

# A Preliminary Study on the Relaxation of Numeric Features in Planning

Antonio Garrido<sup>a</sup>, Eva Onaindía<sup>a</sup> and Donato Hernández<sup>b</sup>

<sup>a</sup> *Universidad Politécnica de Valencia (Spain); {agarridot,onaindia}@dsic.upv.es*

<sup>b</sup> *Universidad de Guanajuato (Mexico); donato@salamanca.ugto.mx*

**Abstract.** Modern AI planners use different strategies to simplify the complexity of current planning problems and turn them more affordable. In this paper, we present a new approach that divides the planning search into two consecutive stages. First, a sequential plan is generated by relaxing the numeric features of actions, thus ignoring the scheduling part of the problem. Second, this plan is parallelised while the satisfaction of numeric constraints is guaranteed, thus including the scheduling part of the problem. In our experiments, we have studied/identified classes of problems where this division is more appropriate and effective *w.r.t.* the plan quality.

**Keywords.** Planning, Scheduling, Multiobjective planning, Resource optimisation

## 1. Introduction

Recent advances in AI planning research have significantly improved the performance of planning algorithms, thus allowing them to deal with more realistic capabilities, such as a richer action model that considers time, resources, numeric features and multiobjective optimisation criteria [1,2,5]. In consequence, current planning models can better reflect real application scenarios, such as domains for logistic problems, ground traffic on airports, space applications, etc. However, as the problems become harder and more complex, the difficulties to solve them become harder as well, and planners need to come up with new strategies and techniques to enhance their performance. Modern planners (see IPC-2004<sup>1</sup>) use techniques that simplify the original problem to reduce its complexity in different ways: i) separating the planning and scheduling parts of the original problem; ii) relaxing the problem constraints and using heuristics; iii) performing preprocessing techniques to speed up the planning process; iv) decomposing and reducing the original problem into subproblems that are solved independently; etc.

This paper presents a preliminary study on a new kind of relaxation based on the ideas expressed in [3] that uses some of the previous techniques to deal with numeric variables (including time) in multiobjective planning. **First**, it disjoins the planning part from the scheduling one by separating the structural (propositional) part of the plan from the part dedicated to resource usage (numeric variables and resource management). **Second**, the planning algorithm is divided into two stages: i) generation of a sequential plan

---

<sup>1</sup>More information about the International Planning Competition (IPC-2004) and the domains/problems used in the paper in <http://ipc.icaps-conference.org>

```

(:durative-action fly
:parameters (?a - aircraft ?c1 ?c2 - city)
:duration (= ?duration (/ (distance ?c1 ?c2) (slow-speed ?a)))
:condition (and (at start (at ?a ?c1))
                (at start (>= (fuel ?a)
                               (* (distance ?c1 ?c2) (slow-burn ?a))))))
:effect (and (at start (not (at ?a ?c1)))
             (at end (at ?a ?c2))
             (at end (increase total-fuel-used
                          (* (distance ?c1 ?c2) (slow-burn ?a))))
             (at end (decrease (fuel ?a)
                              (* (distance ?c1 ?c2) (slow-burn ?a))))))

```

**Figure 1.** A typical action for `fly` of a logistics domain.

without considering numeric conditions/effects; and ii) parallelisation of this plan, including new actions to satisfy the numeric (resource) constraints. The first stage performs a straightforward relaxation. In addition to the classical relaxed version of a planning graph [5], which does not consider negative effects nor mutex relations, we also relax the numeric variables on actions. This makes a clear distinction between sketching the structure of the plan and fulfilling the numeric constraints (conditions and effects of actions). **Third**, no mutex calculus is performed while planning, but it is extracted as a preprocessing stage [3].

## 2. Numeric variables in actions

In this paper, we use durative actions as defined in PDDL2.x [1,2], with three types of conditions (*start*, *invariant* and *end*):  $SCond(a)$ ,  $Inv(a)$  and  $ECond(a)$ . In a numeric approach, each condition can be either a classical propositional condition or a numeric condition that expresses a constraint as a tuple  $\langle f\text{-exp1}, \text{binary-comp}, f\text{-exp2} \rangle$ , where  $f\text{-exp1}$  and  $f\text{-exp2}$  represent functional expressions (which may contain numeric variables), and  $\text{binary-comp} \in \{<, \leq, =, >, \geq\}$  is a binary comparator. Additionally, durative actions have two types of effects (*start* and *end*):  $SEff(a)$  and  $EEff(a)$ . Each effect can be either a positive or negative propositional effect, or a numeric effect as a tuple  $\langle f\text{-head}, \text{assign-op}, f\text{-exp} \rangle$ , where  $f\text{-head}$  represents a numeric variable, and  $\text{assign-op} \in \{:=, +=, -=, *=, /=\}$  is an assignment operator which updates the value of  $f\text{-head}$  according to the functional expression  $f\text{-exp}$ .

Figure 1 shows an example of an action `fly` which requires an aircraft to be at the origin (propositional condition), but also to have enough fuel to perform the flight (numeric condition). On the other hand, `fly` makes the aircraft to be at the destination (propositional effect), increases the variable `total-fuel-used` and decreases the level of fuel of the aircraft (numeric effects). A new variable `total-time`, which represents the makespan of the plan, is always included by default and modified by the field `:duration` present in each action. Note that this allows us to treat the duration as any other numeric variable with **no** special distinction. Moreover, each planning problem can define a multiobjective metric function to assess the quality of the plan, such as  $(+ (* 5 (\text{total-time})) (* 0.01 (\text{total-fuel-used})))$ . The application of this metric allows to find plans where several weighted criteria that play an important role in the plan are also considered.

### 3. The 2-stage planning approach

#### 3.1. Stage 1. Generation of a sequential plan

This stage generates a sequential plan that works with causal constraints but without considering the numeric conditions and effects of actions (see Algorithms 1 and 2). Each action with numeric conditions/effects is relaxed and transformed into an action where numeric features are ignored (only propositional conditions/effects are maintained). For instance, the action `fly` will move the aircraft from one city to another ignoring the requirements of `fuel` and `total-fuel-used`. It must be noted that this transformation is trivial and entails no complexity at all. On the other hand, we generate a sequential plan because it can be calculated more easily than a parallel one. Clearly, neither overlapping nor mutex relations between actions needs to be considered.

The sequential plan is created in a forward way through a plan space search in a structure called *set\_of\_plans* that contains all the plans  $\{\Pi_i\}$  which are sorted by their cost. Each  $\Pi_i$  has two structures: *Seq<sub>i</sub>* that contains the actions that form the sequential plan and determine the current state; and *Relax<sub>i</sub>* that contains a relaxed plan from the current state to the problem goals. A relaxed plan consists of a partially ordered set of actions, with no mutex considerations as in [5], connected by causal links in which all the propositional (sub)goals hold.

```
1: set_of_plans ← empty plan  $\Pi_i$  (Seqi = Relaxi =  $\emptyset$ )
2: while set_of_plans  $\neq \emptyset$  do
3:   extract the lowest cost  $\Pi_i$  from set_of_plans
4:   if  $\Pi_i$  is a solution plan then
5:     exit with success
6:   else
7:     Relaxi ← Relax( $\Pi_i$ )
8:     Costi ← Cost( $\Pi_i$ )
9:     RAi ← Relevant_Actions( $\Pi_i$ ) {relevance analysis}
10:    Generate_Successor_Plans( $\Pi_i$ , RAi)
```

Algorithm 1: Generation of a sequential plan.

Each iteration of Algorithm 1 extracts the lowest cost  $\Pi_i$  from *set\_of\_plans*. *Relax<sub>i</sub>* of  $\Pi_i$  is generated as indicated in [3], using the actions that require less prior actions to be executed and have the best cost *w.r.t.* the problem metric. On the other hand, the plan cost is heuristically calculated as:  $Cost(\Pi_i) = |Seq_i| + \omega |Relax_i|$ , where  $|Seq_i|$  and  $|Relax_i|$  represent the number of actions in each structure, respectively, and  $\omega$  allows to set the heuristic estimation<sup>2</sup>. The algorithm also performs a relevance analysis (step 9) by means of a backward process that only selects the relevant actions *RA<sub>i</sub>* to achieve the problem goals from the current state. This helps reduce the branching factor (actions that are executable) when generating the successor plans of  $\Pi_i$ .

Algorithm 2 performs the real search by exploring the plan  $\Pi_j$  generated after executing each  $a_j$  in *RA<sub>i</sub>*. This algorithm uses a Hill-climbing variation; the strategy is to record into *set\_of\_plans* every successor  $\Pi_j \mid Cost(\Pi_j) \leq Cost(\Pi_i)$ , or the successors with minimal cost if all successors have a cost worse than  $Cost(\Pi_i)$ . Moreover, the

---

<sup>2</sup>Greater values of  $\omega$  may find a solution faster but degrading its quality (we are currently using  $\omega = 3$ ).

```

1: for all  $a_j \in RA_i$  do
2:   if  $a_j$  is executable in the current state of  $Seq_i$  then
3:     construct a new  $\Pi_j$  from  $\Pi_i$ , with  $Seq_j \leftarrow Seq_i \cup \{a_j\}$ 
4:     if current state of  $Seq_j$  is already memoized then
5:       discard  $\Pi_j$ 
6:     else
7:        $Relax_j \leftarrow Relax(\Pi_j)$ 
8:        $Cost_j \leftarrow Cost(\Pi_j)$ 
9:       if  $(Cost_j \leq Cost_i) \vee (Cost_j = \min_{\forall \Pi_k} (Cost(\Pi_k)))$  then
10:         $set\_of\_plans \leftarrow set\_of\_plans \cup \{\Pi_j\}$ 
11:        memoize current state of  $Seq_j$ 

```

Algorithm 2: *Generate\_Successor\_Plans*( $\Pi_i, RA_i$ ).

algorithm memoizes the state of each sequential plan  $Seq_j$  preventing it from falling in future loops.

The output of this stage is either a sequential plan that is *propositionally sound*, or failure. In the former, the plan would be directly executable in a scenario without numeric variables such as a pure-STRIPS planning problem. Unfortunately, there is no completeness guarantee: steps 9–10 of Algorithm 2 do not record all the successor plans. There exists, however, an important **advantage** here. If the problem is propositionally unsolvable, i.e. there are some propositional (sub)goals that cannot be satisfied, this stage outputs that no feasible plan can be found (**before** analysing numeric actions).

### 3.2. Stage 2. Plan parallelisation and satisfaction of numeric constraints

This stage parallelises the actions of the sequential plan, including the necessary actions to satisfy the numeric constraints. Although there exist some algorithms to parallelise plans [6,8], our approach cannot use them in a straightforward way. The critical difference relies on the initial plan to be parallelised; traditional research focuses on generating a parallel plan using an **executable** partial or total plan as a basis. This means that all the (sub)goals hold, and only a better ordering among actions is required. On the contrary, our initial plan might have unsolved (numeric) conditions that must be solved before the plan becomes executable. Consequently, this stage also needs to include a search procedure for these situations.

Now, each  $\Pi_i$  is extended with a new structure, initially empty, called  $Alloc_i$ , which contains the actions that have been allocated in time and determine the current state. It is important to note that once an action is inserted into  $Alloc_i$ , it will never be removed. For simplicity, let us assume that actions in  $Seq_i$  are in the form  $\langle a_i^1, a_i^2, a_i^3 \dots a_i^n \rangle$ .

Algorithm 3 starts with the sequential plan generated by Algorithm 1. Each iteration extracts the lowest cost<sup>3</sup>  $\Pi_i$  from  $set\_of\_plans$ , and tries to allocate the actions of  $Seq_i$  in the earliest time  $T_{a_i^j}$  at which action  $a_i^j$  could start in  $Alloc_i$ . If  $a_i^j$  is executable in  $Alloc_i$  at time  $T_{a_i^j}$ , it is inserted into  $Alloc_i$ . Otherwise, new numeric actions  $b_k$  that make  $a_i^j$  executable are inserted, constructing new plans and inserting them into  $set\_of\_plans$ . Intuitively, *Find\_Execution\_Time*( $a_i^j, Alloc_i$ ) (step 8) returns the earliest start time

<sup>3</sup>Unlike Algorithms 1–2,  $Cost(\Pi_i)$  is now calculated by evaluating the actions in  $Alloc_i$  and  $Relax_i$  in the problem metric.

```

1:  $set\_of\_plans \leftarrow \Pi_i$ , generated by Algorithm 1
2: while  $set\_of\_plans \neq \emptyset$  do
3:   extract the lowest cost  $\Pi_i$  from  $set\_of\_plans$ 
4:   if  $\Pi_i$  is a solution plan then
5:     exit with success
6:   else
7:     for all  $a_i^j \in Seq_i$  not allocated yet do
8:        $T_{a_i^j} \leftarrow Find\_Execution\_Time(a_i^j, Alloc_i)$ 
9:       if  $a_i^j$  is executable in the current state of  $Alloc_i$  at time  $T_{a_i^j}$  then
10:         $Alloc_i \leftarrow Alloc_i \cup \{a_i^j\}$  { $a_i^j$  is inserted at time  $T_{a_i^j}$ }
11:       else
12:        for all numeric action  $b_k$  that makes  $a_i^j$  executable do
13:          construct a new  $\Pi_k$ , with  $Seq_k \leftarrow \{b_k\} \cup Seq_i$ 
14:           $set\_of\_plans \leftarrow set\_of\_plans \cup \{\Pi_k\}$ 
15:        go to step 2

```

Algorithm 3: Plan parallelisation and satisfaction of numeric constraints.

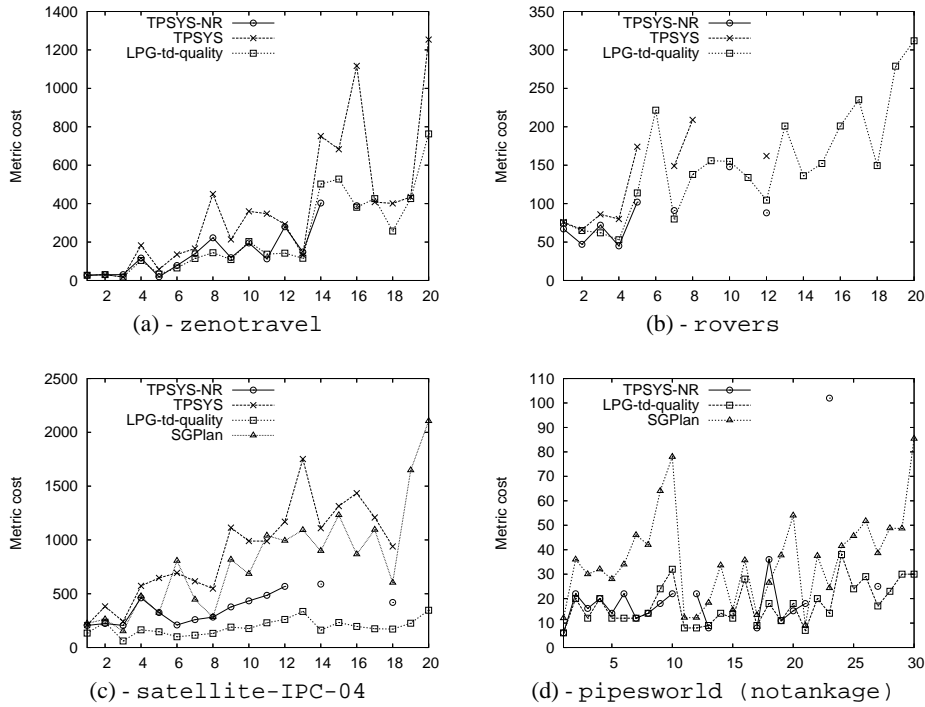
$T$  when action  $a_i^j$  could start and is not mutex in  $Alloc_i$  ( $T = 0$  when  $Alloc_i$  is empty, and  $T$ =end time of the latest  $a_i^k$  that needs to be executed before  $a_i^j$  otherwise).

This stage returns a parallel plan that may contain numeric actions, or failure. The previous algorithms cannot guarantee the completeness and optimality properties. There is an additional drawback: the sequential plan might not be properly parallelised if no numeric action is available to satisfy the numeric conditions. For instance, let us suppose that the available level of fuel for an aircraft is scarce and no `refuel` action is available. If the aircraft runs out of fuel in a journey, the algorithm will not be able to find an executable plan to refuel and eventually, it will return failure. Consequently, it is important to find out in which classes of problems this approach results more appropriate.

#### 4. Experimental results. Analysis of results

In this section, we have used the planner TPSYS presented in [3] and implemented the new 2-stage search with numeric relaxation (from now on TPSYS-NR) on top of it. We have focused on the analysis and validation of the approach rather than in speed and code optimisation. We have run problems from both IPC-2002 and IPC-2004 domains on a Pentium IV 2.6Ghz with 512 Mb., which were censored after 300 seconds.

Our objective is to analyse features in problems that make our approach more useful and effective *w.r.t.* the plan quality in comparison with other state-of-the-art planners. According to the separation used in our approach, the most significant feature is the influence of numeric conditions in the actions. Therefore, we have identified **two** classes of problems: i) problems with a strong dependency on the numeric conditions; and ii) problems with a weak dependency (or no dependency at all) on the numeric conditions. In the former, actions to support the numeric conditions can only be applied in very particular cases (in `rovers`, actions to recharge are very limited). In the latter, there are two possibilities: i) there exist no numeric conditions on the actions (`satellite-IPC-04` and `pipesworld` domains); or ii) numeric conditions exist, but actions to achieve them can be easily added as needed (in `zenotravel`, actions to refuel can be applied in



**Figure 2.** Comparison of TPSYS-NR vs. other modern planners in problems of IPC-2002 and IPC-2004. Note that TPSYS solved just one problem in pipesworld so it is not included in the comparison.

any city). Intuitively, the main distinction between these two classes is that, in the first class, the numeric features are an intrinsic part of the original problem and do change the structure of the plan. On the contrary, in the second class thanks to the absence of complex numeric interactions, the real structure of the plan can be easily abstracted from the numeric part, thus improving the planning performance.

In the **first experiment** (see Figures 2-a,b), we run TPSYS-NR in two IPC-2002 problems and compare its plans with those generated by LPG<sup>4</sup> [4] and TPSYS [3]. We have chosen the latest and improved version of LPG, LPG-*td* [1], because of its distinguished performance. Since we are interested in quality, we have used LPG-*td-quality*, which aims at finding a plan of good quality. Comparison against TPSYS allows us to illustrate the gains made specifically by relaxing the numeric features while focusing on the structural part of the plan. In the *zenotravel* domain (Figure 2-a), the quality of TPSYS-NR is much better than TPSYS and nearly the same as LPG-*td-quality* (in a few problems is even better). In the *rovers* domain (Figure 2-b), TPSYS-NR provides plans with very good quality.

The **second experiment** (see Figures 2-c,d) deals with some problems of the IPC-2004 (only the *satellite-IPC-04* and *pipesworld* domains could be parsed by

<sup>4</sup>Since LPG is based on a non-deterministic local search, we have run each problem five times and extracted the median values. We have used the LPG-*td* version as provided in: <http://zeus.ing.unibs.it/lpg>

TPSYS). Now we include SGPlan (with the results of the competition) in the comparison as it showed distinguished performance in IPC-2004. In the `satellite-IPC-04` domain (Figure 2-c), the plans of TPSYS-NR are better than those of TPSYS and SGPlan, but LPG-*td*-quality generates better plans. On the other hand, in the `pipeworld` domain (Figure 2-d), the quality of TPSYS-NR is significantly better than SGPlan, but only better than LPG-*td*-quality in five problems. In this domain, TPSYS solved only one problem and the quality was worse than TPSYS-NR.

To sum up we can conclude that in classes of problems with weak (or without) dependency, the approach implemented on TPSYS-NR shows an outstanding quality *w.r.t* TPSYS. On the whole, TPSYS-NR solves more problems (49) than TPSYS (39) in these classes of problems, which indicates that separating the structural part from the numeric part shows beneficial and allows to cope with more problems. On the other hand, in classes of problems with strong dependency, TPSYS-NR also shows better quality than TPSYS, though not so impressive because TPSYS-NR has still difficulties to solve all the problems. There exist three reasons for this: i) the search algorithm is not complete; ii) the heuristic estimations are too weak (not informed enough) for some problems; and iii) a sequential plan is found, but the numeric conditions cannot be satisfied. The last situation happens in the `rovers` domain (strong dependency), where rovers can only be recharged at points *with sun*. If a rover runs out of energy far away from these points, the search algorithm fails and no parallel plan is returned. This is a clear example of a plan that needs to take into consideration the numeric features as they modify its structure; i.e. the plan generation is highly influenced by the numeric constraints. If we compare TPSYS-NR with the other planners there are two clear conclusions. First, it is not as fast as them yet, and fails to find some problems (because of the reasons stated above). Second, the quality of the plans show competitive when compared to the planners awarded in the two last planning competitions, particularly with SGPlan. Actually, if we analyse the plans provided by the optimal planners TP4, HSP<sub>a</sub>\* and CPT [1] in the `pipeworld` domain, TPSYS-NR finds the optimal plan in 63% of those problems, although obviously this cannot be generalised. This makes us think that more powerful heuristics will improve the performance of the planner, while taking advantage of the easy relaxation of the numeric features proposed here.

## 5. Conclusions through related work

Traditionally, most planning approaches do not make any planning and scheduling separation in the problem. Although some researchers have attempted to perform resource abstraction while planning [7], most works tend to enrich the heuristic estimations with information on resources. `metric-FF` [5] follows that line, relaxing the delete effects to numeric variables but considering the numeric conditions in the estimations to improve their *informedness*. LPG [4] is based on stochastic local search techniques to move throughout the search space, but again no special distinction is done between planning and scheduling. On the other hand, nearly the whole number of the planners that participated in the last IPC-2004 [1] tend to simplify the original problem to make it more affordable by means of exploring any kind of abstraction. Other alternatives are more similar to the work presented in this paper: i) separating out the planning and scheduling parts of a planning problem; and ii) finding a sequential version of a plan that is later par-

allelised. The advantages of doing this are twofold. First, the original search is divided into two stages, which involve a smaller search space than that of the original problem. Second, different heuristics can be used in each stage: i) planning-oriented heuristics, to find a sequential plan in a fast way trying to optimise the length of the plan; and ii) quality-oriented heuristics, to find a good schedulable parallel plan taking into consideration the problem metric defined by the user.

This paper has presented an extension of the approach proposed in [3] to split a planning problem into **which** actions are going to be used and **when**. Hence, the main contributions of this paper are:

- A basic idea to separate the planning and scheduling parts by initially ignoring the numeric conditions and effects.
- A first stage that generates a sequential plan in a forward way by using a Hill-climbing variation. This stage focuses on the structural part of the plan, dealing with causal constraints and dependencies to solve the (sub)goals, thus dedicating the planning effort on finding a general plan.
- A second stage that parallelises the sequential plan. While it tries to allocate the actions as early as possible to compact the plan, it includes the necessary numeric actions (which usually involve resources) to satisfy the numeric conditions.
- A preliminary study of the classes of problems that result more appropriate for this approach *w.r.t.* the dependency of actions on the numeric conditions.

## Acknowledgments

This work has been partially supported by the Spanish government CICYT projects TIC2002-04146-C05-04 and DPI2001-2094-C03-03, by the Valencian government project GV04A-388 and by the Universidad Polit cnica de Valencia under project 20020681.

## References

- [1] S. Edelkamp, J. Hoffmann, M. Littman, and H. Younes, editors. In *Proc. IPC-2004*, 2004.
- [2] M. Fox and D. Long. PDDL2.1: an extension to PDDL for expressing temporal planning domains. *JAIR*, 20:61–124, 2003.
- [3] A. Garrido and D. Long. Planning with numeric variables in multiobjective planning. In *Proc. ECAI-2004*, pages 662–666, 2004.
- [4] A. Gerevini, A. Saetti, and I. Serina. Planning through stochastic local search and temporal action graphs in LPG. *JAIR*, 20:239–290, 2003.
- [5] J. Hoffmann. The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *JAIR*, 20:291–341, 2003.
- [6] R. Sanchez Nigenda and S. Kambhampati. *AltAlt<sup>P</sup>*: Online parallelization of plans with heuristic state search. *JAIR*, 19:631–657, 2003.
- [7] B. Srivastava, S. Kambhampati, and M.B. Do. Planning the project management way: Efficient planning by effective integration of causal and resource reasoning in RealPlan. *Artificial Intelligence*, 131:73–134, 2001.
- [8] T. Zimmerman and S. Kambhampati. Generating parallel plans satisfying multiple criteria in anytime fashion. In *Proc. Workshop on Planning and Scheduling with Multiple Criteria (AIPS-2002)*, pages 56–66, 2002.