The HTN Domain “Factories”

Malte Sönnichsen,1 Dominik Schreiber2
Karlsruhe Institute of Technology
1malte@soennichsen.xyz 2dominik.schreiber@kit.edu

Introduction
In this paper we present the benchmark domain “Factories” for Hierarchical Task Network (HTN) planning written in the HDDL format (Höller et al. 2020).

In this benchmark domain, the planning objective is to construct a factory by satisfying a dependency graph on certain resources while using trucks to deliver the needed resources from one place to another. As such, our domain is a crossover of traditional logistics and transport domains with a producer-consumer problem.

Overview
An input problem consists of \( f \) factories, \( d \) dependencies, and \( t \) trucks. Each factory requires certain resources in order to be built, which are the factory’s dependencies, and is able to produce certain resources when built. As such, the problem’s factories depend on each another: The construction of some factory \( F \) may require the construction of other factories which produce the resources necessary to construct \( F \). This renders the problem recursive and makes the choice of a hierarchical planning model natural. The set of dependencies can be seen as a Directed Acyclic Graph (DAG) induced by the required and produced resources of each factory.

Transporting resources from a producer to a consumer requires trucks. There can be multiple trucks in the problem, so the tactical choice of which truck should do the delivery can lead to varying plan lengths.

In order to avoid complex conditions, we introduced composite resources which are fused from two other resources. This way, a factory has at most one resource demand in the HDDL while the number of needed actual resources can be arbitrarily high. Also, for the sake of simplicity, the resources a certain factory requires for its construction and for producing a resource itself are identical.

Generating Problems
Our problem generator script features three inputs which are (i) the number of factories, (ii) the maximum number of resource dependencies per factory, and (iii) the number of trucks. The graph of locations is generated as a simple undirected circle of \( n \) locations with up to \( n \) additional random edges, where \( n \) is set to twice the number of factories and trucks. Trucks are placed randomly in the graph. Dependencies are introduced to the problem by iterating over the factories “from left to right” and repeatedly adding dependencies of the current factory to a resource that some factory to the left produces. This ensures that the dependency graph is in fact acyclic and can be ordered. The first factory has no resources and the final factory is the destination object to be constructed.

Properties
The Factories domain is recursive, i.e., it may feature task \( T \) as a possible subtask in a method for task \( T \). The domain is totally ordered.

The domain makes use of positive and negative method preconditions. All constraints are stated conjunctively (“STRIPS-style”), i.e., as a list of literals.

We think the Factories domain is an appealing bench-