Strategies to Configure a Portfolio

Isabel Cenamor
Departamento de Informática, Universidad Carlos III de Madrid
Avda. de la Universidad, 30. Leganés (Madrid). Spain
icenamor@inf.uc3m.es

Abstract

The portfolios of planners have arised as a great idea in automated planning. Their main challenge is how to combine portfolio components, i.e. the base planners. The most extended solution is the static combination of the planners, where the same configuration is used for all the problems in all the domains. This solution is not very flexible, and it might be less efficient than a particular combination of planners for every single problem. In the middle, a portfolio could be configured for a specific domain, so all the problems in such domains are solved with the same portfolio configuration. For this reason, we analyze different ways to configure a portfolio for each problem and create some strategies and techniques to exploit the planning portfolios.

Introduction

In the last years, different planning algorithms has been created in the planning community. However none of them dominates in all cases (domains and/or problems). This fact is presented in the International Planning Competition (IPC), where there is a global winner in each track, but this winner is not the best planner for all problems or in all domains. For this reason, the idea of combining different algorithms or planners is excellent because we can always find an optimal combination. This combination requires the selection of the best planner in each problem. However finding an optimal combination "a priori" is a hard task.

In the state of art, there are several examples. One of them is Fast Downward Stone Soup (FDSS) (Helmert et al. 2011), that won the optimal track and obtained a second position in the sequential satisficing track of 2011 (IPC-7). FDSS uses the same configuration in all domains and problems. Such configuration is obtained with a learning phase with a huge number of problems in different domains. Fast Downward (FD) autotune (Fawcett et al. 2011) refers to the specific configuration resulting from using mean runtime to find an initial satisficing plan as the optimisation metric.

Therefore, FD systems described above can be considered as a set of search method with different heuristics. The specific configurations are the result of a large experimentation to extract a good performance independently of the domain. For each configuration, a sorted list of running time and algorithm is defined, with a group of heuristics. Opposite, our research try to extract a different configuration per problem. Our approach is a dynamic method to configure a portfolio in a flexible way that differentiates the features (and hence, the complexity) of the problems. The utilization of several planners strengthen the diversity of the portfolio, because it does not always run the same configuration or the same components. And it is configurable in function of the system constraints because the environment is not always the same; for example, the time limit, the memory, the number of solutions expected, time request, etc. may be different, depending on the specific deployment of the portfolio.

PbP planner (Gerevini, Saetti, and Vallati 2009) creates a portfolio with one configuration per domain. PbP and PbP2 won the two last learning tracks and introduce several differences in relation to FD system. In the learning track, knowledge is learned for each domain and, in consequence, the planner is configured with a different method. This planner receives knowledge for each domain, and its configuration depends of this knowledge. This fact supposes that the planner does not have the same configuration in all cases like FD system. This approximation also contracts with our research because our approach is more specific: not only it is possible to consider different configuration per domain, but per problem.

We introduce a novel idea that can be used for domain-independent or domain-dependent contexts, given that a portfolio may have a different configuration per instance. We can create a different portfolio as function of the requirements or the constraints, and it can be adaptable for future requirements. There are a lot of possibilities in such constraints, like the time limit. Nowadays, the competitions usually run problems for 1800 seconds (a long time that permits uniform distributions of running time among all base planners to provide good results). However, the system could have a 300 time bound, or even a small slot of time such as 10 seconds. Another possibility is restricting the planner components to compare different portfolio combination techniques. In this situation, the comparative between techniques is fair because the only difference is the chosen algorithm. Besides, portfolios could use more than one core and they may require an special configuration. Other restriction is the number of output planners, for example, selecting one planner for each new instance. Another important issue is the amount and diversity of available training data, which is related also with the specific objective of the portfolio. If we want to build a portfolio for a specific domain, it may be interesting to evaluate whether using training data from executions in other domains is useful or not. As a conclusion, there are a lot of requirements that it is possible to consider to configure a planner portfolio.

In this article we explain a general idea of our research, its process and the preliminary results. We introduce the inputs of the portfolio, its different configurations and the common elements.

Learning Planning Knowledge

The executions of planner generate a lot data, and these data might be analysed to create knowledge through the CRISP-Data Mining process (Chapman et al. 2000; Han, Kamber, and Pei 2006). This methodology appears in the figure 1. The first phase in this process is the business understanding, which means we have to know the objective to extract the new knowledge. Our aim is extracting knowledge to configure a portfolio. The input of the system are the base planners, the domains, the problems and the constraints. An example of the constraints are the global time, the memory limit, etc.

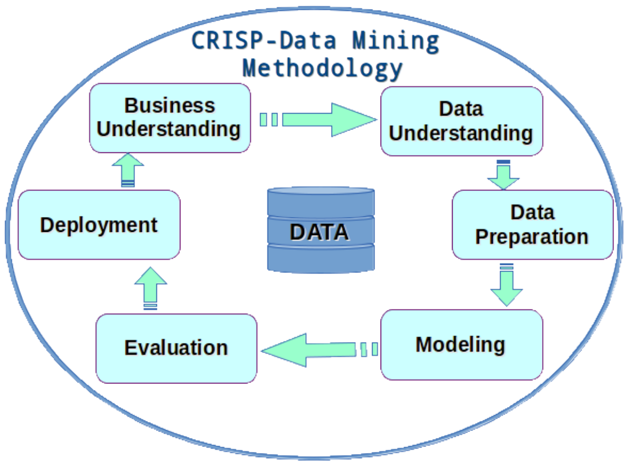

Figure 1: Data workflow of the mining process

A good instance characterization is necessary. For this reason, the planning problems are represented with several features to differentiate each other. An accurate representation of the problems is an important point in this process because it is the source of following steps. This procedure is realized in the second step of the data mining process: data preparation. In this part of the process, the data is cleaned, modified, and also it is possible to create new derived attributes.

The following step is modelling the data through predictive models and evaluating them to select the model with the best performance. In this part, we select and apply various mining functions for the same objective. When the final modelling phase is completed, a model of high quality has been selected for the deployment.

The last step is the deployment, where we use the models to support the configuration of the portfolio. In this process, we use the learning process to exploit the combination of planners. The deployment process appears in the algorithm 1. The input of this algorithm is the problem, the domain and the constraints. If the system has a default configuration, the output is a base strategy. In other case the new problem is transformed in several features and it queries the decision model for predicting the planners and their times. The assigned time has two different variants, the first one is assigned the same time per each planner and the other is to predict the time queriying a regression model.

```
Data: Problem, Domain and Requirements
Result: Configuration Portfolio
if Default Configuration then
    Assign fix time $t_n$ per each planner $p$ in $\mathcal{P}_n$;
else
    Extracted Features;
    Select potential planners in $\mathcal{P}_n \longrightarrow \mathcal{P}_c$ where $\mathcal{P}_n \supseteq \mathcal{P}_c$ ;
    if Regression Model then
        for planner $p$ in $\mathcal{P}_c$ do
            Obtain a time a $p \longrightarrow t_{pn}$
        end
    else
        Assign the same time per each planner $\mathcal{P}$ in $\mathcal{P}_c$;
    end
end
```

Algorithm 1: Deployment of the System

The system configuration have been evaluated with different requirements and each part changes different elements. The general process is the same, but we change initial data, including more problems and domains, the characterization process,including more features, the numbers of CPU, mono core or multi core and the time (1800 and 300 seconds).

Initial Settings

In the first case, we used all the planners of the last IPC in sequential satisficing track (García-Olaya, Jiménez, and Linares López 2011). The domains and the problems came from the same track. The training instances consist of a group of features that describe the problem (47 features) (Cenamor, de la Rosa, and Fernández 2012), the time of the best solution for each planner, and if a planner find the solution. The dataset has 7560 from 27 planner with 20 problem for each of the 14 domains. The created models try to decide which is the correct planner to solve a problem and the execution time the planner needs. We evaluate the models with two different techniques. One of them uses half part of the data for training and the other half to test in the fist evaluation case. The training and test data have similar distributions, i.e. there is the same number of problems per each domain. In the other evaluation method, we used leave one domain out: training with all domains but one domain simulating the results in new domains.

The results of this part has good performance, however it might exist a problem. It is possible that a created model with 140 different problems did not have enough diversity and it did not generalise in the fist case. In the second evaluation the situation is similar because the training has 260 problems. To compare the strategies we consider a, uninformed strategy, that is, the combination of all planners components with the same proportion of time (Equal Time). The knowledge strategies results are better than the Equal time strategy (Cenamor, de la Rosa, and Fernández 2013). However a small training set supposes models with overfitting and that do not generalize well. And the final result (the portfolio) did no solve enough problems to improve a single planner.

For this reason, we decided to increase the number of problems and domains regardless the repeated domains to create another configuration.

Second Settings

In this setting, we consider all the problems in sequential satisfating track in the three last competitions (IPC5-6-7) and the problems in the two last competitions in learning track. In this case, for training all the problem in the IPC-5 and IPC-6 and for test all in the IPC-7. The dataset has 9480 from 790 problems from 25 domains for training. Increasing the number of instances the result model obtain better performance and generalize better than the previous setting. In addition, we created new features to improve the characterization of the problem: the differences between initial states and more information about the problem (97 features in total) are included. Hence, in this configuration, the data preparation phase is more elaborated than the previous one. We could think that the results are lightly worse than in the first setting. However, we believe that the portfolio created in this setting would generalize better to a significantly different test set.

Third Settings

When creating portfolios, to maintain planning diversity is very important because the drawbacks of a planner can be solved by the strengths of the others. However, in the last competition, many planners obtained similar performance, so they are not relevant because we need good planners that provide diversity.

We decide what planners are relevant with a Pareto efficiency technique (Censor 1977) between the quality of the best solution found and the time (in seconds). [1]. The chosen planners receive the same running time, obtaining good performance compared with the previous results, as will be shown in the experiments reported below.

The following step is focused on finding the relevant features, because in data mining, a big group of features does not mean a good characterization. In some cases, a smaller and better informed group is enough to model the data and extract the relevant information. The result of this part is promising because it improves the previous configuration.

[1]Initial study was performed with the 27 planners of IPC-7. In addition, we included LPG-tn (Gerevini et al. 2004).

Other Settings

We configure the system to next planning competition in different tracks(IPC-8). The first track is sequential satisfating, where the only restriction is the time limit (1800 seconds equal than the previous configurations). In this track, the better configuration found (selected planner and features) is the Equal Time Pareto Dominance with LPG-tn planner. In Sequential agile track, the time is 300 seconds. In this case, the base strategy is a sorted list with the medium time to find the first solution. These results came from a statistical study where we obtained the medium time to find the first solution in the selected planners. In this case, the base strategy in sequential satisfating is not enough because the slice is too small (25 seconds per planner) and in the most of the cases, the planners spend most of the time in the preprocess phase, and nor even start the search procedure. The second configuration is based on classification models with the learning phase with the solution in this time (300 seconds). In Sequential multi-core track, the time limit is 1800 seconds but we dispose 4 cores to run the portfolio. The base strategy is the same than in sequential satisfating but in 4 different threads (three planner per each CPU). The second strategy is the result of the classification model running the planners simultaneously.

In the learning track, the time limit is 900 seconds and the training data is specific per domain. The first configuration is the result of learning a classification model and the other configuration is the result of learning the regression model. The last configuration uses the two learning models, one to decide the planners and the second to assign the runtime.

Preliminary Results

In this section, the results of different settings of this research are reported. The evaluation domains are all in the sequential satisfating track in IPC-7. The first column is Equal Time (ET), that consists in assigning the same time to the 27 initial planners, previously mentioned. The second column (C47) is the result of applying the classification model with the first configuration. This model is a decision tree created by WEKA (Witten and Frank 2005) (j48 algorithm) with accuracy 88.75%, and the selected planners are the five with the best confidence in the classification model. This results are extended in a previous work (Cenamor, de la Rosa, and Fernández 2013).

In the third column (ETP) appears the dominance sequential satisfating planners with the same time. In the next columns (C96 and C35), the dominance appears and include LPG-tn planner. The C96 column is the result of the dominance with IPC5-6 and learning tracks for training. The classification model in this case is a decision tree (j48) with 94.63% coverage and the number of planners used is ten. We evaluate many classification algorithms as reported in previous works (Roberts and Howe 2009). The C35 column is the result of the same group of training but it includes the feature selection, the classification model is a decision tree (j48) with 94.82% coverage and the selected planner are the five with the best confidence in the classification model. In all cases the models are the models with best performance.

This results are compared with the winner of the last competition in sequential satisficing track (lama-2011).

	General		Pareto			
	ET	C47	ETP	C96	C35	LAMA
barman	20	20	20	20	20	20
elevators	20	20	20	20	20	20
floortile	8	8	9	7	**11**	6
nomystery	15	**17**	16	**17**	**17**	10
openstacks	20	20	20	20	20	20
parcprinter	20	20	20	20	20	20
parking	12	**20**	14	17	19	**20**
pegsol	20	20	20	20	20	20
scanalyzer	18	18	17	17	18	**20**
sokoban	17	**19**	18	18	**19**	**19**
tibybot	16	19	17	17	18	19
transport	**20**	19	**20**	19	**20**	16
visitall	20	20	20	20	20	20
woodworking	20	20	20	20	20	20
Total	246	260	251	252	**262**	250

Table 1: Preliminary results in the different configurations

In this research we look for a technique to combine planners and improve the result of a single one. The theoretical limit is the number of problems that can be solved by at least one base planner. In the case of the general strategies, such limit is 266 problems because, in some domains all problems are solved by one planner, but in domains like tidybot, nomystery and sokoban, only 19 problems can be solved. In the case of the floortile domain, only 9 problems are solved. The maximum solved problem is increased because LPG-tn solved more problem in floortile domain (3 more problems). In the column C35, the limit is 269 problems.

The results show that there are some accurate configurations and that the performance depends of the input and the characterization of the problem. However, the results indicate that the planner combination is a interesting topic and the proposed approaches are good methods to select the components. In other scope like SAT solvers, this idea have been developed with good results, and the winner of the last competition in portfolio track (Malitsky et al. 2013) use similar process with other components and different output.

Future Work

The results of this research might have different alternatives. The planner combination has better performance than a single planner in most of the cases. However the reason might be that the 80% of the problems in sequential satisfating track of the last competition are solved in less than 70 seconds. For this reason, the strategy to assign the same time for all is effective to improve the result of the winner. Similar strategy (Seipp et al. 2012) is used in optimal planning and it obtained good performance too. A challenge of portfolios resides in finding a good combination in less time, similar to the agile track in the next competition. Nowadays, there are a few restrictions in the competitions, the time is one of them (almost the most important); but there are others constraints that might change the performance of the planners (limitation in the loops, multithreading, the components, etc.). This is an area without enough research and there are exploration possibilities because the experimentation in the planning community is usually run in controllable environments. They tend to have the same configuration and this situation is not authentic in real world. There are a lot of limitations and these can modelled in a planner portfolio (running time, memory, cores number, etc.) This are the requirements explained and

Other ideas rely on fixing the planner components and finding the best strategy, limit the selected planners (output planners) or limit other resources. If it takes into account this fact, the predictive models will be more informative to just select a single planner or a reduced set. This approximation is valid for any time restriction but it is not for a multi-core approximation. In the last case, the portfolio need almost one planner per each thread. This point supposes that there is not only one approximation to configure a planning portfolio, but it is possible to create a general method to configure them.

Other line of research is including the execution information for take decisions in run-time. With the execution information it might strengthen the synergy of the planners. For example, if *planner A* did not find the solution, the *planners C, F and G* neither solved. In this scope, it is possible to create predictive models with that information, and the training data increases with each execution (where learning improves upon the plans created for execution). In addition, it is possible to combine the continuous learning with *a priori* knowledge to configure a portfolio and create a more adaptable method.

Acknowledgements

This work has been partially supported by the Spanish project TSI-090302-2011-6 and TIN2012-38079-C03-02.

References

Cenamor, I.; de la Rosa, T.; and Fernández, F. 2012. Mining ipc-2011 results. In *Proceedings of the Third Workshop on the International Planning Competition*.

Cenamor, I.; de la Rosa, T.; and Fernández, F. 2013. Learning predictive models to configure planning portfolios. In *Proceedings of the Workshop on the Planning and Learning*.

Censor, Y. 1977. Pareto optimality in multiobjective problems. *Applied Mathematics and Optimization* 4(1):41–59.

Chapman, P.; Clinton, J.; Kerber, R.; Khabaza, T.; Reinartz, T.; Shearer, C.; and Wirth, R. 2000. Crisp-dm 1.0 step-by-step data mining guide. *http:///www.crip-dm.oprg/*.

Fawcett, C.; Helmert, M.; Hoos, H.; Karpas, E.; Röger, G.; and Seipp, J. 2011. Fd-autotune: Automated configuration of fast downward. *The 2011 International Planning Competition* 31–37.

García-Olaya, Á.; Jiménez, S.; and Linares López, C. 2011. The 2011 international planning competition. *Universidad Carlos III de Madrid*.

Gerevini, A.; Saetti, A.; Serina, I.; and Toninelli, P. 2004. Lpg-td: a fully automated planner for pddl2. 2 domains. In *In Proc. of the 14th Int. Conference on Automated Planning and*

Scheduling (ICAPS-04) International Planning Competition abstracts. Citeseer.

Gerevini, A.; Saetti, A.; and Vallati, M. 2009. An automatically configurable portfolio-based planner with macro-actions: PbP. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS-09)*.

Han, J.; Kamber, M.; and Pei, J. 2006. *Data mining: concepts and techniques*. Morgan kaufmann.

Helmert, M.; Röger, G.; Seipp, J.; Karpas, E.; Hoffmann, J.; Keyder, E.; Nissim, R.; Richter, S.; and Westphal, M. 2011. Fast downward stone soup. *The 2011 International Planning Competition* 38.

Malitsky, Y.; Sabharwal, A.; Samulowitz, H.; and Sellmann, M. 2013. Algorithm portfolios based on cost-sensitive hierarchical clustering. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, 608–614. AAAI Press.

Roberts, M., and Howe, A. 2009. Learning from planner performance. *Artificial Intelligence* 173(5):536–561.

Seipp, J.; Braun, M.; Garimort, J.; and Helmert, M. 2012. Learning portfolios of automatically tuned planners. In *ICAPS*.

Witten, I. H., and Frank, E. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*. 2nd Edition, Morgan Kaufmann.