Literature Review on AI planning and search

As part of fulfillment of project 3 in AIND, Udacity

1 Background

The field of AI planning were influenced by various other subject, from state-space search to control theory, and from propositional calculus to the practical needs of robotics, scheduling. Interdisciplinary studies led to fruitful results over years. While this writing assignment is more or less an imitation of Russell and Norvig's historical notes at the end of Chapter 10 in Artificial Intelligence: A Modern Approach [1], we would redirect serious readers to some standard surveys for a more complete understanding on AI planning. For an anthology of early work, you could consult Readings in Planning [2]; for planning algorithms in the 1990s, 2 surveys by Weld [3, 4] would be your choices; for an up-to-date comprehensive summary including both classical and stochastic planning, probably LaValle's Planning Algorithms [6] would serve the purpose.

2 Early Stage Developments

STRIPS [7] designed in 1971 for planning component in a robot project has usually been credited as the first major planning system. It illustrates the interaction of influences from state-space search, theorem proving, control theory and practical needs. Its overall control structure was modeled on that of the General Problem Solver [8] published in 1961. Its use of representation language led to the rise of several successors, including Action Description Language [10], the computer-parsable Problem Domain Description Language (PDDL) [11] and the recent PDDL 3.0 extension [12] that includes plan constraints and preferences. Among these languages, PDDL defined nowadays standardized syntax for representing planning problems.

Early AI planning considered only totally ordered (a.k.a. linearly ordered) action sequences. Such planners cannot solve some very simple problems, such as the Sussman anomaly. Their incompleteness was mainly due to the overlook in challenges of interleaving. Efforts to code with the problem were seen in goal-regression planning by Warren's WARPLAN [14], a planner written in a logic programming language. However, the mainstream approach from 1970s to 1990s was to replace total-order planning with partial-order planning (POP).

3 The Age of Partial-Order Planning

The success of POP lies in its ability of detecting conflicts and protecting achieved conditions from interference. Pioneers in this direction includes NOAH planner [17, 18] and NONLIN system [15, 16]. Later, more work were done on this direction for a clear formal exposition: TWEAK [19] allowed proofs of completeness and intractability; McAllester and Rosenblitt [20] give a straightforward description of a complete partialorder planner. By the time of early 90s, SNLP [21] and UCPOP [22] already had widely distributed implementations.

Starting from mid-90s, POP planners lost popularity to state-space planners due to performance issue in large problems, some said. But in 2001, Nguyen and Kambhampati [23] shown that their POP planner REPOP equipped with accurate heuristics derived from a planning graph was competitive with the fastest state-space planners, and scaled up much better than GRAPHPLAN in parallelizable domains.

4 The Rise of State-Space Planners

By introducing an ignore-delete-list heuristic, the first notable state-space planner was McDermott's UNPOP program [24] in 1996. To name a few practical planners for large planning problems in this direction, there are Heuristic Search Planner [26] using forward search; HSPR [25] using backward search; Hoffmann and Nebel's FF [29, 30, 31], probably the most successful state-space searcher to date; and

its variant FASTDOWNWARD [32] with preprocessed action schemas and thus more explicit constraints.

Since one might consider planning as an exercise in controlling combinatorial explosion, an important class of state-space planners are those graph-planning systems. There were various usages of graph-planning toward a solution. By recording mutexes to point out where the difficult interactions, Blum and Furst's GRAPHPLAN system [33, 34] were orders of magnitude faster than the partial-order planners from 1995 to 1997. After that, more graph-planning systems were created: IPP [35], STAN [36], SGP [37], Subbarao Kambhampati [38] (the approach that our course is based on), and LPG [39, 40] (the one with local search technique, winner of the 2002 AIPS).

5 Other Influences

The approach of situation calculus was introduced by John McCarthy very early in 1963 [41]. Our course basically follows Ray Reiter's papers [42, 43]. Contribution in this approach was also made by Kautz et al. [44] on investigating various ways to propositionalize action schemas. They found that the fastest solution times did not necessarily come from most compact forms. Ernst et al. [45] had done systematic analysis for generating propositional representations automatically from PDDL problems. Later in this direction, Kautz and Selman's BLACKBOX planner [46] combined ideas from GRAPHPLAN and SATPLAN; while van Beek and Chen created CPLAN [47] based on constraint satisfaction.

There are also influences of binary decision diagram from the hardware verification community. As early as 30 years ago, Clarke et al. [48] and McMillan [49] suggested using compact data structures for Boolean expressions. In 1998, Cimatti et al. [50] presented a planner based on this approach. In 2001, Vossen et al. [51] surveyed the use of integer programming for planning.

6 Summary

Among different approaches, we are not confident enough to determind which techniques work best on which kinds of problems. A general belief, however, was pointed out in Helmert's 2001 paper [52]; that constraint-based approaches such as GRAPHPLAN and SATPLAN might be the best for NPhard domains, while search-based approaches do better in domains where feasible solutions can be found without backtracking. We also believe that cross-fertilization of ideas from the different areas would continue to improve the overall performance and make larger industrial implementation possible.

References

[1] Russell, Stuart; Norvig, Peter. Artificial Intelligence: A Modern Approach. Prentice Hall.

[2] Allen, J. F., Hendler, J., and Tate, A. (Eds.). (1990). Readings in Planning. Morgan Kaufmann.

[3] Weld, D. S. (1994). An introduction to least commitment planning. AIMag, 15(4), 27-61.

[4] Weld, D. S. (1999). Recent advances in AI planning. AIMag, 20(2), 93-122.

[5] Ghallab, M., Nau, D. S., and Traverso, P. (2004). Automated Planning: Theory and practice. Morgan Kaufmann.

[6] LaValle, S. (2006). Planning Algorithms. Cambridge University Press.

[7] Fikes, R. E. and Nilsson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. AIJ, 2(34), 189208.

[8] Newell, A. and Simon, H. A. (1961). GPS, a program that simulates human thought. In Billing, H. (Ed.), Lernende Automaten, pp. 109124. R. Oldenbourg.

[9] Bylander, T. (1992). Complexity results for serial decomposability. In AAAI-92, pp. 729734.

[10] Pednault, E. P. D. (1986). Formulating multiagent, dynamic-world problems in the classical planning framework. In Reasoning about Actions and Plans: Proc. 1986 Workshop, pp. 4782.

[11] Ghallab, M., Howe, A., Knoblock, C. A., and McDermott, D. (1998). PDDLThe planning domain definition language. Tech. rep. DCS TR-1165, Yale Center for Computational Vision and Control.

[12] Gerevini, A. and Long, D. (2005). Plan constraints and preferences in PDDL3. Tech. rep., Dept. of Electronics for Automation, University of Brescia, Italy.

[13] Waldinger, R. (1975). Achieving several goals simultaneously. In Elcock, E. W. and Michie, D. (Eds.), Machine Intelligence 8, pp. 94-138. Ellis Horwood.

[14] Warren, D. H. D. (1974). WARPLAN: A System for Generating Plans. Department of Computational Logic Memo 76, University of Edinburgh.

[15] Sacerdoti, E. D. (1975). The nonlinear nature of plans. In IJCAI-75, pp. 206-214.

[16] Sacerdoti, E. D. (1977). A Structure for Plans and Behavior. Elsevier/North-Holland.

[17] Tate, A. (1975). Using Goal Structure to Direct Search in a Problem Solver. Ph.D. thesis, University of Edinburgh.

[18] Tate, A. (1977). Generating project networks. In IJCAI-77, pp. 888-893.

[19] Chapman, D. (1987). Planning for conjunctive goals. AIJ, 32(3), 333-377.

[20] McAllester, D. A. and Rosenblitt, D. (1991). Systematic nonlinear planning. In AAAI-91, Vol. 2, pp. 634639.

[21] Soderland, S. and Weld, D. S. (1991). Evaluating nonlinear planning. Technical report TR-91-02-03, University of Washington Department of Computer Science and Engineering.

[22] Penberthy, J. S. and Weld, D. S. (1992). UCPOP: A sound, complete, partial order planner for ADL. In KR-92, pp. 103-114.

[23] Nguyen, X. and Kambhampati, S. (2001). Reviving partial order planning. In IJCAI-01, pp. 459-466.

[24] McDermott, D. (1996). A heuristic estimator for means-ends analysis in planning. In ICAPS-96, pp. 142-149.

[25] Bonet, B. and Geffner, H. (1999). Planning as heuristic search: New results. In ECP-99, pp. 360-372.

[26] Bonet, B. and Geffner, H. (2000). Planning with incomplete information as heuristic search in belief space. In ICAPS-00, pp. 52-61.

[27] Haslum, P. (2006). Improving heuristics through relaxed search – An analysis of TP4 and HSP*a in the 2004 planning competition. JAIR, 25, 233-267.

[28] Haslum, P., Bonet, B., and Geffner, H. (2005). New admissible heuristics for domain-independent planning. In AAAI-05.

[29] Hoffmann, J. (2001). FF: The fast-forward planning system. AIMag, 22(3), 57-62.

[30] Hoffmann, J. and Nebel, B. (2001). The FF planning system: Fast plan generation through heuristic search. JAIR, 14, 253-302.

[31] Hoffmann, J. (2005). Where ignoring delete lists works: Local search topology in planning benchmarks. JAIR, 24, 685-758.

[32] Helmert, M. (2006). The fast downward planning system. JAIR, 26, 191-246.

[33] Blum, A. L. and Furst, M. (1995). Fast planning through planning graph analysis. In IJCAI-95, pp. 1636-1642.

[34] Blum, A. L. and Furst, M. (1997). Fast planning through planning graph analysis. AIJ, 90(1-2), 281-300.

[35] Koehler, J., Nebel, B., Hoffmann, J., and Dimopoulos, Y. (1997). Extending planning graphs to an ADL subset. In ECP-97, pp. 273-285.

[36] Fox, M. S. and Long, D. (1998). The automatic inference of state invariants in TIM. JAIR, 9, 367-421.

[37] Weld, D. S., Anderson, C. R., and Smith, D. E. (1998). Extending graphplan to handle uncertainty and sensing actions. In AAAI-98, pp. 897-904.

[38] Bryce, D. and Kambhampati, S. (2007). A tutorial on planning graph-based reachability heuristics. AIMag, Spring, 47-83.

[39] Gerevini, A. and Serina, I. (2002). LPG: A planner based on planning graphs with action costs. In ICAPS-02, pp. 281-290.

[40] Gerevini, A. and Serina, I. (2003). Planning as propositional CSP: from walksat to local search for action graphs. Constraints, 8, 389-413.

[41] McCarthy, J. (1963). Situations, actions, and causal laws. Memo 2, Stanford University Artificial Intelligence Project.

[42] Reiter, R. (1991). The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In Lifschitz, V. (Ed.), Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy, pp. 359-380. Academic Press.

[43] Reiter, R. (2001). Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. MIT Press.

[44] Kautz, H., McAllester, D. A., and Selman, B. (1996). Encoding plans in propositional logic. In KR-96, pp. 374-384.

[45] Ernst,M.,Millstein, T., andWeld, D. S. (1997). Automatic SAT-compilation of planning problems. In IJCAI-97, pp. 1169-1176.

[46] Kautz, H. and Selman, B. (1992). Planning as satisfiability. In ECAI-92, pp. 359363.

[47] van Beek, P. and Chen, X. (1999). CPlan: A constraint programming approach to planning. In AAAI- 99, pp. 585-590.

[48] Clarke, E. and Grumberg, O. (1987). Research on automatic verification of finite-state concurrent systems. Annual Review of Computer Science, 2, 269-290.

[49] McMillan, K. L. (1993). Symbolic Model Checking. Kluwer.

[50] Cimatti, A., Roveri, M., and Traverso, P. (1998). Automatic OBDD-based generation of universal plans in non-deterministic domains. In AAAI-98, pp. 875881.

[51] Vossen, T., Ball, M., Lotem, A., and Nau, D. S. (2001). Applying integer programming to AI planning. Knowledge Engineering Review, 16, 85100.

[52] Helmert, M. (2001). On the complexity of planning in transportation domains. In ECP-01.