Hybrid Hierarchical Optimization for TFT-LCD Manufacturing Supply Chain

A Master's Thesis Research Plan

Esly Wadan Chou

National Tsing Hua University (NTHU)

Date: 2025-08-21

Advisor: Professor Jerry Chou

Table of Contents

---

1. Problem Context & Motivation

The TFT-LCD manufacturing industry is characterized by multi-billion dollar facilities and a complex product mix including TVs, monitors, and handheld devices. The production process is a multi-stage flow from glass input to the final module, as illustrated below.

TFT-LCD Manufacturing Process Flow

Glass Input → Array (Front) → Module (Back)

Current approaches to optimizing the supply chain face significant challenges:

- **Monolithic models** become computationally intractable due to the problem's scale and complexity.
- **Decomposed approaches** often lose critical dependencies and lead to sub-optimal solutions.
- **Commercial solutions** typically lack the flexibility to adapt to changing business needs.

---

2. Research Questions

Core Question: Can hierarchical optimization effectively coordinate enterprise planning and site scheduling in TFT-LCD manufacturing to achieve superior performance?

Sub-Questions:

1. How can the problem be decomposed into a hierarchy while maintaining solution quality?
2. What information should be communicated between the hierarchical levels to ensure effective coordination?
3. Can a surrogate model, such as a neural network, accelerate decisions for critical resources at the bottom level of the hierarchy?

---

3. Proposed Hierarchical Framework

The research proposes a three-level hierarchical architecture to address the complexity of TFT-LCD manufacturing optimization. Each level uses a different optimization technique tailored to its specific scope and purpose.

- **TOP: Enterprise Planning:** This level uses a **Genetic Algorithm (GA)** or **Mixed-Integer Programming (MIP)** to make high-level decisions, such as product allocation to different sites and financial goals.
- **MIDDLE: Site Scheduling:** This level receives targets from the top level and uses a **Constraint Programming (CP-SAT)** solver to generate detailed schedules for individual sites (e.g., Frontend and Backend).
- **BOTTOM: Resource Model:** This level provides fast feedback to the middle level by using a **Neural Network** as a surrogate model to predict critical resource performance, such as completion times.

The flow of information is hierarchical, with targets flowing down from the top and performance feedback flowing back up from the bottom.

4. Methodology & Approach

Top Level: Enterprise Planning

The top level is a strategic planning model that allocates products across multiple sites.

- **Decision Variables:**
  - Xij: The quantity of product j allocated to site i.
  - Yjk: The production timing of product j in period k.
- **Objectives:** Minimize cost and penalties, or maximize revenue minus cost.
- **Method:** A Genetic Algorithm with a population of 20 strategies, 50 generations, a uniform crossover rate of 0.8, and a Gaussian mutation rate of 0.1.
- **Scale:** The model is tested on a problem with 2 sites, 3 products, and 50 jobs.

Middle Level: Site Scheduling

The middle level creates a detailed schedule for each site based on the allocation from the top level.

- **Model:** A Constraint Programming model.
- **Variables:**
  - startjm: The start time of job j on machine m.
  - endjm: The end time of job j on machine m.
- **Constraints:**
  - **Precedence:** endjm≤startj(m+1).
  - **No-overlap:** Jobs cannot overlap on the same machine.
  - **Capacity:** Resource limits must be respected.
- **Objective:** Minimize the makespan (total completion time).
- **Solver:** Google OR-Tools CP-SAT with a time limit of 60 seconds.

Bottom Level: Resource Surrogate

This level serves as a fast predictor to support decisions at the middle level.

- **Purpose:** To quickly predict completion times and feasibility.
- **Input Features:** Job type, resource load, and queue length.
- **Output:** Predicted completion time and feasibility.
- **Model:** A Neural Network with an input layer of 5 nodes, two hidden layers of 10 nodes each, and a single output node.
- **Training:** The network will be trained on 1,000 historical data samples with an 80/20 train/validation split.

---

5. Coordination Mechanism

The hierarchical framework operates through an iterative process to converge on a high-quality solution.

1. **Top-Down Allocation:** The top level allocates products to sites.
2. **Independent Scheduling:** Each site independently creates its detailed schedule.
3. **Performance Prediction:** The resource surrogate model predicts performance metrics.
4. **Feedback Loop:** Metrics such as makespan, cost, and resource utilization are fed back to the top level.
5. **Re-allocation:** The top level updates its product allocation based on the feedback.

This process repeats for a maximum of 10 iterations, or until the objective function improvement is less than 1% (convergence).

6. Implementation & Timeline

Implementation Stack

- **Programming Language:** Python 3.10+
- **Key Libraries:**
    - **OR-Tools:** For the CP scheduling solver.
    - **PyMOO:** For the Genetic Algorithm implementation.
    - **Scikit-learn:** For the Neural Network surrogate model.
    - **NumPy/Pandas:** For efficient data handling.
    - **Matplotlib:** For visualization of results.
- **Code Structure:** The code will be organized into a modular structure:
    - `/src/top_level` (GA implementation)
    - `/src/middle_level` (CP models)
    - `/src/bottom_level` (Surrogate)
    - `/src/coordination` (Integration)
    - `/src/utils` (Helpers)

Project Timeline

- **Month 1-2: Foundation:** Literature review, problem formulation, and tool selection.
- **Month 2-3: Implementation:** Development of the top, middle, and bottom level models.
- **Month 3-4: Integration:** Implementation of the coordination protocol, testing, debugging, and running experiments.
- **Month 4-5: Documentation:** Thesis writing and results analysis.
- **Month 6: Defense:** Final preparation and thesis defense.

7. Validation, Contributions, and Deliverables

Validation Approach

The proposed framework will be validated using several test cases of varying sizes and compared against established baselines.

- **Test Cases:**
    1. **Toy:** 2 products, 2 sites, 10 jobs.
    2. **Small:** 3 products, 2 sites, 20 jobs.
    3. **Medium:** 3 products, 2 sites, 50 jobs.
- **Comparison Baselines:**
    1. Sequential approach (no coordination).
    2. Monolithic CP model (if tractable).
    3. Greedy heuristics.
- **Metrics:** Solution quality (makespan), computation time, and convergence iterations.

Expected Outcomes

The research aims to demonstrate significant improvements over current methods.

- **Performance Targets:**
    - **Makespan Reduction:** Expected improvement of 20%.

    - **Computation Speed:** Expected to be 3x faster than monolithic models.
    - **Solution Quality:** Expected to be within 90% of the optimal solution.
- **Specific Benefits:**
    - The framework will be tractable for problems with 50+ jobs.
    - It will provide a solution in under 5 minutes.
    - The architecture will be scalable and the decisions will be interpretable.

Research Contributions and Deliverables

- **Working Prototype:** A documented and tested Python codebase of approximately 3,000 lines, containerized with Docker.
- **Master's Thesis:** A 60-80 page thesis detailing the research, including chapters on the introduction, literature review, methodology, implementation, results, and conclusion.
- **Conference Paper (optional):** A paper targeting a regional conference, focusing on the novel methodology.

8. Risk Mitigation

- **Risk: CP solver is too slow.**
    - **Mitigation:** Set time limits and use pre-computation techniques.
- **Risk: Poor coordination between levels.**
    - **Mitigation:** Use a fixed number of iterations and a fallback method.
- **Risk: Surrogate model produces errors.**
    - **Mitigation:** Use a linear model as a backup to the neural network.
- **Risk: Implementation bugs.**
    - **Mitigation:** Conduct unit testing and use small toy problems for debugging.

---

9. Summary and Next Steps

This research plan proposes a hybrid hierarchical optimization framework to solve a real-world industrial problem in TFT-LCD manufacturing. The framework leverages a combination of Genetic Algorithms, Constraint Programming, and a Neural Network surrogate to achieve a scalable and practical solution. The project has a clear timeline, evaluation metrics, and mitigation strategies for identified risks.

Next Steps:

- Approval of the research plan.
- Begin the literature review.
- Set up the development environment.

---

10. Contact Information

Questions & Discussion

Contact:

- esly.wadan@gapp.nthu.edu.tw

Appendix

Backup: Mathematical Formulation

- Top Level (MIP):
  $\min\sum(cost_{ij}\times X_{ij})+penalty$
  $s.t. \sum X_{ij}\geq demand_j$
  $\sum X_{ij}\leq capacity_i$
- Middle Level (CP):
  $\min makespan$
  $s.t. start_{jm}+proc_{jm}=end_{jm}$
  $end_{jm}\leq start_{j(m+1)}$
  no_overlap(jobs, machines)
- Bottom Level (Surrogate):
  $\hat{y}=f(x;\theta)$
  where f is a neural network and θ are the learned parameters.

Backup: Preliminary Experiments

Toy Problem Results (10 jobs, 2 sites):

Method	Makespan	Time (s)
Monolithic CP	45	120
Sequential	52	15
Hierarchical	47	25

Observations:

- The hierarchical approach achieved a 10% improvement in makespan over the sequential method.
- It was approximately 5 times faster than the monolithic CP approach.

- The framework converged in 3-5 iterations for this problem size.