Constraint-Based Agents

An Architecture for Constraint-based Modeling and Local-Search-based Reasoning for Planning and Scheduling in Open and Dynamic Worlds

Alexander Nareyek

Abstract

The thesis [7] focuses on intelligent autonomous agents that can act in a goal-directed manner under real-time constraints and with incomplete knowledge, situated in a dynamic environment where resources may be restricted. To realize such agents, constraint programming, local search, AI planning and scheduling techniques are enhanced and combined. The application domain of computer games serves as an example here.

1 Introduction

Realizing intelligent behavior for autonomous agents is crucial for an endless number of applications. Action-planning capabilities are one of the main needs — of course, being computed in real time, handling the environment's dynamics and openness, and involving the resource situation. Some of these aspects have previously been investigated in isolation (e.g., see [2, 5, 6]), but this thesis is one of the very first publications that describes an integration into a single system. This cannot be accomplished by simply combining well-known techniques; many such techniques need to be substantially extended, and in many cases completely new approaches are required.

To realize a declaratively formulated and efficiently executable search for an agent's behavior plan, the constraint programming framework is applied as the basic paradigm throughout the thesis. This framework is set in a local search context to allow real-time computation and the handling of dynamics and is enhanced by structural search features to enable planning tasks to be treated. To carry out an agent's planning using these techniques, a general planning model is specified.

The application domain of this work is computer games. These fit the problem context very nicely as most of them are played in real time and provide a highly interactive environment in which environmental properties are constantly changing.

2 Local Search

Since dynamics and real-time computation must be supported, a combination of constraint programming and local search has been developed. To this end, a great deal of attention has been paid to facilitating the integration of domain-dependent knowledge. Real-world applications can rarely be tackled without domain-dependent knowledge, and the use of heuristics is crucial for success.

In our local search approach, a specific cost function is specified for each constraint (so-called *global constraints*), returning a value that represents the constraint's current inconsistency/optimality with respect to the connected variables.

For example, a simple `Sum` constraint with two variables a and b to be added and an s variable for the sum could specify its costs as $\texttt{Sum}_{costs} = |a + b - s|$.

In addition, a constraint has a number of heuristics to improve its cost function. For example, a heuristic for the `Sum` constraint could randomly choose one of the related variables and change it such that there are no more costs. Another heuristic might resolve the inconsistency by distributing the necessary change such that all variables are changed by the same (minimal) amount. The constraint must make the choice as to which heuristic to apply on its own.

On top of all constraints is a *global search control* which selects, in each iteration of local search, one of the constraints that is to perform an improvement, i.e., the transition to a neighbor state. Figure 1 shows the control flow.

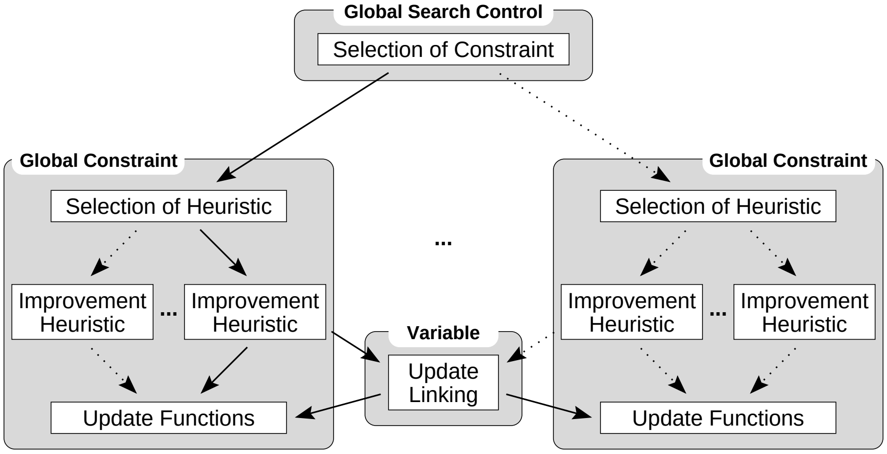

Figure 1: Using global constraints for local search

The combination of local search and constraint programming provides an important building block for our agents. Many other publications focus on problem-specific local search solutions (see [1] for examples). Improved efficiency is the main goal, generality often being disregarded. By contrast, our approach's modular structure of the constraints makes it easy to vary, reuse and extend problem descriptions. However, unlike approaches of other constraint-based local search approaches (e.g., GSAT [8] and GENET [4]), we focus on the global constraints for the solving process, which allows us to exploit domain-specific information by including constraint-specific search control and representation knowledge.

3 Structural Constraint Satisfaction

If constraint programming is to be applied to planning, we face the problem that conventional formulations for constraint satisfaction problems are too restrictive because all the elements and their relations have to be specified in advance. However, in an environment that is open and only partially observable for an agent, it is not clear which objects exist, i.e., how many variables and resources of which kind are available. The closed-world assumption cannot be applied and we cannot restrict the planning process to a given set of state variables. Furthermore, for a planning problem (and many other problem domains), numerous alternative constraint graph structures are potentially valid for realizing a solution to a problem.

The constraint programming framework is therefore enhanced by adding the ability to handle problems in which the search for a valid constraint graph structure is part of the search process. Those problems are called structural constraint satisfaction problems. A main feature of these are so-called structural constraints, which represent requirements with respect to the structure of valid graphs. Productions can be deduced from the formulation of a structural constraint satisfaction problem, allowing search to generate/modify potentially valid graph structures during the search process. The structural constraints are checked to ensure the graph's consistency. Approaches that apply conditional constraint satisfaction (e.g., Graphplan [3] and CPlan [9]), which construct a structure involving all potential plans, are not applicable here because of the open world and the unmanageable size of even small problems.

The concept is combined with the local-search approach, providing the basic mechanism to handle arbitrary structures for an agent's behavior plan, and enabling us to treat open-world problems with a potentially infinite number of objects. Since it is embedded in the general constraint programming paradigm, the framework is not restricted to a certain way of exploring the search space, e.g., according to an increasing plan length.

4 Planning

The above concepts and techniques were not specifically designed for the planning domain only; they are applicable to a whole range of other domains like configuration and design. The planning model presented in the thesis applies them to an agent's behavior planning. The model focuses on resources and also allows a limited handling of an agent's partial knowledge. The example in Figure 2 informally illustrates the planning model's basic elements, which can be represented by specific constraints and variables.

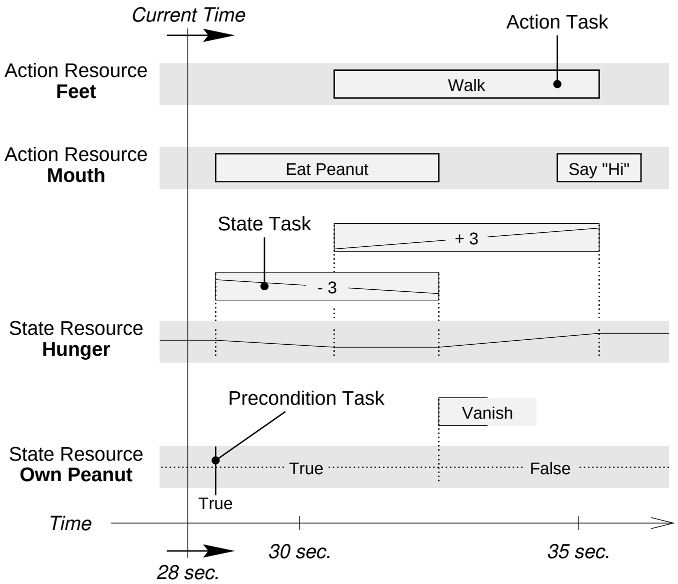

Figure 2: Visualization of the basic plan elements

In contrast to an off-line refinement search, the iterative repair/improvement steps of local search make it possible to easily interleave sensing, planning and execution. After each single repair step, which can usually be computed very quickly, the planning process can involve changes in the situation (see Figure 3).

In the thesis, the interplay of all parts – the realization of an autonomous agent – is demonstrated and some general solving heuristics are presented and applied to the

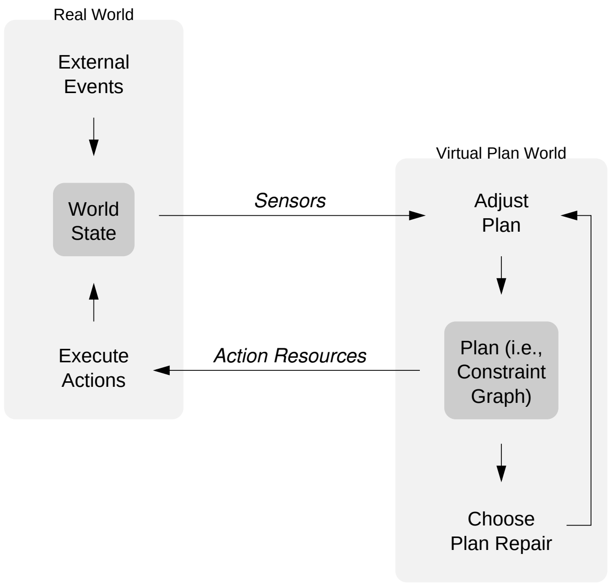

Figure 3: Interleaving sensing, planning and execution

Orc Quest problem and variations of the Logistics Domain. Real-time computation, a plan property's optimization and the handling of dynamics are dealt with.

5 Conclusion

The agent architecture presented here focuses on the computation of an agent's goal-directed behavior. By contrast, most of the existing architectures for autonomous agents focus on communication protocols and interaction, but do not specify how to compute the behavior of the individual agents. On the other hand, standard AI planning approaches do usually not incorporate real-time reasoning, dynamics, resources, optimization and an open world.

The whole architecture is embedded in a constraint programming framework, which makes the approach highly declarative and modular and the developed methods applicable to other search problems. However, several extensions of the constraint programming framework were necessary to achieve adequate efficiency and expressiveness.

To sum up, the presented agent architecture features:

- **Real-Time Reasoning:** The local-search approach enables the system to provide very primitive plans (reactions) for short-term computation horizons, while longer computation times are used to improve and optimize the agent's behavior.

- **Dynamics:** The iterative repair/improvement steps make it possible to easily interleave sensing, planning and execution.

- **Resource Focus:** The search can be conducted in a way that focuses on the satisfaction and optimization of resource-related properties.

- **Incomplete Knowledge:** The model is not restricted to the closed-world assumption and enables an agent's incomplete knowledge to be dealt with.

- **Domain Knowledge:** Even though the model supports fully domain-independent planning, domain-specific heuristic knowledge can easily be integrated by means of the global constraints.

- **Software Engineering:** The system is based on the general search framework of constraint programming and can easily be changed, extended, maintained and reused.

The thesis was done in the context of the EXCALIBUR project, which focuses on the computer-games domain. More information on the EXCALIBUR project is available at:

`http://www.ai-center.com/projects/excalibur/`

References

[1] Aarts, E. H. L., and Lenstra, J. K. eds. 1997. *Local search in Combinatorial Optimization.* Reading, Wiley-Interscience.

[2] *AI Magazine* 20(4), Special Issue on Distributed Continual Planning. 1999. American Association for Artificial Intelligence.

[3] Blum, A. L., and Furst, M. L. 1997. Fast Planning Through Planning Graph Analysis. *Artificial Intelligence* 90: 281–300.

[4] Davenport, A.; Tsang, E.; Wang, C. W.; and Zhu, K. 1994. GENET: A Connectionist Architecture for Solving Constraint Satisfaction Problems by Iterative Improvement. In Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94), 325–330.

[5] Nareyek, A. ed. 2001. *Local Search for Planning and Scheduling.* Reading, Springer LNAI 2148.

[6] Nareyek, A. ed. 2001. Proceedings of the IJCAI-01 Workshop on Planning with Resources. International Joint Conferences on Artificial Intelligence Inc.

[7] Nareyek, A. 2001. *Constraint-Based Agents – An Architecture for Constraint-Based Modeling and Local-Search-Based Reasoning for Planning and Scheduling in Open and Dynamic Worlds.* Reading, Springer LNAI 2062.

[8] Selman, B.; Levesque, H.; and Mitchell, D. 1992. A New Method for Solving Hard Satisfiability Problems. In Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92), 440–446.

[9] Van Beek, P., and Chen, X. 1999. CPlan: A Constraint Programming Approach to Planning. In Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99), 585–590.