when there is sufficient resource available and eliminate the resource as a result. We restrict our attention to the simplest forms of these archetypes (we assume that the domain has already been normalised into LNEs and counterpart negative variables as described above):

Definition 3.1 — A Producer
A ground action a is a producer with respect to a given numeric variable v iff it has an effect $v = v + c$ (where $c > 0$) and has no precondition $v \geq min$.

This producer is not limited by the availability of the resource it is intended to produce. Producers that require bootstrapping obviously exist, but we do not consider them here.

[1]The table shows data for a debugged version of the domain, which will be linked from this paper.

However, we do consider actions that have logical preconditions, such as an action for recharging a solar powered vehicle that requires the vehicle to be in the sun.

A consumer is defined as follows:

Definition 3.2 — A Consumer
A ground action a is a consumer with respect to a given numeric variable v if it has either of:
- a precondition $v \geq c$ and an effect $v = v - c$, with $c \in \mathbb{R}^+$
- a precondition $v > 0$ and an effect $v = v - 1$, and $v \in \mathbb{N}$

Note that PDDL2.1 has no explicit distinction between integer and non-integer variables, but this can be determined in some cases by domain analysis.

If all the actions that decrease a variable v are consumers then we can assert that, if $v \geq 0$ in the initial state, the lower bound on v is 0: no action in the plan can reduce v by more than its current value. Similarly, if all actions increasing v are producers and they all have a precondition $v \leq ub$ then the upper bound on v is $max[ub + c]$ across all producer actions for v. For example, in Settlers, the amount of a resource available at a location can never be less than 0 — there can be no deficit spending of resources.

3.3 Helpful Action Distortion

Although the metric RPG can provide good heuristic guidance, a problem that undermines its effectiveness is that of Resource Persistence (ReP). This problem emerges in logical relaxations, where a proposition that should be deleted by one action is then repeatedly reused by many other actions without observing the costs of reinstating the proposition for each use. Ignoring negative metric effects (as discussed in Section 2.1) leads to a very similar effect: the negative effects of consumers are ignored. In numeric problems, consumers are very common and important — indeed, the purpose of using metric resources is often to manage the interactions between producers and consumers. The problem created by ReP is that if sufficient resource is available for the highest single consumer (the precondition is not ignored in the RPG) then the relaxation allows arbitrarily many consumer actions to be executed without the need for any further producers to be added to the plan. For example in the Settlers domain, once 2 units of each resource have been produced, then no relaxed plan will require the production of any further resources (see Figure 1). Sapa (Do & Kambhampati 2001) attempts to compensate for this by adding a resource production penalty to relaxed plans.

ReP is responsible for creating dead-ends by allowing consumption of resource that cannot be replaced because preconditions of producers cannot be achieved without the expenditure of further (now unavailable) resource. This problem is further confounded by the phenomenon of **Helpful Action Distortion** (HAD). This occurs when ReP causes the relaxed plan to contain no producers even though insufficient resource is available to supply all the consumers in the plan. The consequence is that the helpful actions will not include producers, even if the current state is one in which producers can be conveniently applied. For example, in a Settlers problem that requires wood to be transported between locations, a relaxed plan can use the wood to construct a cart and then load *the same* wood onto the cart to transport it. The planner will not consider producing more wood.

Observing the behaviour of Metric-FF in the Settlers domain, one can see that, first, Metric-FF attempts to find a solution plan using Enforced Hill Climbing (EHC) search, committing to the actions leading to a state S if $h(S)$ is the best seen so far. ReP can lead to dead-ends forcing Metric-FF to resort to exhaustive search. Secondly, the effect of HAD on EHC search in low-resource situations, is to cause exhaustive breadth-first search over the search space forwards from the current local minimum, considering only helpful actions, followed, when the reachable space is exhausted, by a full exhaustive search with helpful action pruning disabled. The producers excluded by HAD will then be considered, but the fruitless breadth-first search step is a considerable overhead: there is a direct correlation between problems in which it occurs and increased planning time.

3.4 Cyclical Resource Transfer

Application of the Metric RPG heuristic in the Settlers domain reveals a second problem phenomenon: **Cyclical Resource Transfer** (CRT). Consider a state in which 1 timber and a cart are at a location, $p1$, and the goal is to have 2 timber at $p1$. Clearly, the solution plan must involve the production of more timber. However, a valid relaxed plan solution, found using the metric RPG heuristic, is:

```
0:  (load v1 p1 timber)
1:  (unload v1 p1 timber)
```

The first action increases the timber on the cart $v1$, and the second exploits this to unload timber from $v1$ back to the location, resulting in 2 timber being at the location. This Cyclical Resource Transfer is the direct consequence of ReP, but with a dangerous twist. Instead of simply ignoring the need to produce a resource that will be required, CRT misleads the planner into supposing that resource can be produced by a completely artificial artifact of the relaxation. Even though no timber production actions are present in the plan, as denoted by Table 1, being able to cyclically shunt resources creates the illusion of production: load 'produces' resource in a cart (subject to there being some at the location), and unload similarly 'produces' resource in a location.

4 A Hybrid RPG–LP Heuristic

We now propose a way to address HAD and CRT in numeric domains by the construction of a linear program alongside the RPG. In contrast to previous work employing LP heuristics (van den Briel *et al.* 2007), where the entire problem is encoded as an LP we use the propositional RPG to manage the propositional aspect of heuristic evaluation. In this sense, our approach is similar to LPSAT (Wolfman & Weld 2000), IP-SAT (Kautz & Walser 2000) and TM-LPSAT (Shin & Davis 2005), using a hybrid of LP and a separate combinatorial solver (in all of these examples, it is a SAT-solver).

4.1 LP Construction

When faced with the problem of evaluating a state, S, the following is known:

- Fixed fluent values at fact layer 0, $v_0(0)..v_0(n)$, denoting the values of the numeric variables in S;
- The actions applicable in S, $a_0(0)..a_0(m_0)$, which form action layer 0 in the RPG;
- The producer/consumer behaviour of each action with respect to the fluent values.

The approach to building the RPG taken in Metric-FF is to perform layer-wise expansion of the planning graph, estimating upper- and lower-bounds of numeric variables, and using these, along with the propositional information, to determine in which layer each action first appears. As we have seen, the upper- and lower-bound estimation is compromised by Cyclical Resource Transfer.

If all of the numeric behaviour in a domain falls under Definitions 3.1 and 3.2 then a linear program is an appropriate model. Otherwise, the part that is linear can be managed in a linear program and the remainder by the existing numeric bounds estimation techniques used in the metric-RPG heuristic. A common non-linear effect is assignment, which we return to discuss later.

There is a natural mapping from the RPG to a Mixed-Integer Program (MIP), as follows. The variables are formed from two collections:

- Fact layer variables: for each fact layer l, we have variables $v_l(0)..v_l(n)$, corresponding to the values held in that layer by each of the numeric variables $v(0)..v(n)$. Propositional facts are not represented.
- Action layer variables: for the actions in layer l (following fact layer l) we have variables $a_l(0)..a_l(m_l)$ denoting whether they have been applied. These are binary variables, taking a value of either 0 or 1.

As a special case, the upper and lower bounds of variables $v_0(0)..v_0(n)$ are constrained to the value the variable holds in the state being evaluated, S. The bounds of the remaining fact layer variables are set using the analysis discussed in Section 3.2. In the Settlers domain, this establishes a lower-bound 0 on each fact layer variable.

For each variable $v(i)$, we define a constant-valued function $change(a(j), v(i))$ denoting the effect of $a(j)$ upon $v(i)$:

$$change(a(j), v(i)) = \begin{cases} c & \text{where } a(j) \text{ produces } c \text{ of } v(i) \\ -c & \text{where } a(j) \text{ consumes } c \text{ of } v(i) \\ 0 & \text{otherwise} \end{cases}$$

We can fully specify the constraint restricting the value of each variable $v_{l+1}(i)$ as:

$$v_{l+1}(i) - v_l(i) - \sum_{j=0..m_l} change(a(j), v(i)).a_l(j) = 0$$

The result is a MIP representing the (linear) numeric structure of planning graph and this can be further relaxed to a LP by allowing the binary action layer variables to be relaxed into real-valued variables in the range $[0, 1]$. The resulting machinery is a powerful tool for reasoning with complex numeric behaviours in planning, with many potential uses. For instance, it could serve as an adjunct to GraphPlan (Blum & Furst 1995) to extend search to numeric problems. We will use it in two ways: to determine action applicability during the RPG expansion phase and to determine how to satisfy numeric preconditions during the RPG solution extraction phase.

4.2 LP Usage: Graph Expansion

The use of the LP during relaxed planning graph expansion is non-trivial. The trade-off between informativeness and cost is critical: each call to the LP solver carries a cost. One possibility is to build the RPG without considering the LP, and then use the LP during plan extraction, similarly to previous work in partial satisfaction planning (Benton *et al.* 2007). However, we use the LP to determine upper- and lower-bounds on variable values at each fact layer during RPG construction. We call the LP twice for each numeric variable in each fact layer: once with the objective to maximise the variable, giving an upper bound and once to minimise the variable, giving a lower bound. With these bounds it is possible to determine which numeric preconditions can be satisfied in the construction of the subsequent action layer. In this way, the LP replaces bounds propagation in the metric RPG and the LP bounds are used in determining action applicability during RPG construction.

4.3 LP Usage: Solution Extraction

Having constrained the construction of the RPG through the use of the LP, the layers at which actions are first applicable are changed (with respect to the Metric RPG), and the earliest achievers for facts (numeric or propositional) are similarly changed. These changes serve to provide more accurate heuristic estimates and more informative action selection during plan extraction. However, the use of the LP also requires changes to the metric relaxed plan extraction. This is a direct consequence of the relaxation of the MIP into the LP, which means that the achievement of particular bounds in the RPG might be based on non-integer assignments to action variables, which must be interpreted as "partial" applications of actions.

The revised relaxed plan extraction algorithm is presented in Algorithm 1. It begins by defining a priority queue of goal layers, ordered deepest first. Each goal layer contains two maps: *prop*, mapping propositional facts onto values in the range $[0, 1]$ and *num*, mapping numeric preconditions onto values into the range $[0, 1]$. There are then two key differences between this algorithm and the original relaxed plan extraction algorithm used in Metric-FF.

First, propositional facts and numeric preconditions are weighted, and these weights are used in determining the heuristic value. To understand why, consider an example in which the LP proposed to achieve a resource availability bound of 1 with an assignment of 0.2 to each of 5 action variables. If we round up these action variables into integer 0-1 values then each will take the value 1 and they contribute 5 to the heuristic value. Instead, the plan uses real-valued weights for facts and actions to reflect the degrees of contributions to the achievement of metric goals. At lines 3–6, the goals are each assigned a weight of 1. While extracting the relaxed plan, we keep the maximum weight seen for each fact and numeric precondition at the appropriate layer (lines 20 and 39) prior to eventually choosing its achiever.

Second, to achieve a numeric precondition at a given layer, the LP is used (line 31). We cannot add numeric preconditions directly to the LP as constraints over the relevant fact layer variables, as they might not be satisfiable,

Algorithm 1: Relaxed Plan Extraction

```
   Data: R - a metric RPG; PG - propositional goals;
         NG - numeric goals
   Result: ha - helpful actions, h - A heuristic value
1  ha ← ∅, h ← 0;
2  q ← deepest-first priority queue of goal layers;
3  foreach p ∈ PG do
4      l ← layer at which p first appears;
5      insert (p, 1) into q[l].prop;
6  foreach f ∈ NG do
7      l ← layer at which f first holds;
8      insert (f, 1) into q[l].num;
9  while q not empty do
10     (l, ⟨prop, num⟩) ← pop(q);
11     foreach (p, w) ∈ prop do
12         h ← h + w;
13         a ← achiever for p;
14         if a in action layer 0 then add a to ha;
15         prop ← prop \ add effects of a;
16         foreach propositional precondition pre of a do
17             l ← layer at which pre first appears;
18             if l > 0 then
19                 if ∃(pre, k) ∈ q[l].prop then
20                     if k < w then k ← w;
21                 else insert (pre, w) into q[l].prop;
22         ... similarly for numeric preconditions of a...;
23     foreach non-linear (f, w) ∈ num do
24         ... as in Metric-FF, modified for weights ...;
25     foreach linear (f, w) ∈ num do
26         foreach variable v used by f do
27             LP' ← LP;
28             if v is a positive metric RPG variable then
29                 LP' = LP' + {v = max(v_l)};
30             else LP' = LP' + {v = min(v_l)};
31             solve LP', minimising weighted action sum;
32             h ← h + w. LP' objective function value;
33             av ← {action variable (a = c) ∈ LP' | c ≠ 0};
34             foreach a ∈ av do
35                 if a is in layer-zero then add a to ha;
36                 foreach propositional precondition pre of a
                   do
37                     l ← layer at which pre first appears;
38                     if ∃(pre, k) ∈ q[l].prop then
39                         if k < w.c then k ← w.c;
40                     else insert (pre, w.c) into q[l].prop;
```

and we do not wish to introduce backtracking into RPG solution extraction. Instead, we consider individual variable bounds separately during graph expansion. Therefore, during regression, we consider the individual variables separately within numeric preconditions. For each variable in a precondition, we use the LP to find the actions that contribute to the variable meeting its bound at the appropriate layer in the graph. For positive variables, we use the upper bound, and for negative variables, the lower bound. In this LP we minimise a weighted sum over the action variables, where the weight attached to an action is 1.1^n, where n is the layer at which the action first appears in the RPG. This weighting causes the LP to prefer earlier actions, which is analogous to a similar preference used in Metric-FF. The LP is guaranteed to be solvable because we are using proven min/max bounds. For each action variable that holds a non-zero value in the LP solution (line 33) we add the relevant propositional preconditions to earlier goal layers with an appropriate weight (line 39).

We will illustrate the impact of this approach on the CRT problem in the Settlers example considered earlier: 1 timber and a cart are at $p1$ and the goal is to have 2 timber at that location. In the Metric RPG, 2 timber become available at $p1$ in fact layer 2, after the application of a *load* followed by an *unload*. In the hybrid LP-RPG this is no longer a way to achieve the fact, as the consumption of timber by the *load* action at $p1$ prevents it. If we suppose that there is an adjacent location, $p2$, with a cabin (to allow timber to be produced) then one possible relaxed plan is:

```
0:  (fell-timber p2)
0:  (move-cart v1 p1 p2)
1:  (load v1 p2 timber)
2:  (unload v1 p1 timber)
```

The action to move the cart from $p2$ back to $p1$ does not appear because of the relaxation of delete effects, but the heuristic is clearly more informed than the Metric RPG heuristic. The relaxed plan length is 4, not 2 — closer to the true plan length (5). We have also averted Helpful Action Distortion: felling timber and moving the cart are considered helpful, rather than the irrelevant step of loading timber onto the cart.

5 Heuristic Reformulation

Having now described the LP heuristic in a form which maps naturally onto the RPG, we describe a series of reformulations, some in terms of LP usage and others in terms of LP structure, that serve to substantially reduce the computational overheads incurred through the use of the LP.

5.1 Static Analysis: Instrumentation Variables

During expansion of the metric RPG, upper- and lower-bounds are calculated on *all* variables and then used to determine whether preconditions are satisfied. However, not all variables are relevant to the satisfaction of numeric preconditions. In many numeric planning problems, there are variables whose sole purpose is in measuring the quality of the solution found (playing roles analogous to *total-time* for temporal problems). These *instrumentation variables* are defined as follows:

Definition 5.1 — Instrumentation Variable
An instrumentation variable v is one which:

- is assigned a value in the initial state;
- never appears in the preconditions of any actions;
- is modified by an effect of at least one action.

The Settlers domain, like many others, has variables of this type, whose values are combined in a weighted sum to give a plan quality metric. In Settlers the variables are Labour, Pollution and Resources. Recognising these variables has two interesting consequences. If we are only concerned with satisficing planning, we need not include action variables in the LP for actions whose sole numeric effect is

to 'produce' instrumentation quantities. When plan quality is important, the instrumentation variables can be used with the plan quality metric to influence action choice during relaxed solution extraction. We leave this interesting option for future work.

5.2 Action Occurrence Bounds

Making implicit variable bounds explicit in the encoding of an LP is a powerful way to improve performance. A bound on the number of occurrences of an action can be added to the LP as an additional constraint limiting the sum of the variables representing that action across all layers. We define occurrence bounds on actions in three ways:

Definition 5.2 — One-Shot Proposition Consumption
A ground action a is one-shot if:

- a has a propositional precondition p, initially satisfied;
- a has p as a delete effect;
- no action has p as an add effect.

A one-shot action occur at most once and only if p is true in the state being evaluated.

Definition 5.3 — Non-Beneficial Action Repetition
It is not beneficial to repeat a ground action a if:

- a has propositional add effects P;
- no action deletes any $p \in P$;
- the numeric effects of a are undesirable: for each variable v decreased by a, no condition $v \leq c$ exists and, for each variable v increased by a, no condition $v \geq c$ exists.

Such actions should occur at most once and only if their effects are required.

Definition 5.4 — Bounded k-Shot Action
A ground action a is k-shot if:

- it has a numeric precondition $v - p \geq c$, with $p, c \in \mathbb{R}_0^+$
- it has an effect $v = v - c$;
- no other action increases v.

Bounded k-shot actions can occur at most $k = \lfloor (v - p)/c \rfloor$ times.

One-shot actions appear in the Settlers domain, within the vehicle-building actions: each deletes a proposition, *(potential ?v)*, true in the initial state and not added by any other action. Actions that it is not beneficial to repeat include the actions for building infrastructure: for instance, once *(build-mine l)* has been applied, nothing deletes the proposition *(has-mine l)*, and applying it again is worthless. Bounded k-shot actions are a generalisation of one-shot actions that can be easily exploited by limiting the sum of the appropriate action variables in the LP to k.

As an extension of the idea of one-shot actions, we can define *sets* of actions, amongst which only one can ever be chosen in the LP. We define one-shot action sets as follows:

Definition 5.5 — One-Shot Action Set
A one-shot action set Ω is a set of one-shot proposition consumption actions, that each irreversibly consume a common proposition, p.

Give an one-shot action set,Ω, we can add a constraint to the LP that the sum of the action variables corresponding to the actions in Ω, across all layers, is ≤ 1.

5.3 Linear Assignment Analysis

As we observe in Section 4.1, assignment effects are, generally, non-linear. In some cases, however, assignment is linear. First, we define *conditioned variables* as follows:

Definition 5.6 — Conditioned Variables
A variable v is conditioned by a set of mutually-exclusive propositions, P, if all actions with numeric preconditions or effects involving v also require one of the propositions $p \in P$ to hold. Hence, until one of P holds, v is logically disconnected from the problem.

Combining Definitions 5.5 and 5.6 we can infer the following: if all the actions adding one of the facts P are in a one-shot action set Ω, and only actions in Ω assign values to v, then assignment to v is reducible to a linear form. For each action $a \in \Omega$, we can replace the assignment effect $v = k$ with an effect $v = v + k$, and initialise $v = 0$. This transformation is safe, since only one assignment to v can ever occur within any solution to the problem, and through Definition 5.6, v is logically disconnected until such an assignment has been made.

In the Settlers domain, assignment occurs only when a vehicle *?v* is produced: the variable *(space-in ?v)* is assigned the appropriate value, and the variables denoting the levels of each of the six resources in the vehicle is initialised. Through linear assignment analysis, we can show this assignment is in fact linear:

- The actions producing a vehicle v form a one shot-group Ω: they all irreversibly delete the fact *(potential v)*.
- Each of the actions in Ω also adds a fact *(is-at v ?p)*, where *?p* is the location in which the vehicle was made. Through static analysis, it is trivial to show all propositions of the form *(is-at v ?p)* are mutually-exclusive, forming a set P.
- All actions refering to or modifying the assigned-to variables have a precondition $p \in P$.

Although this transformation appears quite specific in its behaviour, the pattern is quite general: the initialisation of a variable according to specific conditions.

5.4 Selective Integer Variables

We have described how to construct the constraints as a MIP, then relax it to a LP to reduce solution costs. It is possible, to *selectively* allow some variables to remain as integers, at the risk of increased solution cost. In particular, one can allow the first action layer variables to remain as integers. This has two benefits: we have fewer but equally informative helpful actions (for example, rather than 2 similar actions, being chosen 0.5 times each, just one is chosen) and fewer actions are used *throughout the LP*, reducing the cost of extracting a relaxed plan from the RPG. The integer variables, with an optimisation criterion to minimise their sum, force the LP to decide *what* to do in the first action layer, typically reducing the number of objects participating in the rest of the plan. Our later evaluation shows that this approach improves planner performance despite increasing the complexity class of the heuristic.

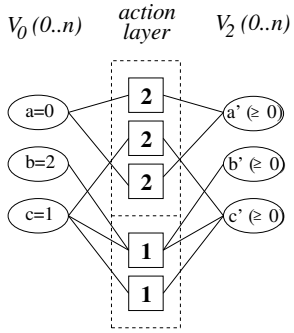

Figure 1: Adding Action Layer 2 to a Compressed LP

5.5 Layer Compression

We have presented the LP as a mapping of the planning graph to action and numeric-value variables. Whilst conceptually clean, the structure has a key weakness. Each time a new action layer extends the RPG, all actions present in the previous layer are present in the new layer. These repeated action variables lead to a rapid growth in the LP.

We avoid this problem by further relaxing the LP using *layer compression*. Rather than replicating action layers during expansion, we augment a single action layer. An RPG action layer contains two sorts of actions: actions new to that layer and actions from previous layers. If an action is new to that layer, we create a new variable for it in the LP, in the range $[0, 1]$, and update the constraints for the subsequent fact layer variables to represent the numeric behaviour of the action. For existing actions, we can simply increment the upper bound for the variable range by 1, thereby allowing the existing action to be applied an additional time if appropriate, just as would be the case if an extra $[0, 1]$ action selection variable were added. The resulting compressed representation is depicted in Figure 1. The existing action layer (layer 1) contains three actions, and two new actions are applicable in action layer 2. Hence, the updated LP contains five action-selection variables: three which have now been present for two layers and are therefore given ranges $[0, 2]$ (labelled as '2'); and two which are new, and hence are bounded $[0, 1]$ (labelled as '1').

The resulting structure is similar to the fact and action spikes used in STAN (Long & Fox 1999), compressing the LP into a space- and computationally-efficient format. The compression loses something in terms of inferential power, but significantly reduces the size of the LP and remains more informed than the metric RPG heuristic. Action variables are weighted in the relaxed plan extraction LP according to the first layer in which they appear. The actions that first appear in the first layer are selected as integer variables.

6 Evaluation

To evaluate the LP-RPG heuristic we compare its use with the metric RPG heuristic. For the metric RPG heuristic we use Hoffmann's Metric-FF, while we have implemented the LP-RPG heuristic in a Metric-FF analogue, using the same search algorithms (enforced hill climbing with helpful actions, followed by WA* if this fails). For comparison with different approaches to managing numbers we also compare with LPG-td (Gerevini, Saetti, & Serina 2004) and Sapa (Do & Kambhampati 2001). The tests use Linux machines with 3.4GHz Pentium D CPUs and set limits of 1.5GB of memory and 30 minutes.

Problem	LP-RPG	Metric-FF	*No Ints*	*Uniform*	LPG-td	SGPlan
ipc-1	1.11	4.46	0.94	1.1	0.1	-
ipc-2	0.24	0.01	0.22	0.24	0.08	-
ipc-3	10.3	31.96	70.12	150.64	0.27	-
ipc-4	9.96	78.58	2.02	4.27	0.15	-
ipc-5	10.9	0.52	7.25	36.43	3.08	-
ipc-6	25.3	9.01	17.64	6.33	2.26	-
ipc-7	-	-	-	-	3.76	-
ipc-9	-	-	-	-	8.8	-
ipc-10	517.5	-	-	1267.4	7.07	-
ipc-11	1123	-	-	-	6.54	-
ipc-12	268.0	-	-	-	4.69	-
ipc-13	-	-	-	-	16.14	-
ipc-14	-	-	-	-	33.22	-
ipc-15	-	-	-	-	80.48	-
ipc-16	-	-	-	-	126.61	-
ipc-17	-	-	-	-	18.95	-
2xhouse overseas	0.81	-	0.71	0.80	-	-
2xtimber overseas	0.67	-	0.57	9.02	-	-

Table 2: Results for the Settlers Domain (time in seconds)

The LP-RPG heuristic is designed to be more informed in domains in which there is interaction between different metric variables. A review of existing benchmarks reveals that there are currently very few, possibly because current planners are not very good at handling this kind of problem. Settlers is the most interesting example. Another domain that offers this kind of interaction is the MPrime domain introduced in IPC1 (McDermott 2000), which is a logistics problem with limited fuel and vehicle capacity, and with fuel supplies being transferrable between locations. In this test we encoded the domain using numbers instead of symbolic constants to represent fuel levels and capacities. As a further interesting example of a domain with interacting numeric variables, we also developed the Trader Domain: 20 problems were created, with between 2 and 5 markets, each buying and selling 16 commodities at a range of (different) prices. The domain includes a cost for travelling between markets[2]. The goal in each case is to have 1000 dirham, having started with 100 dirham and the solutions involve buying commodities cheaply and travelling to markets where they can be sold for a profit. In domains without interactions between different metric variables the LP-RPG offers no benefits over the metric RPG and simply adds overhead. It is straightforward to recognise such domains and turn off the machinery automatically, so we present no results for them.

Results for Settlers are shown in Table 2. In the IPC problem set, the LP-RPG heuristic provides better guidance than the RPG heuristic. The two additional problems involve exploiting a pre-built ship to send goods overseas. In one case the task is to build 2 houses overseas and in the other to have 2 timber overseas. Despite appearing simple, we have yet to find any other planner that can solve either of these problems. The table includes results labelled 'No Ints', which show the performance without using integers for the first action layer variables 5.4, and 'Uniform', which show performance with uniform action weights, rather than 1.1^n, in the LP during solution extraction. The results indicate the extent that both of these help. Nevertheless, the underlying LP-RPG approach is still sufficient to out-perform the metric RPG. LPG-td achieves a superior performance in this domain, using a technique that is also based on propagation

[2] `http://cis.strath.ac.uk/~ac/trader.pddl`.

Problem	LP-RPG	Metric-FF	Sapa	LPG-td	SGPlan
ipc-1	0.01	0.02	0.074	-	-
ipc-2	0.04	0.04	18.447	-	-
ipc-3	0.02	0.02	1.161	-	-
ipc-4	0.02	0.02	133.938	-	-
ipc-5	0.04	0.04	97.801	-	-
ipc-6	3.04	123.06	-	-	-
ipc-7	0.03	0.03	1.095	-	-
ipc-8	1.32	0.11	-	-	-
ipc-9	0.03	0.02	5.867	-	-
ipc-10	3.62	25.83	-	-	-
ipc-11	0.02	0.02	0.162	-	-
ipc-12	0.38	0.04	-	-	-
ipc-13	1.64	75.86	-	-	-
ipc-14	3.30	69.76	-	-	-
ipc-15	0.13	0.07	-	-	-
ipc-16	0.04	0.04	6.700	-	-
ipc-17	0.13	0.09	128.585	-	-
ipc-18	7.46	-	-	-	-
ipc-19	0.30	0.08	-	-	-
ipc-20	0.16	0.06	-	-	-
ipc-21	0.16	0.14	-	-	-
ipc-22	130.50	-	-	-	-
ipc-23	2.27	0.26	-	-	-
ipc-24	0.20	0.14	-	-	-
ipc-25	0.01	0.02	0.061	-	-
ipc-26	0.10	0.03	34.392	-	-
ipc-27	0.01	0.02	0.264	-	-
ipc-28	0.01	0.02	129.511	-	-
ipc-29	0.02	0.02	0.222	-	-
ipc-30	0.07	0.03	149.135	-	-

Table 3: Results for the MPrime Domain (time in seconds)

Problem	LP-RPG	Metric-FF	Sapa	LPG-td	SGPlan
trader-1	0.82	-	-	-	-
trader-2	1.39	-	-	-	-
trader-3	1.47	-	-	-	-
trader-4	1.35	-	-	-	-
trader-5	1.52	-	-	-	-
trader-6	1.94	-	-	-	-
trader-7	15.20	-	-	-	-
trader-8	1.50	-	-	-	-
trader-9	0.88	-	-	-	-
trader-10	14.50	-	-	-	-
trader-11	0.78	-	-	-	-
trader-12	37.41	-	-	-	-
trader-13	2.25	-	-	-	-
trader-14	7.15	-	-	-	-
trader-15	2.17	-	-	-	-
trader-16	4.69	-	-	-	-
trader-17	4.78	-	-	-	-
trader-18	2.52	-	-	-	-
trader-19	1.53	-	-	-	-
trader-20	3.20	-	-	-	-

Table 4: Results for the Trader Domain (time in seconds)

of bounds through a reachability graph, but relying on local search to extract a plan. Sapa cannot solve these problems.

The metric RPG heuristic applied to the numeric encoding of MPrime gains nothing from the numeric encoding and reduces to the propositional RPG guidance. However, in the LP-RPG heuristic, considering resources in a numeric form helps. Table 3 shows run-times of Metric-FF, the LP-RPG planner and Sapa for 30 problems. LPG-td cannot solve any of these problems. The results show that LP-RPG solves all 30 problems and that, for harder problems, where the overheads of constructing the LP do not dominate, it finds a plan an order of magnitude faster than its rivals.

Table 4, shows results for the Trader domain. No other planner can solve these problems. LP-RPG solves all 20 problems with plans ranging in lengths between 54 and 252 steps. The solutions typically have two phases: build up collateral with low-cost-small-margin goods, before switching to high-cost-high-profit goods, travelling longer distances to trade, once sufficient collateral is acquired. The metric RPG builds relaxed plans that involve buying a single expensive item and then selling it back multiple times to achieve the necessary wealth. This is a worthless guide to either helpful actions or goal distance.

7 Conclusions

This paper presents a novel use of linear programming to support a relaxed plan heuristic in forward state-space search planning. We have only begun to explore the possibilities of this structure, but we have already shown that it significantly improves the informedness of the Metric RPG heuristic in problems where numeric variables interact. Although LPG-td has impressive performance in some Settlers problems, its performance appears brittle and other domains that exhibit interesting numeric interactions defeat it. We know of no other planners capable of tackling this range of numeric problems, despite the importance that this kind of numeric resource management has in real planning domains. Our work offers an important step in extending planners to manage more realistic problems.

Acknowledgement

The authors thank Chris Beck for helpful discussions during the development of this work.

References

Benton, J.; van den Briel, M.; and Kambhampati, S. 2007. A Hybrid Linear Programming and Relaxed Plan Heuristic for Partial Satisfaction Planning Problems. In *Proc. Int. Conf. on AI Plan. and Sched. (ICAPS)*.

Blum, A., and Furst, M. 1995. Fast Planning through Planning Graph Analysis. In *Proc. Int. Joint Conf. on AI (IJCAI-95)*.

Chen, Y.; Wah, B. W.; and Hsu, C. 2006. Temporal Planning using Subgoal Partitioning and Resolution in SGPlan. *J. of Art. Int. Res.* 26:323–369.

Do, M. B., and Kambhampati, S. 2001. Sapa: a domain-independent heuristic metric temporal planner. In *Proc. European Conf. on Planning (ECP'01)*.

Edelkamp, S. 2003. Taming numbers and durations in the model checking integrated planning system. *J. Art. Int. Res.* 20:195–238.

Fox, M., and Long, D. 2003. PDDL2.1: An Extension of PDDL for Expressing Temporal Planning Domains. *J. Art. Int. Res.* 20:61–124.

Gerevini, A.; Saetti, A.; and Serina, I. 2004. Planning with Numerical Expressions in LPG. In *Proc. 16th European Conf. on AI (ECAI'04)*, 667–671.

Helmert, M. 2002. Decidability and undecidability results for planning with numerical state variables. In *Proc. AI Plan. and Sched. (AIPS)*.

Hoffmann, J. 2003. The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables. *J. of Art. Int. Res.* 20:291–341.

Kautz, H., and Walser, J. 2000. Integer optimization models of AI planning problems. *Knowledge Eng. Rev.* 15(1):101–117.

Koehler, J. 1998. Planning under resource constraints. In *Proc. European Conf. on AI (ECAI'98)*, 489–493.

Long, D., and Fox, M. 1999. Efficient Implementation of the Plan Graph in STAN. *J. of Art. Int. Res.* 10:87–115.

Long, D., and Fox, M. 2003a. Exploiting a Graphplan Framework in Temporal Planning. In *Proc. Int. Conf. on AI Plan. and Sched. (ICAPS)*.

Long, D., and Fox, M. 2003b. The 3rd International Planning Competition: Results and Analysis. *J. of Art. Int. Res.* 20:1–59.

McDermott, D. 2000. The 1998 AI planning systems competition. *AI Magazine* 21(2):35–55.

Shin, J.-A., and Davis, E. 2005. Processes and Continuous Change in a SAT-based Planner. *J. Art. Int.* 166:194–253.

van den Briel, M.; Benton, J.; Kambhampati, S.; and Vossen, T. 2007. An lp-based heuristic for optimal planning. In *Principles and Practice of Constraint Programming (CP 2007)*, 651–665.

Wolfman, S., and Weld, D. 2000. Combining linear programming and satisfiability solving for resource planning. *Knowledge Eng. Rev.* 15(1).