Automatic Problem Decomposition for Distributed Planning

Mark Iwen & Amol Dattatraya Mali,
EECS, University of Wisconsin, Milwaukee WI 53211
Phone: 1-414-229-6762, Fax: 1-414-229-2769, iwen2724@csd.uwm.edu, mali@miller.cs.uwm.edu
Content Areas: Distributed Planning, Graphplan

Abstract

(Appears in the proceedings of the International Conference on Artificial Intelligence (IC-AI), Volume 1, June 2002, Las Vegas, pp. 411-417.)

Distributed planning is highly useful in planning in complex real-world domains. Several advances have recently occurred in centralized plan synthesis under classical assumptions. In this paper, we show how the planning graph constructed by the influential planner Graphplan can be used to automatically and effectively decompose a problem so that the resulting subproblems can be allocated to multiple planners. We introduce interaction graphs which can be used to automatically decompose a planning problem into entirely independent subproblems under certain conditions. The resulting subproblems can be passed to efficient classical planners and their plans can be combined. We present an empirical evaluation of three two agent planners (D-FF, D-HSP and D-GP) which solve subproblems generated by the interaction graph-based decomposition. Each of the three two agent planners synthesizes plans faster than the corresponding single agent planner (FF, HSP and Graphplan) on several problems. Most of the work in distributed planning has hitherto focused on important issues like cooperation, coordination, communication and negotiation among various planning agents. Our work shows how recent advances in classical planning can be used to improve the computational efficiency of distributed planners.

1 Introduction

Distributed planning is the problem of finding a plan that helps a set of agents in a given initial configuration to collectively achieve their subgoals. In most distributed planners, multiple agents synthesize individual plans which are merged to get globally correct plan. The plan merging process involves resolution of conflicts. Importance of distributed planning is clear from the DARPA initiative air battle management program [Baker & Greenwood 1987], air campaign planning [Wilkins & Myers 1998], and the DARPA initiative pilot's associate program. A brief review of various distributed planning techniques is given in [Mali & Kambhampati 2002].

Most of the research in distributed planning has focused on important issues of coordination, communication, cooperation and negotiation among various agents. This is clear from various articles [Durfee 1999; Grosz et al 1999; desJardins et al 1999; desJardins & Wolverton 1999; Tambe & Jung 1999] in the special issue of AI Magazine on distributed planning. Research in centralized planning has focused on reducing plan synthesis times and improving quality of plans. Several advances have occurred in an efficient synthesis of plans in centralized fashion (using only one agent) under classical assumptions in last seven years. Our work is motivated by our long term goal of extending/adapting recent advances in classical planning to distributed planning in domains involving metric time, resource quantities, quantifiers and conditional effects.

An important and perhaps the first step in solving a distributed planning problem is to decompose the given problem. Currently there are no general and automatic techniques to effectively decompose planning problems into subproblems with limited interactions. Effective automatic decomposition is very important since it can reduce overall plan synthesis time. Most distributed planners use a manually specified decomposition. The ones that do a fully automatic decomposition don't do it efficiently. Given a conjunctive goal like $(g_1 \wedge g_2 \wedge ... \wedge g_n)$, these planners usually simply divide it into n subgoals if there are n planning agents. These planners do not consider interactions among subgoals. Some planners do a semi-automatic decomposition using knowledge specified by humans. For example, if different floors of a building are identified by a human as different regions, electrical and plumbing tasks on different floors are assigned to different planners. Similarly, if different rooms are identified as different non-interacting or minimally interacting regions, tasks to be carried out inside different rooms are assigned to different planners. One of the recent advances in

classical planning is the Graphplan planner [Blum & Furst 1997] which uses a compact representation called "planning graph". This graph can be constructed in low order polynomial time. In this paper, we show how information from planning graph can be used to automatically split a given problem efficiently. We introduce interaction graphs which can be used to achieve a highly effective decomposition in certain domains. An advantage of an interaction graph over a planning graph is that interaction graph can also allow splitting of the initial state of a planning problem and that it splits a problem into entirely independent subproblems under certain conditions, eliminating the cost of resolving conflicts among plans of individual agents. This paper makes the following contributions:

- We show how information from planning graph can be used to efficiently decompose goal and the set of actions so that various subgoals and actions can be allocated to various planning agents.
- We show how a distributed planner can use the planning graph-based decomposition to efficiently synthesize plans without losing soundness and completeness.
- We show how information from planning graph can be used to reduce the cost of resolving conflicts among plans of various agents at the time of merging them.
- We show how interaction graphs can be used to decompose problems.
- We show via empirical evaluation that interaction graph-based decomposition allows three simple distributed planners D-GP, D-FF and D-HSP to synthesize plans faster than the corresponding centralized planners Graphplan, Fast Forward planner [Hoffman 2001] and Heuristic Search Planner [Bonet & Geffner 2001].
- Our work shows that distributed planners can exploit recent advances in classical planning in at least two ways: (i) by using efficient classical planners and (ii) by using effective automatic problem decomposition techniques.

2 Background

As in most classical planners, we use STRIPS representation of actions. Predicates like $on(A, B)$ can be considered as propositions by rewriting them as $onAB$. An action (e.g. move(A,B,C)) is a ground instance of an operator (e.g. move(x,y,z)). Capitalized letters in arguments of an action are specific objects and lowercase letters are variables. o_i denotes an action. A planning problem is specified as $< I, G, O >$, where I denotes completely specified initial state, G denotes goal and O denotes the set of all actions in domain. We assume that I contains only true propositions. O_i denotes the set of actions that agent i can use to synthesize its individual plan. Clearly, $O_i \subseteq O$. I_i and G_i respectively denote initial world state and conjunctive goal of planning problem of agent i. n denotes number of agents doing planning. $A_i, i \in [1, n]$ denotes an agent. $(G_1 \wedge G_2 \wedge ..... \wedge G_n) = G$. FSS means forward state space. PG stands for planning graph. IG stands for an interaction graph.

Graphplan: Graphplan works in 2 phases. The first involves growing a PG and is called the PG construction phase. This is a forward phase, beginning with the initial state. The second phase is the solution extraction phase. This is backward search phase starting with the goal. PG has two kinds of levels called **action levels** and **proposition levels**. The 0th proposition level is the same as the initial state. In general, the i th proposition level is immediately succeeded by the i th action level. And, the i th action level immediately precedes $(i + 1)$ th proposition level.

The i th action level in the PG contains all actions whose all pre-conditions appear in the i th proposition level. There is also a dummy action called a **no-op**, **maintenance action**, or **persistence action** in the i th action level for each proposition in the i th proposition level. The pre-condition and effect of this action is the proposition for which the action was created. This action is included in the PG because if no action changing the truth of the proposition occurs, the truth of the proposition remains same. The $(i+1)$ th proposition level is the union of the i th proposition level and the effects of the actions in the i th action level. LA_i denotes the set of actions in i th action level in the PG for the problem $< I, G, O >$.

A key to the efficiency of Graphplan is the inference of binary **mutex** (mutually exclusive) relations. Two kinds of mutexes are found: (i) mutexes between actions, and (ii) mutexes between propositions. The PG is grown by including an action level and a proposition level if no solution is found in it by backward search and the backward search is repeated. A PG is said to **level off** if the none of the following change when the PG is grown further: (i) the number of actions in last action level, (ii) the number of propositions in last proposition level, (iii) the number of pairs of actions that are mutex in last action level, and (iv) the number of pairs of mutex propositions in last proposition level. The PG in Fig. 1 has 3 proposition levels and 2 action levels. M1, M2, M3 and M4 are actions in the domain. The pre-conditions and effects of these actions are shown on the left and right sides of the boxes respectively. The goal is achievable with the plan Step 0: M2, Step 1: M1 & M3.

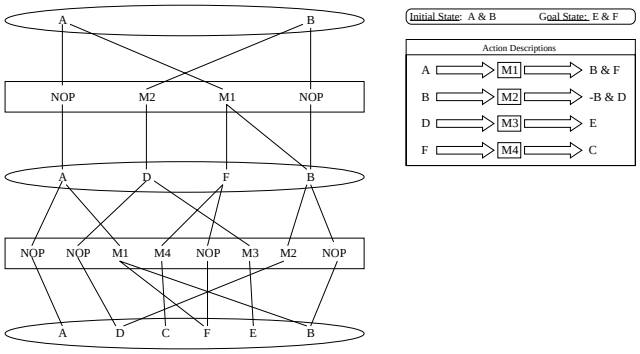

Figure 1: A planning graph

3 PGs for Problem Decomposition

One can make following valid observations about a PG.

(1) A PG of k action levels for $< I, G, O >$ contains all actions that occur in a plan of k steps for any planning problem whose initial state is same as I.

(2) If an action o_i does not occur in j th action level of PG for $< I, G, O >$, it cannot occur at any step q, $0 \leq q, q \leq j$ in any plan for any planning problem whose initial state is same as I.

(3) A world state which contains $(p \wedge q)$ cannot be achieved by any action sequence of k steps starting from I, if p and q are mutex in k th proposition level of the PG for $< I, G, O >$.

(4) If a proposition p does not appear in k th proposition level in the PG for $< I, G, O >$, then any world state containing p cannot be achieved by any plan of k steps starting from I.

(5) If the PG is grown until its j th action level is same as $(j+1)$ th action level, then actions that do not appear in its j th action level are not relevant to solving the planning problem. Note that such a PG is not same as a leveled off PG.

(6) If the PG for $< I, G, O >$ is grown until its i th proposition level is same as $(i-1)$ th proposition level, any proposition p that does not appear in $(i-1)$ th proposition level can never be made true, starting from I. As a result any world state or goal containing p is not achievable from I.

Observations 3 and 4 have been reported in [Nguyen & Kambhampati 2000]. Our problem decomposition strategies below are based on the six properties of PG stated above.

Guideline 1: If a conjunctive goal G is split into n conjunctive subgoals $G_1, G_2, G_3 \ldots G_n$, then agent A_i, $i \in [1, n]$ can be given actions that appear in the action level j of the PG for $< I, G_i, O >$ whose $(j+1)$ th proposition level contains all subgoals from G_i such that no two of them are mutex in this proposition level. Note that this guideline is a heuristic because if there is a ternary mutex among subgoals in G_i, some actions needed to solve the problem of reaching G_i from I may not be present in the planning graph whose $(j+1)$ th proposition level does not contain any mutex between two subgoals from G_i. Such missing actions can be provided to the agent later when it fails to find plan with actions from such a planning graph.

Guideline 2: If $\cup_{i=1}^{n} O_i = O_G$, where O_G is the set of actions in last action level j of PG such that $LA_j = LA_{j-1}$, then each action relevant to solving the planning problem $< I, G, O >$ can be included in the individual plan of some agent.

Note that if actions from the set O_G rather than O are used, some irrelevant actions may be eliminated from consideration, since $O_G \subseteq O$. Guidelines 1 and 2 show how PGs can be used to split set of only relevant actions and assign them to agents such that **(i)** each agent A_i has all the actions needed to achieve its goal G_i and **(ii)** each action relevant to achieving G is available to some agent for inclusion in its individual plan.

Note that PG for $< I, G, O >$ contains more information than individual PGs for problems $< I, G_i, O_i >$, $i \in [1, n]$. Information from the PG for the problem $< I, G, O >$ can reduce effort in resolving conflicts between individual plans. Conflicts that exist between plans are: (i) mutex actions, and, (ii) subgoals not achieved (or deleted). While merging individual plans, conflicts are resolved by (i) removing actions and/or (ii) reordering actions and/or (iii) including actions. Algorithms for merging plans are reported in [Foulser et al 1992]. Note that if an action o_x does not appear in the PG for $< I, G, O >$ at k th action level, it cannot appear in global plan at step $j, j \in [0, k]$. Thus while reordering actions from individual plans at the time of plan merging, o_x should not be placed at any step $j, j \in [0, k]$. In particular, information from PG for $< I, G, O >$ allows conflict resolver to identify useless reorderings, avoiding some failures. The conflict resolver can introduce actions from O_G that do not appear in individual plans to avoid loss of completeness. The leveled off PG has more information than the PG for which $LA_j = LA_{j-1}$ and this can be used in efficient conflict resolution. For example, actions that are mutex in the last action level of the leveled off PG are always mutex for given I and O. These should not be included at the same step in global plan.

Assuming that the n individual plans are merged only after all of them are generated in parallel, the total planning time will be maximum of the individual

planning times plus the plan merging time. This can be reduced by splitting G so as to reduce individual planning times and cost of plan merging. The differences between the indices of the first proposition levels in the PG for $< I, G, O >$ in which various subgoals of various agents appear, and, the indices of the first proposition levels in which they occur such that no two of them are mutex, provides a very useful information for splitting a problem. For example, consider the goal $(a \wedge b \wedge c)$ which is split so that a is assigned to A_1 and $(b \wedge c)$ is assigned to A_2, such that a, b and c are propositions. Let the indices of the first proposition levels in the PG for $< I, (a \wedge b \wedge c), O >$ in which a, b and c occur be 4,5 and 6 respectively. Let the indices of the first proposition levels in this PG in which all subgoals from the three pairs $(a, b), (b, c)$ and (a, c) occur such that no subgoals in same pair are mutex be 20, 5 and 6 respectively. Let the length of shortest plan for this problem be 22. The individual plans may need a significant modification at the time of plan merging in this case where goal is split into a and $(b \wedge c)$. In this problem, $(a \wedge b)$ and c may be a better split. Higher order mutexes (ternary, quaternary etc.) can provide more useful information, though it is more expensive to find them.

4 IG for Problem Decomposition

We denote a graph by $< V, E >$ where V is the set of vertices in the graph and E is the set of edges. An IG is an undirected, simple bipartite graph. Decomposition based on this graph is obtained by detecting if it is disconnected and finding its connected components if it is disconnected. A graph is connected if there is a path to reach any of its vertices from any other vertex.

An IG for the problem $< I, G, O >$ is constructed in the following manner: Vertices are created for each proposition true in initial state and each proposition needed true in goal. Two vertices v_i and v_j are connected only if the corresponding propositions have a common primary object such that the proposition in v_i belongs to initial state and the proposition in v_j belongs to goal. Vehicles, cities, airports and other locations in transportation logistics are secondary objects. We assume that the planning domain contains both primary objects and secondary objects and that each proposition from goal contains at least one primary object. In general, we assume that whether an object is primary or secondary is specified. In blocks world and transportation logistics domain, objects have limited types (blocks, table, plane, truck, city, airport location etc.) and all predicates describing relations between objects are unary or binary. We have an implemented heuristic which correctly finds all secondary objects in these two domains using the restricted nature of relationships in these domains. We assume that types of objects like block, package, truck, location etc. are given under a separate category in problem specification and that they are not a part of initial state. So vertices for $package(P_1), package(P_2)$, $Truck(T_1), Truck(T_2), Airport(Heathrow)$ etc. do not appear in an IG. The reason for ignoring these from the IG is to keep the number of edges low. Propositions from I which contain only secondary objects are not represented in the IG. Example of an IG is given in Fig. 2. Blocks are primary objects and table is the secondary object. Note that the size of an IG depends only on I and G. It is not affected by O at all.

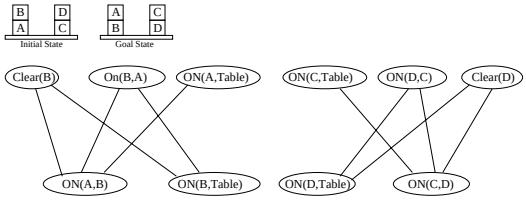

Figure 2: Disconnected IG. The IG has 2 components.

There is a polynomial time algorithm (pg. 274, [Deo 1999]) for testing if a graph is disconnected and finding all of its connected components if it is disconnected. We have implemented this to find the components of an IG. Different planning problems corresponding to different components of the IG (when it is disconnected) can be allocated to different agents. For plans of individual agents to be free of conflicts, whether there are shared resources must be considered. We consider a plane to be one kind of resource, truck to be another kind of resource etc. We have the following guideline on the decomposition of a problem into p independent subproblems:

Guideline 3: If an IG of a planning problem $< I, G, O >$ has p connected components and there are at least p resources of each kind, the planning problem can be split into p independent subproblems $< I_1, G_1, O_1 >$, $< I_2, G_2, O_2 >$, $< I_3, G_3, O_3 >$, $...< I_p, G_p, O_p >$, such that $I = (I_1 \wedge I_2 \wedge ... \wedge I_p)$, $(G_1 \wedge G_2 \wedge G_3 \wedge ... \wedge G_p) = G$ and $O_1 \subset O, O_2 \subset O, ..., O_p \subset O$, if any one object of each resource type is enough for an agent i to generate its individual plan, irrespective of the state of the resource object in I_i.

Let us consider the example in Fig. 2. IG yields the following problems $< I_1, G_1, O_1 >$ and $< I_2, G_2, O_2 >$. $I_1 = (clear(B) \wedge on(B, A) \wedge on(A, Table))$ and $G_1 = (on(A, B) \wedge on(B, Table))$. $I_2 = (clear(D) \wedge on(D, C) \wedge on(C, Table))$ and $G_2 = (on(C, D) \wedge on(D, Table))$. O_1

is set of all ground instances of $move(x, y, z)$, such that $x \neq Table, x \neq y, y \neq z, x \neq z, x, y, z \in \{A, B, Table\}$. O_2 is set of all ground instances of $move(x, y, z)$, such that $x \neq Table, x \neq y, y \neq z, x \neq z, x, y, z \in \{C, D, Table\}$. Note that an action involving primary objects of both agents like $move(A, B, D)$ need not be considered.

4.1 D-FF, D-HSP & D-GP

The FF planner is a forward state-space planner using a heuristic based on relaxed planning graph. This planner won outstanding performance award at the AIPS-2000 planning competition. The HSP 2.0 planner is a state-space planner computing estimates of distance to goal from various world states. This planner did very well at the AIPS-1998 and 2000 competitions. It allows a very flexible experimentation through choice of direction of search, heuristic and weight value to be used in computing path costs. The planners D-FF, D-GP and D-HSP work in the following manner. Planner X is FF or HSP or GP: **(i)** Construct an IG for the given problem. If the IG is connected, perform breadth-first FSS search until a state I' is found such that the IG for $< I', G, O >$ is disconnected. In this search, an IG is constructed for each world state visited. Visits to same world state are avoided. **(ii)** Form two subproblems, each based on half the total number of components of the IG. If there are an odd number of components n, then one subproblem is formed using ceiling of $\frac{n}{2}$ components and the other subproblem is formed using the remaining components. Thus the IG is considered to have only two components. Proceed to step (iii) only if restrictions on resources stated in guideline 3 in section 4 are fulfilled. **(iii)** Solve each subproblem using planner X independently. **(iv)** Combine the two plans to get part of global plan. Append this part of the global plan to the path in tree of FSS planner that lead from I to I', to get final plan.

4.2 Relaxing the Assumptions

In section 4.1, we showed how independent subproblems can be derived from an IG under certain assumptions about known nature of objects, I, G and resource availability. The notion of constructing a graph containing vertices for various propositions from I and G and connecting a vertex from I to a vertex from G if there is one or more common object between these two is useful even if the assumptions in section 4.1 do not hold. One can still construct such a graph and form planning subproblems based on its components. These problems can be given to various agents and their plans could be merged. Some agents may not be able to generate plans achieving their subgoal if they do not have a required action. Such partial plans can be completed. At the time of merging plans, the conflict resolver can introduce actions from O_G obtained from the PG for $< I, G, O >$ to find a globally correct plan. This avoids loss of completeness.

5 Empirical Evaluation

Our empirical results are shown in Fig. 3, 4 and 5. Several classical planners are publicly available for experimentation. There are no such distributed planners to the best of our knowledge. The performance of D-FF, D-HSP and D-GP could have been compared with such planners. Our breadth-first FSS Code in C, FF version 2.2, our implementation of Graphplan in C++ and HSP 2.0 with h1plus heuristic, forward direction and weight of 2 were used in these experiments. The experiments were conducted on a Dual Intel Pentium II 400 MHz SunOS 5.7 machine. The times for D-FF, D-HSP and D-GP are found by adding (i) FSS search time, (ii) time for construction of IGs and testing for their disconnectedness, (iii) maximum of planning times of the two agents, and, (iv) plan merging time. Since the IG and FSS search-based decomposition lead to independent subproblems, no conflicts between plans of two agents had to be resolved. The times reported are cpu seconds, except those stated to be in cpu miliseconds (ms). - denotes that no speedup was obtained. The problems will be made available to other researchers. The problems with names starting with bw are from blocks world. bw-large.a is a benchmark problem. Actions in this domain are ground instances of the operator $move(x, y, z)$ where $x \neq y, y \neq z, x \neq z, x \neq Table$. This means moving block x from top of block y or table to top of block z or table. Grippers are not represented in the problem, so multiple blocks whose tops are clear can be moved at same time. Other problems are from the transportation logistics domain in which packages are to be delivered to be appropriate locations using planes and trucks. Speedup for planner D-X in Fig. 3 is found by dividing the time needed by planner X by the time needed by D-X. Speedup of planner D-X is denoted by S in the column next to the one for solving times for D-X.

We did not obtain speedup on some problems mainly because of the time needed by the FSS search to yield subproblems. The problem bw-1 was difficult to split for the automatic decomposition technique, because of being very serial. Note that we do not claim to have planners necessarily faster than HSP, FF or GP. The aim of our experiments is to see if distribution of planning activity can improve performance of HSP, FF and GP. On several problems from the AIPS-2000 compe-

tition which are very large or highly serial, the distributed planners either ran out of memory or took longer.

The maximum number of actions in search spaces of two planning subproblems (N1, N2) and in the search space of given problem (N) are reported in the table in Figure 4, along with information about objects in the problems. The maximum number of actions is found by computing the number of all ground instances based on the objects that an agent had access to. The number of objects in blocks world was same as the number of blocks. The data for transportation logistics problems shows the number of planes, cities, packages, trucks, airport locations and non-airport locations respectively. These results show that IG is very effective at reducing the maximum number of actions. Loss of plan optimality can be a significant problem in distributed planners. Global plans generated by D-GP were close to optimal plans on some problems. Number of steps and actions in optimal plans and plans generated by FF, D-FF, HSP, D-HSP, GP and D-GP are shown in Fig. 5.

6 Conclusion

Most of the research in distributed planning so far is about important issues of cooperation, communication, coordination and negotiation among agents. Problem decomposition is one of the key problems in distributed planning. Current distributed planners do not use general and effective automatic problem decomposition techniques. In this paper we showed how planning graphs and interaction graphs can be used to decompose problems effectively. Both representations can be constructed in low order polynomial time. We showed how distributed planners can use decompositions based on PG and IG. Several efficient classical planners have been developed in last seven years. We showed via empirical evaluation that the distributed versions of three such planners (HSP, FF, and, GP) were significantly benefitted by the IG and FSS search-based decomposition technique. Our work shows that distributed planning can be benefitted by recent progress in classical planning in at least two ways: (i) by using efficient classical planners and (ii) by using general and effective automatic problem decomposition techniques.

Acknowledgement: This work is supported by NSF grant IIS-0119630 to Amol Mali. The authors thank Subbarao Kambhampati, Dana Nau, Ichiro Suzuki and Dan Weld for useful comments on distributed planning.

References

[**Baker & Greenwood 1987**] T. C. Baker and J. R. Greenwood, Star: An environment for development and execution of knowledge-based planning applications, Proceedings of DARPA knowledge-based planning workshop, Dec. 1987.

[**Blum & Furst 1997**] Avrim Blum and Merrick Furst, Fast planning through planning graph analysis, Artificial Intelligence 90, 1997, 281-300.

[**Bonet & Geffner 2001**] Blai Bonet and Hector Geffner, Heuristic search planner 2.0, AI Magazine, Fall 2001, Volume 22, No. 3, pp. 77-80.

[**Deo 1999**] Narsingh Deo, Graph theory with applications to engineering and computer science, Prentice Hall, 1999.

[**Durfee 1999**] Edmund H. Durfee, Distributed continual planning for unmanned ground vehicle teams, AI Magazine, Volume 20, Number 4, Winter 1999, pp. 55-62.

[**Foulser et al 1992**] David E. Foulser, Ming Li and Qiang Yang, Theory and algorithms for plan merging, Artificial Intelligence journal, Volume 57, Number 2-3, 1992, pp. 143-182.

[**Grosz et al 1999**] Barbara J. Grosz, Luke Hunsberger and Sarit Kraus, Planning and acting together, AI Magazine, Volume 20, Numer 4, Winter 1999, pp. 23-34.

[**Hoffmann 2001**], Jorg Hoffmann, FF: The fast forward planning system, AI Magazine, Volume 22, Number 3, Fall 2001, pp. 57-62.

[**desJardins et al 1999**] Marie E. desJardins, Edmund H. Durfee, Charles L. Ortiz Jr., and Michael J. Wolverton, A survey of research in distributed continual planning, AI Magazine, Volume 20, Number 4, Winter 1999, pp. 13-22.

[**desJardins & Wolverton 1999**] Marie desJardins and Michael Wolverton, Coordinating a distributed planning system, AI Magazine, Volume 20, Number 4, Winter 1999, pp. 45-54.

[**Mali & Kambhampati 2002**] Amol Mali and Subbarao Kambhampati, Distributed Planning, To appear in the Encyclopaedia of Distributed Computing, Kluwer Academic Publishers.

[**Nguyen & Kambhampati 2000**] XuanLong Nguyen and Subbarao Kambhampati, Extracting effective and admissible heuristics from the planning graph, Proceedings of the National Conference on Artificial Intelligence (AAAI), 2000.

[**Tambe & Jung 1999**] Milind Tambe and Hyuckchul Jung, The benefits of arguing in a team, AI Magazine, Volume 20, Number 4, Winter 1999, pp. 85-92.

[**Wilkins & Myers 1998**] David E. Wilkins and Karen L. Myers, A multiagent planning architecture,

Problem	HSP	D-HSP	S	FF	D-FF	S	GP	D-GP	S
bw-1	0.11	4.25	-	0.02	4.25	-	3.481	4.29	-
bw-large.a	0.2	0.66	-	0.02	0.61	-	12.55	1.65	7.6
bw-2	0.41	0.1	4.1	0.06	0.07	-	39.22	0.43	91.2
bw-3	0.08	0.05	1.6	0.03	0.04	-	1.535	77 ms	19.9
bw-4	0.41	0.76	-	0.08	0.67	-	15.98	4.09	3.91
bw-5	$>$ 30 min.	430.46	$>$ 4.18	4.06	2.62	1.55	$>$ 30 min.	$>$ 30 min.	-
logistics-1	0.42	0.07	6	0.14	0.06	2.33	$>$ 30 min	72 ms	$>$ 25000
logistics-2	0.1	0.06	1.7	0.06	0.05	1.2	0.202	48 ms	4.21
logistics-3	1.2	0.12	10	0.3	0.09	3.33	$>$ 30 minutes	0.169	$>$ 10650
logistics-4	5.74	0.24	23.92	0.69	0.19	3.63	$>$ 30 minutes	0.342	$>$ 5263
logistics-5	26.81	2.4	11.17	6.22	1.59	3.91	$>$ 30 min.	$>$ 30 min.	-
logistics-6	199.54	6.3	31.67	53.73	2.96	18.15	$>$ 30 min.	$>$ 30 min.	-

Figure 3: Empirical results - Planning times and speedup

Problem	N1, N2	N	Objects
bw-1	48, 18	294	7
bw-large.a	294, 4	648	9
bw-2	180, 180	1584	12
bw-3	48, 48	448	8
bw-4	4, 448	900	10
bw-5	3840, 11638	57798	39
logistics-1	90, 90	1240	2,4,4,4,4,4
logistics-3	130, 130	1536	2,4,8,4,4,4
logistics-4	402, 402	5820	2,6,6,6,6,6
logistics-5	3554, 3554	49224	2, 12, 12, 12, 12, 12
logistics-6	4494, 4494	56712	2, 12, 24, 12, 12, 12

Figure 4: Maximum number of actions and the number of objects

Problem	D-GP	GP	D-FF	FF	D-HSP	HSP	Optimal
bw-1	12/9	17/9	12/9	11/11	12/9	11/11	11/9
bw-large.a	16/9	10/4	12/10	6/6	11/9	8/8	6/4
bw-2	12/6	12/6	12/6	12/12	12/6	12/12	12/6
bw-3	11/8	12/7	11/8	10/10	11/8	10/10	10/7
bw-4	17/8	16/5	18/16	15/15	16/14	12/12	11/5
logistics-1	50/12	-	42/21	44/44	50/25	51/51	32/12
logistics-2	24/11	26/10	24/17	23/23	26/19	25/25	23/10
logistics-3	58/12	-	50/25	52/52	62/31	63/63	40/12
logistics-4	50/11	-	64/32	67/67	74/37	75/75	48/11

Figure 5: Number of steps and actions in plans.

Proceedings of international conference on artificial intelligence planning systems (AIPS), 1998, pp. 154-162.