# The Scanalyzer Domain: Greenhouse Logistics as a Planning Problem

**Malte Helmert**
Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Georges-Köhler-Allee 52
79110 Freiburg, Germany
helmert@informatik.uni-freiburg.de

**Hauke Lasinger**
LemnaTec GmbH
Schumanstr. 18
52146 Würselen, Germany
hauke.lasinger@lemnatec.com

## Abstract

We introduce the Scanalyzer planning domain, a domain for classical planning which models the problem of automatic greenhouse logistic management.

At its mathematical core, the Scanalyzer domain is a permutation problem with striking similarities to common search benchmarks such as Rubik's Cube or TopSpin. At the same time, it is also a real application domain, and efficient algorithms for the problem are of considerable practical interest.

The Scanalyzer domain was used as a benchmark for sequential planners at the last International Planning Competition. The competition results show that domain-independent automated planners can find solutions of comparable quality to those generated by specialized algorithms developed by domain experts, while being considerably more flexible.

## Plant Phenomics and Smarthouses

Modern sequencing methods have made it possible for researchers in biology to easily produce a wealth of information about the *genotype*, the genetic make-up, of an organism. However, genetic information is rarely an end in itself; instead, we are usually much more interested in the *phenotype*, the observable traits and characteristics of an organism, and information about the genotype is only useful in so far as it allows us to make predictions about the phenotype.

For example, one of the most important goals of plant research is to increase the yield and resilience of crops. Modern genetics gives us the tools to selectively breed crops with unprecedented accuracy. However, to relate the performance of plants to their genetic characteristics, we still need to actually grow, observe and carefully analyze them. The field of *plant phenomics* is concerned with large-scale quantitative studies of plant phenotypes (Furbank 2009; Finkel 2009).

Smart greenhouses (*smarthouses* in the following) are an important technology for this purpose. A smarthouse consists of one or more greenhouses together with imaging facilities that collect data about the plants being grown, including infrared, visible light and fluorescence imaging. To achieve high throughput, imaging must operate fully autonomously. For this purpose, plants are transported between the greenhouses and imaging facilities by a system of conveyor belts. The *Scanalyzer planning domain*, named after the LemnaTec Scanalyzer phenotyping platform for which it was conceived, models the problem of controlling these conveyor belts. This is difficult for three reasons.



Firstly, it is necessary to navigate in fairly confined spaces. Typical greenhouses are long and narrow and can require contorted travel routes due to space and money constraints. (It is cheaper to use a few long conveyor belts than a larger number of shorter segments.)

Secondly, to enable the highest possible throughput, the conveyor belts should be packed with plants to the largest possible extent, and hence one usually cannot transport a given plant or batch of plants through the greenhouse towards the imaging facility without also moving the plants located on the way. Therefore, one must simultaneously activate several conveyor belts that form a *cycle*, so that a single operation involves many simultaneous plant movements, some of them potentially undesirable.

Thirdly, smarthouses can be large. For example, the *Plant Accelerator* at the University of Adelaide, in operation since January 2010, is the largest deployment of the Scanalyzer platform to date. Designed for a throughput of 2400 plants per day, it houses more than 1 km of conveyors. Even for much smaller systems, phenotyping runs can require significant amounts of time – in one deployment up to 7 hours, of which 3 hours are due to delays caused by plant routing.

The rest of this paper is structured as follows. In the next section, we introduce (one variant of) the greenhouse logistics problem by way of example. We then explain why classical planning algorithms are a good match for it and describe the Scanalyzer domain as used at the Sixth International Planning Competition (IPC-2008). We also provide a brief comparison of (domain-independent and domain-dependent) planner performance. Finally, we discuss some generalizations and future challenges and conclude.

## The Problem

Consider Fig. 1, which shows a top-down view of the conveyor system in a small, but otherwise representative deployment of the Scanalyzer platform. This example was used as instance #3 of the Scanalyzer domain at IPC-2008. All conveyor belts only travel in one direction, as indicated by the arrows. In a quiescent state of the system, plants are only located on the horizontal conveyor belts $A$–$F$ in the

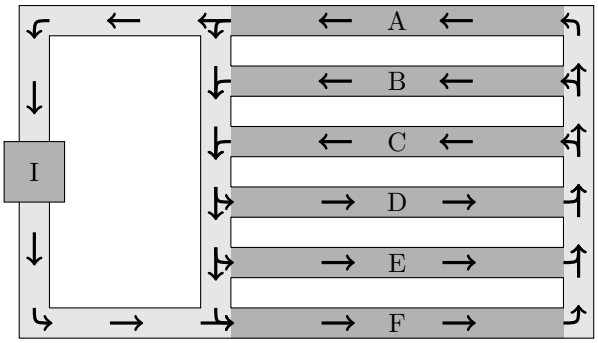


Figure 1: Example scenario. The objective is to transport all plants located on conveyor belts $A$–$F$ through the imaging chamber $I$ for analysis and to end up in a state where all plants are back at their original location.

right part of the picture, indicated by darker color. We will refer to these six belts as *segments* in the following. The objective is to transport all plants through the imaging chamber $I$ (at the left) and then return them to their original segments. Observe that with the given travel directions, only plants on segment $A$ can enter the conveyor belt leading to the imaging chamber, and plants leaving the imaging chamber must enter segment $F$ afterwards.

Each segment contains a large number of plants (say 52). There is no available free space, so it is not possible to transport plants from, say, segment $C$ to segment $E$ without simultaneously making room on segment $E$. In the problem variant we consider here, *all* plants located on a given segment must travel together, so that we can consider them as a single *batch*. An atomic operation consists of, e. g., activating segments $C$ and $E$ as well as the middle and right vertical belts to swap the contents of $C$ and $E$.

Grouping plants into batches in this way restricts the set of possible plans and may introduce plans that are more costly than what would be possible when considering individual plants, so it is desirable to remove this limitation. However, the currently used algorithms do require such grouping to keep the state space of the problem reasonably small. We will come back to this issue when discussing generalizations and challenges, but for now we treat a batch of plants that occupies a complete segment as a unit.

This means that we can represent a quiescent state of the system by denoting which of the six batches (called 1–6 in the following) is contained on which segment, together with information that denotes which batches have been analyzed by the imaging chamber already. We write such states as 6-tuples of the form $\langle a, b, c, d, e, f \rangle$, where each entry denotes which batch is contained on the corresponding segment, and batches that have been analyzed are marked with a star. Using this notation, the initial state of the example is $\langle 1, 2, 3, 4, 5, 6 \rangle$ and the goal state is $\langle 1^*, 2^*, 3^*, 4^*, 5^*, 6^* \rangle$.

In the example, there are ten atomic operations: we can swap the contents of any of the segments $A$, $B$, $C$ with any of the segments $D$, $E$, $F$ without traveling through the imaging chamber (a *rotate* operation), and we can swap the contents of segments $A$ and $F$ while transporting the contents of segment $A$ through the imaging chamber (a *rotate-and-analyze* operation). For example, three of the ten successor states of the initial state are $\langle 1, 5, 3, 4, 2, 6 \rangle$ (via operation *rotate*$(B, E)$), $\langle 6, 2, 3, 4, 5, 1 \rangle$ (via *rotate*$(A, F)$) and $\langle 6, 2, 3, 4, 5, 1^* \rangle$ (via *rotate-and-analyze*$(A, F)$).

We remark that allowing more complicated atomic operations, as well as interleaving certain operations, appears intuitively feasible. For example, one might want to execute operations like *rotate-and-analyze*$(A, F)$ and *rotate*$(C, D)$ concurrently. However, limitations of the sensors and actuators that control the conveyor belts usually preclude such levels of sophistication, so that the problem can indeed be adequately modeled as purely sequential, using only the described atomic operations.

Another practical observation is that traveling through the imaging chamber is significantly more expensive than the other operations, and hence the IPC-2008 domain formulation assigns a higher cost to *rotate-and-analyze* operations than to *rotate*. The domain-specific solver currently used by LemnaTec, however, assumes that all operations have the same cost. For this particular conveyor system the distinction does not matter because redundant *rotate-and-analyze* steps can be replaced by corresponding *rotate* steps.

We conclude our discussion of the example scenario by remarking that an optimal solution consists of 6 *rotate-and-analyze* and 8 *rotate* operations.

## The Scanalyzer PDDL Domain

For several reasons, classical planning systems appear to be a good match for the Scanalyzer planning problem. Firstly, the (often limiting) classical assumptions of a fully observable, static world with a single planning agent and deterministic state transitions are adequate for the problem. Unforeseen changes to the environment or goals are rare enough for offline planning to be a suitable approach.

Secondly, the flexibility offered by a general planner is useful because the Scanalyzer platform is deployed in many different kinds of greenhouses with different sizes and layouts. Also, a general planner can adapt to different scenarios easily. For example, there may be a requirement to move certain plants between different parts of the greenhouse in addition to imaging them. For a general planner, this can simply be expressed by defining a suitable goal. As another example, a conveyor belt may need to be shut down for maintenance or be otherwise unavailable. A general planner can be adapted to such a situation by simply removing some of the available operators.

Thirdly, the algorithms employed by current classical planners, which are mostly based on forward search in the state space, are adequate for the problem. The allowed operations in the domain are reminiscent of classical permutation puzzles like TopSpin (e. g., Yang et al. 2008), for which the best known algorithms are based on heuristic forward search. Indeed, the domain-specific solver currently used by LemnaTec is based on A$^*$ with a domain-specific heuristic.

Finally, the problem is easily expressible in the usual planning formalisms. In fact, the original problem specification as formulated by a domain expert and implemented in

```
(define (domain scanalyzer)
  (:requirements :typing :action-costs)
  (:types segment batch - object)
  (:predicates (on ?b - batch ?s - segment)
               (analyzed ?b - batch)
               (CYCLE-2 ?s1 ?s2 - segment)
               (CYCLE-2-WITH-ANALYSIS ?s1 ?s2 - segment))
  (:functions (total-cost) - number)
  (:action rotate-2
   :parameters (?s1 ?s2 - segment ?b1 ?b2 - batch)
   :precondition (and (CYCLE-2 ?s1 ?s2)
                      (on ?b1 ?s1) (on ?b2 ?s2))
   :effect (and (not (on ?b1 ?s1)) (not (on ?b2 ?s2))
                (on ?b1 ?s2) (on ?b2 ?s1)
                (increase (total-cost) 1)))
  (:action rotate-and-analyze-2
   :parameters (?s1 ?s2 - segment ?b1 ?b2 - batch)
   :precondition (and (CYCLE-2-WITH-ANALYSIS ?s1 ?s2)
                      (on ?b1 ?s1) (on ?b2 ?s2))
   :effect (and (not (on ?b1 ?s1)) (not (on ?b2 ?s2))
                (on ?b1 ?s2) (on ?b2 ?s1) (analyzed ?b1)
                (increase (total-cost) 3))))

(define (problem example) (:domain scanalyzer)
  (:objects A B C D E F - segment
            b1 b2 b3 b4 b5 b6 - batch)
  (:init (= (total-cost) 0)
         (CYCLE-2 A D) (CYCLE-2 A E) (CYCLE-2 A F)
         (CYCLE-2 B D) (CYCLE-2 B E) (CYCLE-2 B F)
         (CYCLE-2 C D) (CYCLE-2 C E) (CYCLE-2 C F)
         (CYCLE-2-WITH-ANALYSIS A F)
         (on b1 A) (on b2 B) (on b3 C)
         (on b4 D) (on b5 E) (on b6 F))
  (:goal (and (analyzed b1) (analyzed b2) (analyzed b3)
              (analyzed b4) (analyzed b5) (analyzed b6)
              (on b1 A) (on b2 B) (on b3 C)
              (on b4 D) (on b5 E) (on b6 F)))
  (:metric minimize (total-cost)))
```

Figure 2: PDDL formalization of the Scanalyzer example.

LemnaTec's planner could be faithfully converted to a planning domain in the STRIPS fragment of PDDL in less than an hour. Figure 2 shows the relevant part of the PDDL formalization of the example task as it was used at IPC-2008, with some changes in terminology for clarity. (The full domain contains some additional operators for dealing with half-size batches, as discussed later.)

## Scanalyzer at IPC-2008

At IPC-2008, the Scanalyzer domain was used as a benchmark in two tracks, *sequential satisficing* and *sequential optimization*. In the optimization track, planners were required to produce minimal cost solutions and were scored according to the number of solved tasks. In the satisficing track, suboptimal solutions were accepted, but planners were scored according to the cost of the generated solutions. Both tracks used a 30 minute timeout. The planning domains used for evaluation were unknown to the participants at the time the planners were submitted. (In fact, the Scanalyzer PDDL domain did not yet exist then.)

| Layout 1 | | | Layout 2 | | | Layout 3 | | | Layout 4 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Size** | **DSS** | **IPC** | **Size** | **DSS** | **IPC** | **Size** | **DSS** | **IPC** | **Size** | **DSS** | **IPC** |
| 6 | **18*** | **18*** | 6 | **22*** | **22*** | 6 | **26*** | **26*** | 6 | **22*** | **22*** |
| 8 | **24*** | **24*** | 8 | **30*** | **30*** | 8 | **36*** | **36*** | 8 | 32 | **30*** |
| 10 | **30*** | **30*** | 10 | **38*** | 44 | 10 | **46*** | **46*** | 10 | **40** | 44 |
| 12 | **36*** | **36*** | 12 | **46*** | 54 | 12 | **56*** | 60 | 12 | — | **56** |
| 14 | **42*** | **42*** | 14 | **54*** | 64 | 14 | **66*** | 72 | 14 | — | **66** |
| 16 | **48*** | **48*** | 16 | **62*** | 74 | 16 | — | **86** | 16 | — | **84** |
| 18 | **54*** | **54*** | 18 | — | **84** | 18 | — | **94** | 18 | — | **94** |
| 20 | **60*** | **60*** | 20 | — | **94** | 20 | — | **108** | 20 | — | **106** |
| 22 | **66*** | **66*** | 22 | — | **104** | 22 | — | **114** | 22 | — | **116** |
| 24 | **72*** | **72*** | 24 | — | **114** | 24 | — | **134** | 24 | — | **128** |

Table 1: Comparison of solution cost between the domain-specific solver (*DSS*) by LemnaTec and the best IPC-2008 result (*IPC*). Best performances shown in bold; optimal solutions marked with stars. The running example is Layout 3, size 6. IPC-2008 used sizes 6–18 of the first three layouts.

In the Scanalyzer domain, the same 30 problem instances were used in both tracks. The first 21 of these are "regular" Scanalyzer tasks with a similar flavor to our example scenario (Figs. 1 and 2). The remaining 9 tasks encode a generalized problem, discussed in the following section. The regular tasks differ in the number of segments (6–18) as well as the conveyor system layout, with all layouts suggested by domain experts from LemnaTec. For this paper, we extended this benchmark suite slightly by adding larger tasks with up to 24 segments and an additional family of conveyor layouts. Table 1 compares the performance (in terms of solution quality) of the best-performing planners at IPC-2008 to LemnaTec's domain-dependent solver on the extended benchmark suite. For both the domain-independent planners and LemnaTec's solver, memory was a more limiting factor than runtime, with the 2 GB memory limit usually exceeded within a few minutes.

The results show that while the A*-based LemnaTec solver sometimes produces better solutions (especially for Layout 2), the domain-independent planners scale to instances of larger size. In one case for Layout 4, the domain-independent systems even produce a better solution. (Optimality of the LemnaTec solver is not guaranteed despite the use of A* as it uses an inadmissible heuristic.) Note that these instances are far from trivial to solve. Blind search approaches begin to fail for the size-10 problems, which already have $3.7 \cdot 10^9$ reachable states, and are hopeless for the size-12 problems with $2.0 \cdot 10^{12}$ reachable states.

## Generalizations and Challenges

Smarthouse logistics offers many challenges beyond what we discussed so far. Most importantly, there is considerable interest in removing the restriction that all plants on a segment must be moved together as a single batch, which is severely limiting for two reasons.

Firstly, for tasks where the objective is to image all plants and then return them to their origin, the best batched solutions may be more expensive than the best solutions that do not use batching.

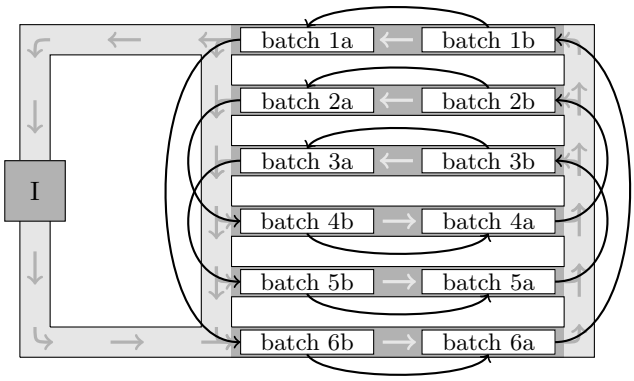


Figure 3: Example scenario with half-segment batches: transport all batches to the imaging chamber $I$ and then to the goal positions indicated by the curved arrows.

Secondly, many relevant planning tasks cannot be modeled at the abstraction level of full-segment batches. For example, in a common scenario the plants in the greenhouse need to be imaged and then moved to different locations from their original ones, according to some rotation schedule that ensures that plants reside in different areas of the greenhouse over time, to ensure even growing conditions. With full-segment batches, this goal cannot be properly expressed because plants initially located at the front of a segment will remain at the front of a segment in every quiescent state. One solution for this deficiency is to split segments into several virtual subsegments and group plants into batches at the level of these smaller subsegments.

Figure 3 shows an example of this, where each batch occupies one half of a segment. At the abstract level, we can still represent the state space of such a problem as a $k$-tuple that indicates which batches are located where and which batches have been analyzed. For example, in the scenario shown above the initial state could be represented as $\langle 1\mathrm{a}, 1\mathrm{b}, 2\mathrm{a}, 2\mathrm{b}, 3\mathrm{a}, 3\mathrm{b}, 4\mathrm{b}, 4\mathrm{a}, 5\mathrm{b}, 5\mathrm{a}, 6\mathrm{b}, 6\mathrm{a}\rangle$ and the goal as $\langle 1\mathrm{b}^*, 6\mathrm{a}^*, 2\mathrm{b}^*, 4\mathrm{a}^*, 3\mathrm{b}^*, 5\mathrm{a}^*, 2\mathrm{a}^*, 4\mathrm{b}^*, 3\mathrm{a}^*, 5\mathrm{b}^*, 1\mathrm{a}^*, 6\mathrm{b}^*\rangle$.

The main difference to the earlier problem instances is that operators now rotate the contents of four tuple positions, rather than just swapping two positions. For example, the operation *rotate*$(A, F)$ applied to a state $s = \langle s_1, \ldots, s_{12}\rangle$ would permute the first two and last two tuple entries, leading to state $\langle s_2, s_{12}, s_3, \ldots, s_{10}, s_1, s_{11}\rangle$.

While the domain-specific solver employed by LemnaTec cannot currently express problem instances of this kind, they are easy to formalize in PDDL; all that is needed, compared to Fig. 2, is to add operators *rotate-4* and *rotate-and-analyze-4* for performing such larger rotations, together with predicates *CYCLE-4* and *CYCLE-WITH-ANALYSIS-4* to encode the topology of the conveyor system.

The IPC-2008 formulation of the Scanalyzer domain contains these generalizations, and benchmark instances #22-30 used at the competition encode half-segment problems similar to this example, with three different kinds of conveyor layouts and 4–12 half-segment batches. However, the participating planners scaled much worse on these instances than on the regular ones. Only two systems could solve the problem instance shown in Fig. 3 (#30 in the benchmark suite), and then only with plans of very poor quality (cost 161, compared to a supposed optimal cost of 55). While this is still significantly better than the performance of blind search, which does not scale to any of the instances with 12 half-segments, solving half-segment problems of substantial size remains a significant challenge.

If half-segment problems are not yet hard enough, the representation can of course be refined further, up to the extreme of planning at the level of individual plants. With each segment having room for up to 52 plants, the state space of the example 6-segment problem would then contain $312! \cdot 2^{312} = 1.8 \cdot 10^{738}$ states, which certainly requires significantly more insight to find reasonable solutions. Moreover, at this level the assumption of modeling the problem as purely sequential would need to be revisited, since it is clearly desirable to have more than one plant use the vertical conveyor belts connecting the segments at the same time.

## Conclusion

We have introduced the *Scanalyzer* planning domain, which models planning problems that arise in the context of managing *smarthouses*, smart greenhouses used for large-scale automated experiments in plant phenomics.

The characteristics of these planning problems make them an attractive target for classical planners. The Scanalyzer domain has been used as a benchmark at IPC-2008, where the best-performing planners have produced very notable results. Compared to domain-specific solvers, general planners have the advantage of being more flexible in several dimensions as well as being available as off-the-shelf tools.

Due to its nature as a real application problem with a well-defined combinatorial structure, easily expressible in PDDL, we believe that the Scanalyzer domain is of interest for practically and theoretically minded researchers alike, in addition to being a suitable benchmark for domain-independent classical planners. Practically relevant scaling challenges offer an interesting area for future research.

## Acknowledgments

Many thanks to Robert Mattmüller, who helped with the Scanalyzer PDDL formalization, implemented the problem generator and produced the reference plans for IPC-2008.

This work was partly supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS). For more information, see `http://www.avacs.org/`.

## References

Finkel, E. 2009. With 'phenomics,' plant scientists hope to shift breeding into overdrive. *Science* 325:380–381.

Furbank, R. T., ed. 2009. *Plant Phenomics*, volume 36 (10–11) of *Functional Plant Biology*. CSIRO Publishing.

Yang, F.; Culberson, J.; Holte, R.; Zahavi, U.; and Felner, A. 2008. A general theory of additive state space abstractions. *JAIR* 32:631–662.