Heuristic Planning with Resources

Ioannis Refanidis[1] and **Ioannis Vlahavas**[1]

Abstract. This paper presents GRT-R, an enhanced version of the GRT planner capable of explicitly handling resources. GRT is a domain independent heuristic STRIPS planner, which works in the space of the states. The heuristic computes off-line, in a pre-processing phase, estimates for the distances between the domain's facts and the goals. These estimates are utilized during the search process, in order to obtain values for the distances between the intermediate states and the goals.

We propose an explicit representation of resources in a numerical way. There are two kinds of resources: the consumable and the allocable ones. In the pre-processing phase, GRT-R assigns to the domain's facts vectors of costs. The first cost is an estimate of the distance between a fact and the goals, while the remaining costs estimate the amount of consumable resources needed to achieve that fact. GRT-R assigns each fact with a set of such vectors, each one of them corresponding to a different way of achieving the fact. During the search process, GRT-R assigns similarly each intermediate state with such a vector, based on the vectors of the state's facts, with the intention to minimize the distance between the state and the goals, without exceeding the available resources. Allocable resources are taken into account only while searching, in order to preserve the validity of the states. Performance results show that GRT-R copes well in domains that embody resources.

Key words: Planning, Heuristic Search, Automated Reasoning

1 INTRODUCTION

Resource handling is a vital issue in every real-world planning problem, demanding usually a special treatment. However, most of the known planners do not cope with resources, or, in the best case, they treat them within the poor STRIPS formalism. Recent examples are the "Mystery" and "MPrime" domains introduced in the AIPS-98 planning competition [13], where resources have been encoded as atoms.

What is needed is a different treatment of resources. This concerns both the representation of the problems and the algorithms used to solve them. In this paper we tackle with the resources in the framework of heuristic state-space planning. We concentrate on the GRT planner [15], a domain-independent heuristic state-space planner for STRIPS worlds. GRT is a descendant of the ASP/HSP planners ([2], [3], [4]).

GRT planner works as follows: Before the search phase, in a pre-processing phase, it computes estimates for the distances of all the facts of a domain from the goals. These distances are utilized during the search process, in order to obtain estimates for the distances of the intermediate states from the goals. GRT proceeds forward, using a typical best-first search strategy. GRT takes into account the positive interactions that arise while trying to achieve several facts simultaneously, producing quite accurate estimates. This does not happen with its ancestors, which consider the various facts of a domain as strictly independent, resulting usually in overestimates.

Performance results show that GRT performs quite competitive against other planners of the GRAPHPLAN [1] and SATPLAN [9] style, in domains like the blocks world, the logistics and the gripper. However, GRT, as also ASP/HSP, cannot tackle effectively other domains, like the "Mystery" or the "Mprime". In [14] two main inefficiencies of heuristic state-space planners have been identified. The first is that they often fall in local optimal states, which is difficult to overcome. The second inefficiency is that resources cannot be handled effectively within the pure STRIPS formalism. As it is proposed in [14], local optimality can be faced with the exploitation of XOR-constraints. Another approach that overcomes local optimal states, by combining hill-climbing and breadth first search strategy, has been recently proposed [8].

In this paper we confront resources in the context of heuristic planning. We extend the STRIPS formalism so as to explicitly represent resources in a numerical way. We distinguish two kinds of resources: the consumable resources and the allocable ones. Consumable resources are those that monotonically decrease, i.e. the fuel of a truck. Allocable resources are those the availability of which can both decrease and increase by an agents's actions, i.e. the free volume in a truck. We introduce the GRT-R planner, an extension of the GRT planner, which is capable of handling explicitly represented resources. In the pre-processing phase, GRT-R assigns vectors of costs to the facts. The first cost estimates the distance between a fact and the goals, while the remaining costs estimate the amount of consumable resources needed to achieve that fact starting from the goals. GRT-R assigns each fact with a number of such vectors, each one of them corresponding to an alternative way of achieving the fact. During the search process, each intermediate state is assigned similarly with a vector of costs, based on the vectors of the state's facts, with the intention to minimize the distance between the state and the goals, without exceeding the available resources. Allocable resources are also

[1] Aristotle University, Dept. of Informatics, 54006, Thessaloniki, Greece, email: {yrefanid, vlahavas}@csd.auth.gr

taken into account during the search process, in order to preserve vent the validity of the states. Performance results show that GRT-R performs quite satisfactory in domains that embody resources, like the Mystery one.

The rest of the paper is organized as it follows: Section 2 briefly presents the original GRT planner. Section 3 extends the STRIPS formalism, so as to explicitly represent resources in a numerical way. Section 4 presents in detail the GRT-R planner. Section 5 outlines how resources can be combined with XOR-constraints. Section 6 presents comparative results for the GRT-R and other state-of-the art planners. Finally section 7 summarizes the paper and poses future directions.

2 THE GRT PLANNER

In STRIPS [5], each ground action a is represented by three sets of facts: the precondition list $Pre(a)$, the add list $Add(a)$ and the delete list $Del(a)$, where $Del(a) \subset Pre(a)$. A state s is defined as a collection of ground facts and an action $á$ determines a mapping from a state s to a new state $s'=res(s,a)$. In the formalization used henceforth, the set of constants is assumed to be finite and no function symbols are used, so that the set of ground actions is finite. An action $á$ can be applied to a state s, if $Pre(a) \subseteq s$. Then, the new state s' is defined as:

$$s' = res(s,a) = s - Del(a) + Add(a) \qquad (1)$$

GRT computes, in a pre-processing phase, estimates for the distances between the domain's facts and the goals, i.e. the number of actions that should be applied to the goals, in order to achieve backwards the various facts of the domain.

In order to apply operators to the goals, the domain operators have to be inverted. Suppose that we have an operator $á$ and two states s and s', such that $á$ is applicable in s and $s' = res(s,a)$. The *inverted operator* a' of $á$ is an operator, such that $s = res(s', a')$. The inverted operator is defined from the original operator as it follows:

$$\begin{aligned} &Pre(a')=Add(a) + Pre(a) - Del(a) \\ &Del(a')=Add(a) \\ &Add(a')=Del(a) \ . \end{aligned} \qquad (2)$$

If case where the goals do not constitute a complete state description, it is impossible to apply the inverted operators to it. The solution adopted is to enrich the goals with all the facts that are not in contradiction with the goals. The new goal set constructed in this way is called the *enriched goal state*. The enriched goal state can be constructed either manually [15], or in an algorithmic way [16], or finally by exploiting domain dependent knowledge [6].

As mentioned above, in the preprocessing phase GRT assigns to each fact p of the domain an estimate for the number of backwards actions needed to achieve this fact starting from the goals. Moreover, each fact is also assigned with a list of other facts $\{r_1, r_2, ..., r_N\}$, denoted as the *related facts*. These are facts that are added by some action among those achieving p and are not deleted by any subsequent action. The related facts are potentially co-achieved while trying to achieve p and are taken into account when estimating distances.

We can outline the way in which GRT computes the above estimates with the following rules:

1) All the facts of the enriched goal state are assigned zero distances and empty lists of related facts.
2) If a fact p can be achieved by an (inverted) action a, then:
   a) The cost of achieving p is a function of the costs and the lists of related facts of a's preconditions.
   b) The list of related facts assigned to fact p is defined as:

$$rel(p)= Pre(a) \cup_i rel(q_i : q_i \in Pre(a)) \cup Add'(a) - \{Del(a), p\} \qquad (3)$$

   where $rel(f)$ denotes the related facts of any fact f and $Add'(a)$ denotes the facts that are first achieved by a ($Add'(a) \subseteq Add(a)$).

All the information obtained by the GRT algorithm in the pre-processing phase is stored in a table. Since this table is obtained through greedy regression of the goals, it is called the "Greedy Regression Table" of each problem, thus coming up the acronym GRT. During the search process, the distances between the intermediate states and the goals are estimated as a function of the costs and the lists of related facts of the state's facts (it is about the same function as the one mentioned in rule 2a above). A detailed presentation of GRT algorithm can be found in [15], while extensions concerning the manipulation of local optimal states with the use of state constraints can be found in [14].

3 REPRESENTING RESOURCES

We propose the explicit representation of resources in the most natural format, which is the numerical one. Let us consider the "Mystery" domain, firstly introduced in the AIPS-98 planning competition [13]. This domain consists of a number of cities, connected via edges, a number of packages that have to be transferred from their initial locations to their destinations and of a number of trucks. Each city has an initial amount of fuels. A truck can only move from city *c1* to an adjacent city *c2*, if *c1* has at least one unit of fuels. During the movement the truck consumes this unit.

In the original domain representation, the various quantities of fuels are represented by relations of the form:

```
(fuel fuel0)
(fuel fuel1)
(fuel fuel2)
etc.
```

while the relations between the various fuel quantities are represented also by relation as:

```
(adjacent_fuel fuel0 fuel1)
(adjacent_fuel fuel1 fuel2)
etc.
```

and the initial amount of resources in each city as[1]:

```
(city_fuel city1 fuel3)
etc.
```

Finally, actions that consume resources, e.g. moving a truck, are represented like the next one:

[1] Note that in the AIPS-98 competition there have been used different predicate and atoms names, however in this paper we translated them in more meaningful ones for simplicity.

```
(:action move
:parameters (?tr ?c1 ?c2 ?f1 ?f2)
:precondition (and (truck ?tr) (city ?c1)
(city ?c2) (adjacent_cities ?c1 ?c2)
(fuel ?f1) (fuel ?f2) (adjacent_fuel ?f1 ?f2)
(at ?tr ?c1) (city_fuels ?c1 ?f2))
:effect (and (not (at ?tr ?c1)) (at ?tr ?c2)
(not (city_fuel ?c1 ?f2))
(city_fuel ?c1 ?f1)))
```

The above representation formalism does not distinguish resources from other kind of state description information, thus preventing planners from exploiting special resource management techniques. Moreover, the number of ground actions increases according to the number of different resource availability levels, thus making harder both the preprocessing and the search phase of the planning process.

In the formalism used in GRT-R resources are explicitly declared as a special kind of state description information. So, we add declarations of the following form in each problem's definition:

```
(resource R1)
(resource R2)
...
(resource RN)
```

where Ri are the various resources. Furthermore we add to the initial state description declarations of the form:

```
(amount R1 V1)
(amount R2 V2)
...
(amount RN VN)
```

denoting the initial availability of each resource.

Next, we permit that resources can be used in relations with other atoms. Finally, we enhance operator definitions with explicit declarations of the resources consumed by them.

For example, consider the STRIPS-MYSTY-X-1 problem of the "Mystery" domain [13]. This problem has six cities and one truck, so we declare seven resources, corresponding to the fuel of the cities and the volume of the truck:

```
(resource r1)
(resource r2)
...
(resource r6)
(resource v1)
```

We relate these resources with their corresponding objects:

```
(city_fuel city1 r1)
(city_fuel city2 r1)
...
(city_fuel city6 r6)
(truck_volume truck1 v1)
```

We include in the initial state propositions for the initial availability of each resource:

```
(amount r1 1) (amount r2 2) (amount r3 4)
(amount r4 6) (amount r5 5) (amount r6 3)
              (amount v1 3)
```

Finally we declare the three operators of this domain, i.e. *move*, *load* and *unload*, by separating the resource requirements from the precondition and the effect lists. The definitions of operators *move* and *unload* follow:

```
(:action move
:parameters (?tr ?c1 ?c2 ?f)
:precondition (and (truck ?tr) (city ?c1)
(city ?c2) (adjacent_cities ?c1 ?c2)(at ?tr
?c1) (resource ?f) (city_fuel ?c1 ?f))
:effect (and (not (at ?tr ?c1)) (at ?tr ?c2))
:resources (amount ?f 1))

(:action unload
:parameters (?tr ?c ?p ?v)
:precondition (and (truck ?tr) (city ?c)
(resource ?v) (package ?p) (at ?tr ?c)
(in ?p ?tr) (truck_volume ?tr ?v))
:effect (and (not (in ?p ?tr)) (at ?p ?c))
:resources (amount ?v -1))
```

Note that it is not necessary to determine which of the resources are consumable and which are allocable. This discrimination can be performed automatically, taking into account the ground actions of the domain, i.e. resources appearing with negative cost in at least one ground action are allocable, the others are consumable.

4 THE GRT-R HEURISTIC

For each fact of a domain, in the pre-processing phase, GRT-R computes an estimate not only for the number of backwards actions needed to achieve it starting from the goals, but also for the amount of consumable resources needed. Thus each fact p is assigned with a vector of costs of the form:

$$<Actions, R_1, R_2, \ldots, R_N>$$

where N stands for the number of the consumable resources.

We denote henceforth as *vector*(p) the above vector and as *dist*(p) the estimate for the number of actions needed to achieve p (i.e. *dist*(p) is the first element of *vector*(p)). *vector*(p) could be computed by the following rules:

$$vector(p)= \begin{cases} <0,0,...0>, \text{ if } p \in Goals' \\ AGGREGATE(q_1, q_2,..., q_M) + <1,r_1,...,r_N>, \\ \text{if there is an action } a, \text{ such that for each} \\ q_i \in Prec(a),\ i=1, 2, ..., M,\ dist(q_i)< \infty. \\ r_i\text{'s},\ i=1,..,N, \text{ are the resources consumed} \\ \text{by } a. \\ <\infty, \infty, ..., \infty> \text{ otherwise} \end{cases} \quad \textbf{(4)}$$

where *Goals'* denotes the enriched goal state. Function AGGREGATE estimates the total cost (actions and consumable resources) for achieving a set of facts simultaneously, based on the costs of achieving them individually and their lists of related facts. The list of related facts of p are still computed by formula 3. Next, function AGGREGATE is presented in detail.

Function AGGREGATE

Description: The function estimates the cost of achieving a set of facts simultaneously, based on the costs of achieving them individually and on their lists of related facts.

Input: A set of facts $\{q_1, q_2, ..., q_M\}$, their vectors $vector(q_i)$ and their lists of related facts $rel(q_i)$.

Output: An estimate for the cost of achieving the facts simultaneously.

1. Let $M_1=\{q_1,q_2,...,q_M\}$. Let $Vector$= <0,0,...,0> (1 + N elements).
2. While ($M_1 \neq \emptyset$) do:
   a) Let M_2 be the set of facts $q_i \in M_1$ that are not included in any list of related facts of another fact q_j of M_1, without q_j being also included in their list of related facts. More formally:

$$M_2=\{q:q \in M_1, \forall\, q' \in M_1, q \in rel(q') \Rightarrow q' \in rel(p)\}$$

   b) Let M_3 be the set of those facts of M_1 that are not included in M_2, but are included in at least one of the lists of related facts of the elements of M_2.

$$M_3 = \{p: p \in M_1 - M_2, \exists\, q \in M_2, p \in rel(q)\}$$

   c) Sum the vectors of the facts of M_2. For equivalence classes of facts where each one is included in the list of related facts of the others, consider their common vector once. Add the result to the $Vector$.
   d) Let $M_1 = M_1 - M_2 - M_3$.
3. Return $Vector$

Function AGGREGATE does not simply add the vectors of q_i's. In each iteration facts are partitioned in equivalence classes, such that the facts of each class contain one another in their lists of related facts. The facts in each class have all been achieved by the same inverted action and they have the same vectors. So, step 2c sums the costs of these classes. The number of iterations performed by AGGREGATE is bounded by the initial size of M_1, but usually a single iteration is performed.

It is obvious that a fact p can be achieved in several ways, since it is possible for more than one inverted actions to have p in their add lists. The approach adopted by the GRT planner (also by ASP and HSP) was to consider only the minimum distance costs. However, when resources are taken into account, the notion of the minimum cost is vague. There is a minimum cost for the number of actions needed to achieve p, while there are also minimum costs for each one of the consumable resources.

GRT-R faces this problem by assigning each fact not only with a single vector of costs but with a set of such vectors. A cost vector is considered, if there is not any other vector with equal or better values in all of its elements. A separate list of related facts is also kept for each vector.

If each fact can be assigned with a set of vectors and lists of related facts, then formula 4 should be modified. More specifically, function AGGREGATE should be applied to any combination of different vectors of the q_i's, resulting in more than one vectors for fact p. From the resulted vectors, those that are totally surpassed by other vectors are rejected. Similarly, formula 3 is also applied to any combination of lists of q_i's related facts.

However, by taking into account all the possible combinations of vectors, this approach faces the risk of combinatorial explosion. This risk depends on the number of vectors per fact and the number of preconditions per action. Our experience with the mystery domain is that in average there are two vectors per fact and one or two preconditions per action, so the overhead is tractable. In addition, this overhead is totally compensated by the reduced number of ground actions.

Function AGGREGATE is also used during the search process, in order to provide estimates of the distances between the intermediate states and the goals. In this case, the input is the facts of a state, with their sets of vectors and lists of related facts. However, due to the potential large number of facts in a state, all the combinations of facts' vectors are not taken into account. Instead, a greedy approach has been adopted, which tries to find the vectors with the minimal distances for the state's facts. If the resulting vector exceeds the available resources, a limited search is performed, trying to compromise the steps with the resources.

5 RESOURCES AND XOR-CONSTRAINTS

In [14], the notion of the XOR constraints was introduced in GRT, in order to cope with the problem of local optimal states. XOR constraints are relations between sets of ground facts, where exactly one of them can hold in each state. State constraints are used in the pre-processing phase, in order to decompose a planning problem in a sequence of sub-problems that have to be solved in serial. Generally, these sub-problems are easily solvable by heuristic planners; thus the total time needed to solve them is substantially shorter than the time needed to solve the original problem.

The decomposition of a problem into sub-problems is based on identifying chains of actions linking the initial state facts with the goals. With the introduction of resources, for each initial state fact there are more than one chains of actions that link it with a goal fact, each one having its own cost vector. In this case, we can select this combination of vectors, i.e. a vector for each initial state fact, which leads to the minimum distance, without exceeding the available resources. Then, the decomposition is identical to that presented in [14].

6 PERFORMANCE RESULTS

A prototype of GRT-R has been implemented using C++. We ran the planner in some of the Mystery problems introduced in the AIPS-98 competition. The test platform was identical to the one used in the competition, i.e. Pentium II 300 MHz having 128 MB. Table 1 presents comparative results between GRT-R and BLACKBOX, HSP, IPP and STAN ([3], [10], [11], [12])

GRT-R performed well, solving the problems in quite competitive times. Note that the rest of the Mystery problems are either unsolvable or particularly hard.

Problems	BLACKBOX		HSP		IPP		STAN		GRT-R	
	actions	time	actions	time	actions	time	actions	time	actions	time
prob01	5	111	5	2233	5	100	5	40	5	200
prob02	9	6122	22	67303	9	12500	9	409	9	6590
prob03	4	615	4	3948	4	1220	4	197	4	990
prob09	8	1436	8	7275	8	1930	8	165	8	1470
prob11	7	742	12	4489	7	390	7	109	7	280
prob17	4	3484	10	86878	4	20640	4	1746	4	3940
prob25	4	117	4	2275	4	110	6	46	4	160
prob27	7	598	5	4279	7	1020	7	281	5	330
prob28	7	545	18	2746	7	260	7	64	7	220
prob29	4	507	5	3360	4	890	4	107	4	550
prob30	12	5642	18	100143	13	10020	12	7729	12	28180

Table1. Performance results for the Mystery domain (time in msecs)

For the above problems we used an algorithmic approach to compute the enriched goal state [16]. Mutual exclusive relations between the facts of the domain are computed in the pre-processing phase, in order to determine the feasibility of achieving (and co-achieving) facts. Consumable resources are taken into account in an admissible way. The application of this algorithm to each problem produces an overhead in the total processing time.

7 SUMMARY AND FUTURE WORK

In this paper we presented a new approach of handling resources in the framework of heuristic state-space planning. We introduced GRT-R, an evolution of the GRT planner, being enhanced with the ability to manipulate resources. GRT-R adopts an extended STRIPS representation, where resources are explicitly expressed in a numerical format. In a pre-processing phase, GRT-R assigns to the facts of a domain vectors, which estimate the number of backwards actions and the amount of consumable resources needed to achieve the facts starting from the goals. These vectors are utilized during the search process, in order to obtain estimates for the cost (actions plus resources) of achieving the goals from each intermediate state. Performance results show that GRT-R performs quite competitive in domains that embody resources.

The main drawback of the above approach and concurrently the most promising challenge is the utilization of an admissible heuristic for the estimation of the resource consumption. In such a case the planner would be able to prune states during search, since it would be certain that these states will not lead to the goals, due to lack of resources. A first attempt to have admissible state-space heuristics can be found in [7]. However, it remains to be investigated the applicability of this (or any other) approach to the resources framework.

REFERENCES

[1] A.L. Blum and M.L. Furst, *Fast planning through planning graph analysis*, 14th Intl. Joint Conference on Artificial Intelligence, 636-642, 1995.

[2] B. Bonet, G. Loerincs and H. Geffner, *A robust and fast action selection mechanism for planning*, 14th Intl Conference of the American Association of Artificial Intelligence (AAAI-97), 714-719. Providence, Rhode Island: AAAI Press, 1997.

[3] B. Bonet and H. Geffner, *Heuristic Search Planner*. 4th Intl. Conf. on Artificial Intelligence Planning Systems (AIPS-98) Planning Competition, Pittsburgh, 1998.

[4] B. Bonet and H. Geffner, *Heuristic Planning: New Results*, 5th European Conference on Planning, 359-371, Durham, UK 1999.

[5] R.E. Fikes and N. J. Nilsson, 'STRIPS: A new approach to the application of theorem proving to problem solving', *Artificial Intelligence*, **2**, 189-208 (1971).

[6] A. Gerevini and L. Schubert, *Inferring State Constraints for Domain-Independent Planning*, 15th International Conference of the American Association of Artificial Intelligence (AAAI-98), 905-912, Madison, Wisconsin: AAAI Press, 1998.

[7] P. Haslum and H. Geffner, *Addmisible Heuristics for Optimal Planning*, 5th International Conference on AI Planning and Scheduling, AAAI Press, pp. 140-149, 2000.

[8] J. Hoffmann, *A Heuristic for Domain Independent Planning and its Use in an Enforced Hill-climbing Algorithm*, Workshop on New Results in Planning, Scheduling and Design, ECAI-2000, Berlin, August 2000.

[9] H. Kautz and B. Selman, *Pushing the envelope: Planning, propositional logic and stochastic search*, 13th International Conference of the American Association of Artificial Intelligence (AAAI-96), 1194-1201. Portland, AAAI Press, 1996.

[10] H. Kautz and B. Selman, *BLACKBOX: A New Approach to the Application of Theorem Proving to Problem Solving*, Workshop on Planning as Combinatorial Search (AIPS-98). Pittsburgh, 1998.

[11] J. Koehler, B. Nebel, J. Hoffmann and and Y. Dimopoulos, , *Extending Planning Graphs to an ADL Subset*, 4th European Conference on Planning (ECP-97), 273-285. Springer LNAI 1348, 1997.

[12] D. Long and M. Fox, 'Efficient Implementation of the Plan Graph in STAN', *Journal of Artificial Intelligence Research*, **10**, 87-115 (1998).

[13] D. McDermott, *The official AIPS-98 competition page. ftp://ftp.cs.yale.edu/pub/mcdermott/aipscomp-results.html* (1998).

[14] I. Refanidis and I. Vlahavas, *Exploiting State Constraints in Heuristic State-Space Planning*. 5th International Conference on Artificial Intelligence Planning Systems (AIPS-2000). Breckenridge, 2000.

[15] I. Refanidis and I. Vlahavas, *GRT: A Domain Independent Heuristic for STRIPS Worlds based on Greedy Regression Tables*, 5th European Conference on Planning, 346-358 (preprints). Durham, UK, 1999.

[16] I. Refanidis and I. Vlahavas, *On Determining and Completing Incomplete States in STRIPS Domains*, IEEE International Conference on Information, Intelligence and Systems. Washington, US, 289-296, 1999.