PIPSS*: A System based on Temporal Estimates

Yolanda E-Martín, María D. R-Moreno, and Bonifacio Castaño

Abstract AI planning and scheduling are two closely related areas. Planning provides a set of actions that achieves a set of goals, and scheduling assigns time and resources to the actions. Currently, most of the real world problems require the use of shared and limited resources with time constraints when planning. Then, systems that can integrate planning and scheduling techniques to deal with this kind of problems are needed.

This paper describes the extension performed in PIPSS (Parallel Integration Planning and Scheduling System) called PIPSS*. PIPSS combines traditional state space heuristic search planning with constraint satisfaction algorithms. The extension is based on heuristic functions that allows the planner to reduce the search space based on time estimations that imposes temporal constraints to the scheduler. The purpose is to achieve a tighter integration respect to the previous version and minimize the makespan. Results show that PIPSS* outperforms state of the art planners under the temporal satisficing track in the IPC-08 competition for the tested domains.

1 Introduction

Some planning systems use heuristic functions to search in the state space. These planners are called Heuristic Search Planners (HSPs). HSPs are based on the use of evaluation or heuristic functions, combined with search algorithms to explore the search space toward a desired state.

Yolanda E-Martín
Departamento de Automática, Universidad de Alcalá e-mail: yolanda@aut.uah.es

María D. R-Moreno
Departamento de Automática, Universidad de Alcalá e-mail: mdolores@aut.uah.es

Bonifacio Castaño
Departamento de Matemáticas, Universidad de Alcalá e-mail: bonifacio.castano@uah.es

This work is focus on the PIPSS [2] system. PIPSS is composed of HPP [1] and OOSCAR [9] as planner and scheduler respectively. The relaxation heuristic used on HPP is based on ignoring delete actions. This technique defines a relaxation problem ignoring the action delete effects. The heuristic cost is estimated as the length, in actions number, between a certain state s and any goal. The estimation is extracted from a planning graph.

The objective of PIPSS* is to extract temporal estimations from the planning graph, moreover the cost heuristic, that allow us to reduce the makespan of a problem. Thus, from calculated estimates, we will force the scheduler to find a plan whose makespan does not exceed the calculated estimation in the planner. Thereby, if the scheduler does not find a plan that satisfies the restriction, shall inform the planner in order to find another plan that does comply.

That is, the underlying idea developed on top of PIPSS is to get plans for achieving the goals in as few steps as possible.

The paper is structured as follows. The next section details the PIPSS system components. Then, the PIPSS extension (PIPSS*) is described. Next, experimental results are discussed. Finally, some conclusions are presented.

2 PIPSS

PIPSS (Parallel Integrated Planning and Scheduling System) is a system that integrates planning and scheduling. It is able to solve planning problems with time and multicapacity resources information through scheduling techniques.

PIPSS emerges from the union between the HPP planner and the main scheduling algorithm from the OOSCAR system. The next subsections explain each of its components.

2.1 HPP: Heuristic Progressive Planner

HPP is a heuristic progressive PDDL planner based on FF [10]. But HPP introduces changes in the operators instantiation. It includes a new module called *analysis reachability module* that is able to exclude irrelevant domain-dependent operators in the planning process. The analysis performed by this module avoids the expansion of several parts of the search tree[1] and can obtain better results in the planning process.

In particular, HPP builds before the planning process, three sets of operators called *A*, *B* and *C vectors*. The *A vector* contains all the possible operators of the instantiation process (as FF does). The *B vector* has fewer operators than *A vector*, which are the result of employing the relaxed GraphPlan [12] that FF uses when it

[1] A Search tree is defined as a graph that considers all possible paths in the network. The tree nodes represent states, and its branches executed actions that achieve states.

calculates its heuristic. Finally, the *C vector* is generated using an additive heuristic h_{add} as the one used in HSP planner [11] for computing the heuristic cost.

2.2 ISES Algorithm

OOSCAR (Object-Oriented SCheduling ARchitecture) is a scheduling system that works with time and resources to deal with the RCPSP/max problem (Resource Constrained Project Scheduling Problem). The RCPSP problem is a kind of scheduling problem that attempts to look for a way of ordering activities along time, which are restricted by precedence relations (some of them cannot start until some others have finished) and which use renewable resources (resources whose availability is a fixed quantity for every unit of time). The goal is to find initial times for activities so that the makespan is minimized.

The main algorithm used by OOSCAR to find feasible solutions for the RCPSP/max problems is ISES (Iterative Sampling Earliest Solutions) [8]. Basically, ISES is a sample optimization algorithm that iterates another algorithm called ESA (Earliest Start Algorithm), which is in charge of returning time- and resource- consistent solutions. ESA solves temporal restrictions using a temporal network (TN)[2] and avoids conflicts due to resources by imposing additional precedence relations between pairs of activities that are responsible of such conflicts. ISES just asks ESA for several solutions to try to find one with a better makespan.

2.3 PIPSS Architecture

The outstanding features of PIPSS, which allows to carry out the extension described in the next section, is its open and object-oriented architecture for planning and scheduling integration . This architecture is based on interfaces that allow interoperability of different planning algorithms, scheduling techniques and search or integration schemes of both. Also this gives the ability to be configured to run on any combination of the possibilities available. Figure 1 shows the architecture. 2.

PIPSS has two kind of planning search: Enforced Hill Climbing and Greedy Best-First Search — *Planning Interface*. One type or scheduling, ISES, or the possibility to disable scheduling — *Scheduling Interface*. And also several types of planning and scheduling integrated search schemes in *Search Interface*. Besides this, the system can use any of the three vectors that HPP builds (see previous subsection).

[2] A TN is defined in [3] as a directed graph whose nodes represent Time Points (TPs) and whose arcs represent distance constraint between TPs. Nodes represent points on a time line where temporal changes happen, and arcs represent activities duration and distance constraints between activities and events.

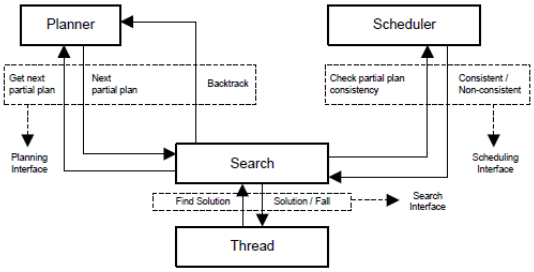

Fig. 1 PIPSS Architecture

3 PIPSS*

In this section we describe the extension introduced in PIPSS, called PIPSS*, that minimizes the makespan of its predecessor. To understand the implementation, it is important to review the relaxed GraphPlan concept.

3.1 Relaxed GraphPlan

GraphPlan [12] is based on a compact structure called a Planning Graph. A Planning Graph is a directed graph composed of two types of nodes distributed on several levels. Starting at level 0, the even levels contain information about the propositions that are reached — *fact layers*, denoted as *Sx*, where *x* is the level number. The odd levels contain actions whose preconditions are present at the previous level — *action layers*, denoted as *Ax*. The edges represent the precondition and effect relations (positive (*add*) and negative (*del*)), between facts and actions.

The Relaxed Graphplan build process has two distinct phases:

- Graph Expansion: builds the planning graph until all goals are reached. In level 0, the fact layer is made up of the initial state (*InitState*) problem facts (in Figure 2 is represented as letter *q*), and the action layer contains all actions applicable to *InitState* (in the example is showed as letters *A* and *B*) . The union of all those actions's add effects (ignoring delete effects) with the facts that are already there, forms the second fact layer (letters *q* and *r* in the figure). This process is repeated until a fact layer is reached that contains all goals (in the example goal state are letters set *q*, *r* and *p*).
- Plan Extraction: a backward-chaining strategy level by level is used. The process consists of: given a set of objectives in t, where t is the last graph level, find a set of actions in level t-1 at reach these goals. Then, the preconditions of these actions form a set of subgoals in t-1. If the new objectives of t-1 can be obtained in t-1 steps, then the original objectives can be achieved in t steps. If on the contrary they cannot be reached, the process looks for a different action combination. The

process continues until a solution is found, or stops if any of actions combinations are valid.

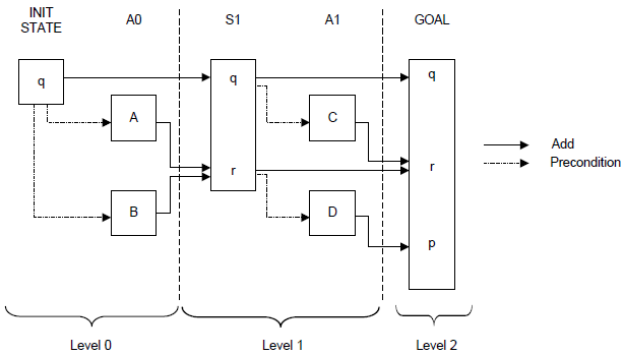

Fig. 2 Relaxed GraphPlan General Structure

The relaxed GraphPlan used as a h_{RG} heuristic consists on building, in any search step, a planning graph from which a solution to the relaxed problem could be extracted. The length of the solution, i.e. the number of actions selected is taken as an estimation of how far the goal is from the current state. HPP like FF, uses this heuristic.

3.2 Extension

The aim of the extension is to provide a tighter communication between the two main components of PIPSS. This means that the planner and the scheduler exchange information in order to guide the search process, thus pruning some parts of the states space. In our first implementation of PIPSS the only information that the planner and the scheduler exchanged was the time and the resources consistency returned by the scheduler. The scheduler could not give any other information back to guide the planner process until the inconsistency was produced. So sometimes we were spending time on searching for a solution that looked in advance we could have known it was inconsistent. Then, the motivation of our work is to estimate the makespan and detect earlier inconsistencies. For this extension we have defined the following terms associated to the relaxed GraphPlan and the search tree.

- The related definitions to the relaxed GraphPlan are:
  - W_{ij} is called the duration of action i at level j. For each level *SumWeight* (SW_j) is defined as:

$$SW_j = \sum_{i=1}^{a_j} W_{ij} \tag{1}$$

Where a_j is the number of actions of level j, and SW_j the sum of all action durations of level j.

– *MacroInitialEstimation (MIE)* is the sum of durations of all the actions of a relaxed GraphPlan (*RG*). It is defined as:

$$MIE(RG) = \sum_{j=0}^{nl} SW_j = \sum_{j=0}^{nl} \sum_{n=1}^{a_j} W_{nj} \tag{2}$$

– *MinimumWeight0 (MinW0)* is the minimum value of the actions durations of the level *j=0*. It is defined as:

$$MinW0(RG) = minW_{i0}, i = 1..a_0 \tag{3}$$

Figure 3 shows an example calculation for the variables W_{ij}, SW_j, *MIE(RG)* and *MinW0(RG)*. Note that every action has an associated duration, taken from the problem, represented by the W_{ij} variable. In level 0 we have $W_{10} = 3$ and $W_{20} = 8$. And for level 2 there are $W_{11} = 7$ and $W_{21} = 5$. In addition, we can see what are the values of the variables SW_j for each level and how it is computed. The values obtained are $SW_0 = 11$ and $SW_1 = 12$. Thus, we can observe that the variable value *MIE(RG)* corresponds to the sum of all the variables SW_j previously calculated. Therefore $MIE(RG) = 23$. Al last, for computing *MinW0(RG)* is observed that there are two actions at level 0, then the *MinW0* value for *RG* will be equal to the $min(W_{10}, W_{11})$ value , i.e. *MinW0(RG)=3*.

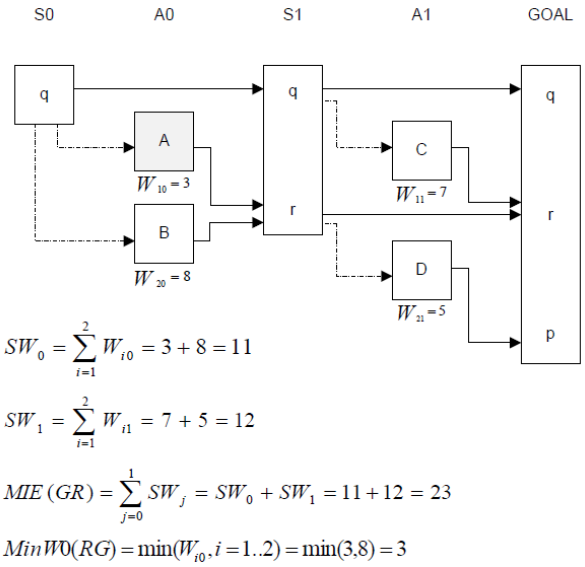

Fig. 3 Example Calculation of Variables *W, SW, MIE* and *MinW0*

- The related definitions to the search tree are:
  - For each tree node V_k (where k is a node identifier) associated to RG_k, where $V_0 = RG_0 = RG$ is the root tree node, the (2) and (3) equalities become true:

$$MIE(V_k) = MIE(RG_k) \tag{4}$$

$$MinW0(V_k) = MinW0(RG_k) \tag{5}$$

  - $MK(V_k)$ is the makespan value returned by the temporal network for the $V_0 - V_{k-1}$ branch. In particular, $MK(V_0) = 0$.

$$MK(V_k) \tag{6}$$

  - *MicroEstimation (mE)* is the sum of the minimum value of the actions durations of the level 0 and the makespan value returned by the scheduler. It is defined as:

$$mE(V_k) = MinW0(V_k) + MK(V_k) \tag{7}$$

    The Figure 4 shows a search tree where the node V_2 is the latest that has been expanded. Suppose that the Figure 3 shows the RG associated with node V_{11}. Then, the value of (5) is 3. Suppose that $MK(V_1) = 6$ then, the value of (7) is 9.

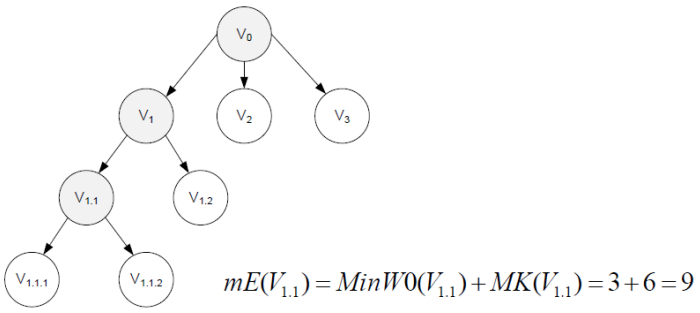

Fig. 4 Example Calculation of Variable *mE*

Next, the concepts previously defined are explained within the system PIPSS*.

Figure 5 shows the algorithm used. Initially we call the function *ExpandNode* that implements a pruning technique to reduce the search space. It takes as a makespan the value of $MIE(V_0) = MIE(RG_0) = MIE(RG)$ if the user does not introduce an initial makespan value. This estimate may not approach the real value because it assumes that the plan is executed sequentially, while in reality it is possible that certain actions can be run in parallel. However, this estimate is used to establish an upper bound. So during the search process those nodes whose value (computed by the

sum of (4) and (6)) exceed $MIE(V_0)$ are not taken into account in the search space. As these branches would guide towards solutions that deviate from the minimum makespan. That is why they are discarded.

Every time a tree node is expanded, the nodes which do not fulfill the mentioned condition are ruled out. Among the nodes that do fulfill it (saved in a list ordered by decreasing value of h_{RG}), the node with the best h_{RG} heuristic is selected (using *getBestNode*).

When this happens, it generates an incomplete partially ordered plan from an incomplete totally ordered plan, which will be sent to the TN in order to find a temporal- and resource- consistent solution. For this partial plan an estimate of the makespan is calculated through the *CalculatedES* function. We do this using (7). The reason to compute in this way the estimation is because the plans sent to the temporal network are partial plans. That is, in every iteration a new operator extracted from level 0 is included in the partial plan. As the aim is to minimize the makespan of the solution, the most promising estimation is equal to the sum of (6) and (5).

If the solution returned by the temporal network is consistent, the function *ExpandNode* is called again with the successor with the best h_{RG} value. Otherwise, the function selects the next successor.

Function EstimateMk (cNode,openL,Mk)

cNode the actual state of the problem
mE MicroEstimation value of *cNode*
bNode the best heuristic succesor of *cNode*
openL set of nodes that belong to the search space, initially empty
P a plan, initially empty
$MK(V_k)$ makespan value returned by the temporal network
Mk cNode makespan value
TN temporal network

```
1. openL = ExpandNode(cNode)
2. bNode = getBestNode(openL)
3. P = getPartialPlan(cNode,P)
4. mE = CalculateES(cNode)
5. If IsConsistent(getTNsolution(P),mE)
   Then
      Mk = MK(cNode)
      EstimateMk(bNode,openL,Mk)
   Else
      EstimateMk(cNode,openL,Mk)
6. return P, totalTime
```

Fig. 5 PIPSS* pseudo-algorithm.

4 Experimental Results

The PIPSS extension has been tested against several participants planners of the International Planning Competition of 2008 (IPC-08)[3]. This section is divided into three subsections, each one corresponding to the domains used for experimental purposes.

The three domains are standard PDDL durative domains. The first is *Satellite*, taken from IPC-02[4]. The other domain is called *PipesWorld*, taken from IPC-06[5]. And the last domain called *Openstacks*, is taken from IPC-08. PIPSS* only supports a subset of PDDL2.1 [7], that is why we have chosen these domains.

Six planners have been used for the experimental evaluation:

- PIPSS*: uses the *A vector*, greedy best-first search, ISES and Temporal Search.
- $SGPlan_6$ [4]: partitions the planning problems into subproblems by parallel decomposition, and it uses Metri-FF as the search algorithm. $SGPlan_6$ is the only planner (from IPC-08) that supports PDDL3 [6].
- Metric-FF [14]: is the numeric version of the FF planner. It does a forward search in the state space and uses the relaxed GraphPlan heuristic that ignores the delete lists. It supports PDDL2.1.
- CPT [13]: is an optimal temporal planner based on POCL [13] and constraint programming with makespan optimization via branch and bound. It supports PDDL2.1.
- DAE1 and DAE2 [5]: hybridizes evolutionary algorithms with classical planning methods by dividing the initial problem into a sequence of subproblems, solving each subproblem in turn, and building a global solution from the subproblem solutions with the CPT planner. Each planner uses different strategies for the description of intermediate goals. The first one, DAE1, uses only the predicates that are still present in the goal of the problem, and the second one, DAE2, uses a subset of all predicates.

All these planners (except PIPSS*) took part of IPC-08 temporal satisficing track whose objective is to minimize the total plan duration (makespan).

For each domain we show a table that represents the makespan for each solved problem. The bold values symbolize the best makespan. The total time given to solve each problem was 1800 seconds and all the planners have been launched under the same platform, Windows XP, and those that only run over Linux have been executed under a virtual machine in the same PC. The computer run with an Intel Core 2 Duo processor (2.27Ghz) and 2Gb of RAM memory. Next, an evaluation about the runtime is explained.

---

[3] http://ipc.informatik.uni-freiburg.de/

[4] http://planning.cis.strath.ac.uk/competition/

[5] http://zeus.ing.unibs.it/ipc-5/

4.1 Satellite Domain

The objective of the Satellite domain is to collect image data from the earth surface through a network of satellites that cover various zones. Every satellite can perform several actions such as turning, switching on and off on-board instrumentation, taking pictures, and selecting among different modes of calibration. The total number of problems is 36 and the complexity gradually increases the number of satellites and instruments, and the number of directions to perform on them. The first problem begins with a single satellite that can take seven different directions, with one instrument and three calibration modes. The last problem has 9 satellites that can take 204 different directions, 22 instruments and 5 modes of calibration.

Table 1 shows the percentage of problems solved by each planner. SGPlan_6 solves the highest number of problems followed by PIPSS*. CPT, DAE1 and DAE2 solve very few problems.

Table 1 Problems solved in the Satellite Domain

Name	SGPLAN_6	PIPSS*	METRIC-FF	CPT	DAE1	DAE2
Percentage	86%	50%	44%	14%	14%	14%

Table 2 shows the makespan of the solutions for all the planners. PIPSS* gets better makespan values than the other five until problem 22. From here PIPSS* doesn't solve any more problems while SGPlan_6 solves until number 30. Metric-FF solves fewer problems than the other 2. It is important to remark that SGPlan_6 was the winner of IPC-08 temporal satisfying track. Planners CPT, DAE1 and DAE2 only solve until problem 7. This is why the values are kept constant after this problem.

Table 2 Values Problems solved in the Satellite Domain

SYSTEM	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	P21	P22
SGPLAN_6	221	236	236	422	392	545	344	406	879	912	864	1434	953	717	837	1200	**869**	803	1667	1728	1889	1615
PIPSS*	-	**128**	**128**	**128**	-	**92**	144	**66**	**100**	**137**	**215**	**180**	**110**	**195**	**242**	-	-	**82**	**196**	**170**	**216**	**329**
METRIC-FF	208	225	225	355	251	262	293	224	-	-	281	-	-	463	-	**687**	-	322	467	-	1408	1536
CPT	**132**	152	152	-	**103**	-	**56**	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DAE1	**132**	152	152	-	**103**	-	**56**	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DAE2	**132**	152	152	-	**103**	-	**56**	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Regarding the runtime, DAE1 and DAE2 have the worst performance. PIPSS* is the slowest of the four other planners compared. CPT is the third best followed by SGPlan and Metric-FF.

4.2 PipesWorld Domain

The PipesWorld domain have a network of full fuel tanks (petrol, diesel and derivatives) and fuel empty tanks. They are all interconnected through a complex pipelines network. In turn, the tanks are grouped into areas or sectors. The purpose of this domain is to transport different fuel types of some tanks to others. The complexity of the problem increases with the increasing number of tanks and pipes, besides the different types of fuel. There are 50 problems in the domain. The first begins with 6 available tanks, 3 areas and 2 pipes. Moreover, the last problem has 30 tanks, 5 zones and 5 pipes. Depending on the used pipe the transport time will be higher or lower.

Table 3 shows the planners along with a percentage, indicating how many problems they solve. The best planner in this case is Metric-FF followed by PIPSS*. DAE2 planner is the worst in these terms.

Table 3 Problems solved in the PipesWorld Domain

Name	METRIC-FF	PIPSS*	SGPLAN$_6$	CPT	DAE1	DAE2
Percentage	54%	40%	20%	12%	12%	6%

Table 4 shows the makespan of the solutions for the CPT, DAE1, DAE2, Metric-FF, PIPSS* and SGPlan$_6$ planners. PIPSS* obtains better makespans than the other five until problem 24. From here PIPSS* solves one more problem while Metric-FF solves 9 more. SGPlan$_6$ only solves 10 problems. The rest of planners solve until problem 6.

Table 4 Values Problems solved in the PipesWorld Domain

SYSTEM	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	P21	P22	P23	P24
SGPLAN$_6$	6	22	16	16	14	14	14	22	40	46	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PIPSS*	**2**	**5**	**5**	**6**	**5**	**6**	**5**	**7**	**5**	**13**	**12**	12	-	-	20	-	**16**	24	-	-	**4**	**10**	**4**	**11**
METRIC-FF	6	22	16	16	14	14	18	24	46	54	19	**11**	-	**19**	**12**	-	-	**18**	**11**	**21**	10	21	-	-
CPT	6	20	12	12	12	12	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DAE1	3	10	6	**6**	6	**6**	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DAE2	3	10	56	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Regarding the runtime, the best execution time is achieved by Metric-FF and with almost the same execution time comes SGPlan$_6$. Then CPT and finally PIPSS*. Here also DAE1 and DAE2 have the worst runtime.

4.3 Openstacks Domain

In the Openstacks domain a manufacturer has a number of orders, each for a combination of different products with the restriction of one product at time. There are 30 problems in the domain. The first problem begins with 5 orders, and the last starts with 31 orders.

Table 5 shows the planners used along with the solved problems in %. In this domain Metric-FF, PIPSS* and $SGPlan_6$ solve all the problems. CPT, DAE1 and DAE2, with the same percentage results, are the worst planners.

Table 5 Problems solved in the Openstacks Domain

Name	METRIC-FF	PIPSS*	$SGPLAN_6$	CPT	DAE1	DAE2
Percentage	100%	100%	100%	13%	13%	13%

Table 6 shows the makespan of the solutions. PIPSS* obtains better makespan than the other two. Although, in the first six problems all the others planners performs better than PIPSS* as problems become more complicated PIPSS* improves significantly the results.

The reason PIPSS* offers greater execution times may be because the temporal restriction imposed for the scheduler is very strict, and repeatedly forces the planner to perform backtrack. But in the three domains PIPSS* gets the better makespans in the problem solved.

5 Conclusions

In this paper we have described PIPSS*, the extension performed on the PIPSS system. PIPSS* has been designed with the purpose of producing a tighter communication between the two systems that is composed of, the planner HPP and the scheduler algorithm ISES.

Results show that PIPSS* performs, in the tested domains, better than the best IPC-08 planner under the temporal satisficing track, $SGPlan_6$. PIPSS* develops a method that estimates, based on temporal heuristic that allows to guide the search process, solutions close to the minimum makespan. This is accomplished by setting certain temporary variables for each tree successor node. Additionally, a makespan restriction is imposed upon the scheduler in order to find a solution that does not exceed the estimated value.

Acknowledgements This work has been founded by the Junta de Comunidades de Castilla-La Mancha project PEII09-0266-6640.

Table 6 Values Problems solved in the Openstacks Domain

ITEM	SGPLAN$_6$	PIPSS*	METRIC-FF	CPT	DAE1	DAE2
P1	87	230	87	**84**	85	**84**
P2	168	128	157	**114**	127	**114**
P3	170	100	148	**85**	87	**85**
P4	131	133	148	**87**	111	**87**
P5	115	**107**	116	-	-	-
P6	195	**145**	176	-	-	-
P7	168	**108**	112	-	-	-
P8	178	**112**	169	-	-	-
P9	199	**108**	124	-	-	-
P10	214	**110**	214	-	-	-
P11	201	**123**	176	-	-	-
P12	368	**135**	139	-	-	-
P13	318	**146**	223	-	-	-
P14	265	**122**	139	-	-	-
P15	279	**116**	135	-	-	-
P16	288	129	**120**	-	-	-
P17	396	**162**	195	-	-	-
P18	295	**135**	281	-	-	-
P19	305	**159**	195	-	-	-
P20	397	**144**	253	-	-	-
P21	408	**137**	259	-	-	-
P22	432	**146**	197	-	-	-
P23	566	**147**	207	-	-	-
P24	493	**163**	286	-	-	-
P25	441	**205**	211	-	-	-
P26	446	**151**	243	-	-	-
P27	312	**158**	261	-	-	-
P28	507	**183**	216	-	-	-
P29	436	**156**	218	-	-	-
P30	387	**181**	265	-	-	-

References

1. M. D. R-Moreno and D. Camacho and A. Moreno: HPP: A Heuristic Progressive Planner. In: The 24th Annual Workshop of the UK Planning and Scheduling Special Interest Group, pp: 8-18, December, London, UK (2005).
2. J. Plaza and M. D. R-Moreno and B. Castano and M. Carbajo and A. Moreno: PIPSS: Parallel Integrated Planning and Scheduling System. In: The 27th Annual Workshop of the UK Planning and Schedulinmg Special Interest Group, December, Edinburgh, UK (2008).
3. R. Cervoni and A. Cesta and A. Oddi: Managing Dynamic Temporal Constraint Networks. In: The 2nd International Conference on Artificial Intelligence Planning Systems, Chicago, USA (1994).
4. C. W. Hsu and B. W. Wha: The SGPlan Planning System in IPC-6. In: International Planning Competition 6, Corvallis, OR, USA (2008).
5. J. Bibai and P. Savéant and M. Schoenauer and V. Vidal: DAE: Planning as Artificial Evolution (Deterministic part). In: International Planning Competition 6, Corvallis, OR, USA (2008).
6. A. Gerevini and D. Long: Plan Constraints and Preferences in PDDL3. The Language of the Fifth International Planning Competition. Technical Report, Department of Electronics for

Automation, University of Brescia, Italy (2005).
7. M. Fox and D. Long: PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. University of Durham, February, Durham, UK (2002).
8. A. Cesta and A. Oddi and S. Smith: An Iterative Sampling Procedure for Resource Constrained Project Scheduling with Time Windows. In: The 16th International Joint Conference on Artificial Intelligence, Stockholm, Sweden, (1999).
9. A. Cesta and G. Cortellessa and A. Oddi and N. Policella and A. Susi: A Constraint-Based Architecture for Flexible Support to Activity Scheduling. Lecture Notes in Artificial Intelligence, **2175**, 369–381 (2001).
10. J. Hoffmann and Bernhard Nebel: The FF Planning System: Fast Plan Generation Through Heuristic Search. Artificial Intelligence Research, **14**, 253–302 (2001).
11. B. Bonet and H. Geffner: Planning as Heuristic Search. Articial Intelligence Research, **129**, 5–33 (2001).
12. A. Blum and M. Furst: Fast Planning Through Planning Graph Analysis. Artificial Intelligence Research, **90**, 281–300 (1997).
13. V. Vidal and H. Geffner: Branching and Pruning: An Optimal Temporal POCL Planner based on Constraint Programming. Artificial Intelligence Magazine, **170(3)**, 298–335 (2006).
14. J. Hoffmann: The Metric-FF Planning System: Translating 'Ignoring Delete Lists' to Numeric State Variables. Artificial Intelligence Research, **20**, 291–341 (2003).