Reinforcement learning in logistics: Optimizing transshipments in dynamic spare parts networks

Willem van Jaarsveld - w.l.v.jaarsveld@tue.nl
Headlines

- RL powered software agent AlphaGo beats top player Lee Sedol (Silver et al, 2016)
- AlphaZero taught itself from scratch how to master Chess, Go & Shogi (2017)
1. Reinforcement Learning (RL) in logistics: Justified excitement vs hyperbole
2. RL for spare parts networks
3. Conclusion
Reinforcement learning

- Studies software agents that learn good strategies from interaction with an environment.
- Agents learn/train by **millions of interactions** with simulated environment:

  ```
  State → Go Simulator → Action
  Reward: Win (1)/Loss (0) → Software agent
  ````

  and generalize beyond situations encountered in simulation (e.g. using neural net)
RL (e.g. AlphaGo/ AlphaZero):
- Tries **tens of millions** of actions before mastering an environment.
- Uses unambiguous reward signal (win/lose) of games during training.

Logistics and supply chain:
- Nature of tasks changes over time: only some $10 - 1000$ actions relevant to specific task.
- Difficult to determine reward.
Modern RL handles **challenging and dynamic environments** that traditional optimization fails to get a grip on.

Many logistics and supply chain problems are dynamic and challenging

Approach:

1. Design simulator $\approx$ single-player **serious game** for training the AI.
2. Develop and train agent for serious game
3. Iterate game rules until it accurately represents real-life challenge
4. Use agent to make real decisions
Consider manufacturer of **high-tech equipment**
  - Examples could be wafer-steppers, IGT systems, . . .

Costly equipment downtime

Good after-sales service is key
  - Uptime guarantees, e.g. 97% *uptime* quarterly

Spare parts availability!
Worldwide spare parts network
Focus: Expensive parts that fail rarely and randomly (e.g. a component/module).

Business environment:
- Capacitated supply $\Rightarrow$ limited available inventory
- Part failures and resupply causes inventory imbalance
- Transshipments restore balance

Why RL?
- Manual is labor intensive ($> 10000$ SKUs).
- High degree of automation already present.
- Traditional techniques don’t cut it.
Formulation as serious game

Single spare part type. Information requirements:

▶ Network: customers (○), strategic (▲) and forward warehouses (▼)
▶ MTBF + installed base data → part failure frequency per customer.
▶ Supplier production capacity (parts/year)

Dynamics:

▶ Upon part failure at customer (❌), supply part as quickly as possible.
▶ Resupply to main warehouse (NL).
▶ After any event: Evaluate transshipment need.
Handle part failure
Handle part failure
Evaluate transshipment need
Evaluate transshipment need
Evaluate transshipment need
No further transshipments
Evaluate transshipment need
Rewards

- Reward = Minus Cost.
- Regular (proactive) transshipment cost (warehouse to warehouse)
  - e.g. flat handling fee + fee/km (3pl cost structure)
- Emergency (reactive) transshipment cost (warehouse to customer)
  - Similar to reactive, but $8\times$ more expensive.
Rewards

- Reward = Minus Cost.
- Regular (proactive) transshipment cost (warehouse to warehouse)
  - e.g. flat handling fee + fee/km (3pl cost structure)
- Emergency (reactive) transshipment cost (warehouse to customer)
  - Similar to reactive, but 8× more expensive.
- Emergency shipping **time** (WH to Customer)
  1. Any warehouse at trucking distance (~ 1 − 6 hrs)
  2. ▲ on same continent (~ 12 − 36 hrs)
  3. ▼ on same continent, or / ▲ on different continent (~ 48 − 72 hrs)

Frequency of flight connections.
Costs of downtime

- Example: Customer was promised 97% uptime / quarter as part of contract.
- **Loss of goodwill** of OEM depends on customer performance relative to target:
  - Above target (🟢) ⇒ substantial loss (4000/hr).
  - Below target (🔴) ⇒ extreme loss (16000/hr).
- Performance status subject to change!
Cust contract status changed - evaluate transshipment need
Algorithm

1. Rule based (as benchmark)
2. Monte-Carlo tree search:
   - Simplified for highly uncertain environment (shallow trees).
   - Optimized: Parallel implementation, variance reduction, etc.
   - Acknowledgement: Remco Dijkman, Yingqian Zhang (both TU/e), Joep van den Bogaert (ex TU/e, JADS)
3. Agent that plays perfectly (small instances only).
Results: annual costs per year

<table>
<thead>
<tr>
<th></th>
<th>Full environment</th>
<th>Smaller environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule based (benchmark)</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Monte Carlo tree search</td>
<td>73%</td>
<td>75%</td>
</tr>
<tr>
<td>Optimal</td>
<td>?</td>
<td>63%</td>
</tr>
</tbody>
</table>

RL can reduce (in game) costs with $\sim 27\%$ compared to rule-based. Room for further improvement.

---

$^1$Comp. time maxed at 10s/action on single node
Discussion

- Proof of concept of RL feasibility in part networks.
- Further steps (environment):
  - Detailed customer contract status (time remaining & actual uptime %)
  - Failure during replenishment (for frequently failing parts).
  - Condition monitoring: Dynamic failure probabilities.
- Algorithm: combine MC tree search with neural network to boost performance.
Conclusion

- Training RL agents directly from L&SC data is at present infeasible
- Use of serious games that represent practical challenge is promising
  - Domain knowledge is essential when applying RL/AI in logistics!
- RL handling of dynamic problems is excellent (unlike traditional optimization).
  - Cf. van de Bogaert et al. (2019).
Always interested to hear feedback: w.l.v.jaarsveld@tue.nl