Travel Time Optimization for Public Utility Job Scheduling

Jacob Feldman, IntelEngine, CTO
(732) 287-1531 feldman@ilog.com

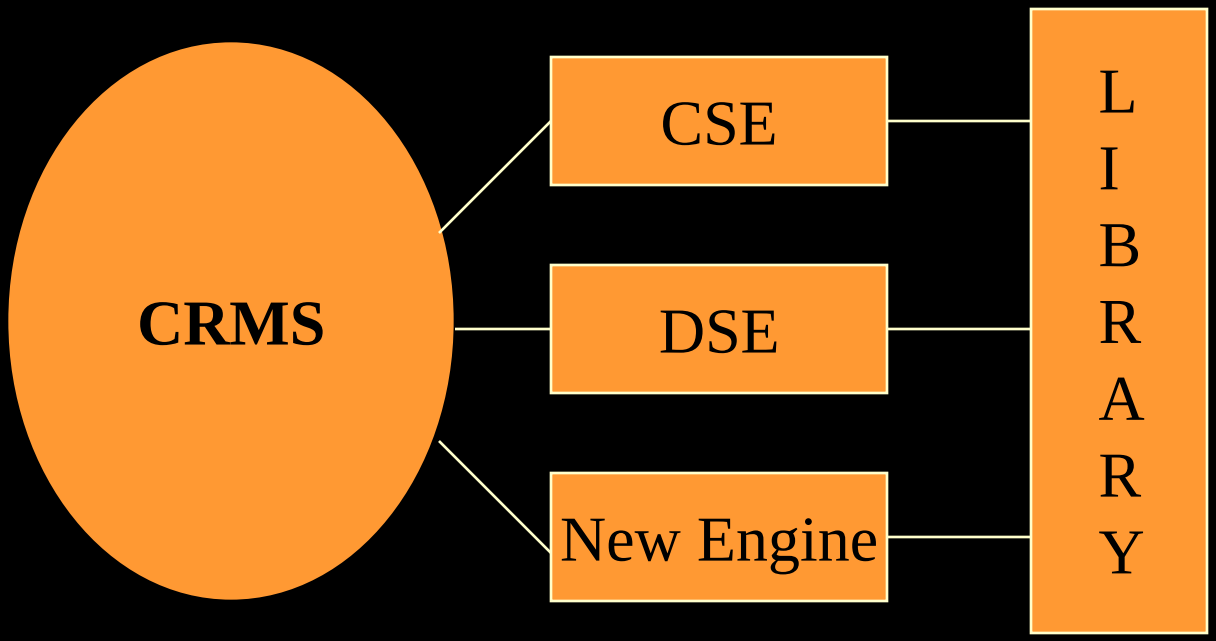
Nicholas P. Sekas, LILCO
(516) 391-6729 nsekas@lilco.com

LILCO Resource Management System

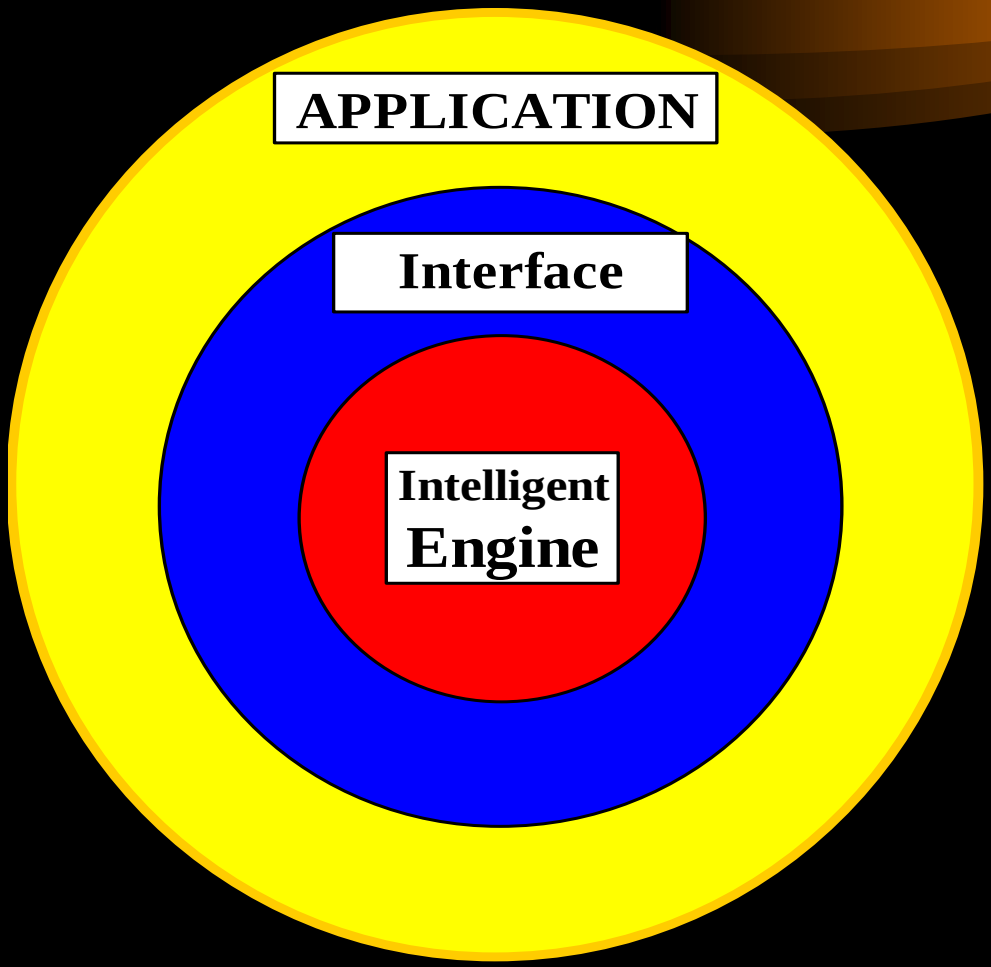
- More than 1 million customers
- More than 5000 employees
- Service territory 1,230 square miles.
- Hundreds jobs per day
- Job requires a mix of people skills, vehicles and equipment

Family of Scheduling Engines

- Construction/Designer scheduling engines
- Library Public Utility Scheduling Engines (PULSE)



Application, Interface, Engine (Pattern "Engine")



Interface - Abstract Classes

- Each interface class specifies only virtual accessors & modifiers (no data members). Sample:

```
class Designer {
    public:
    virtual Skill* getSkill() const = 0;
    virtual void assignJob(Job* job) = 0;
    virtual int getSelectionCost() = 0;
    …..
};
```

Application: Concrete SubClasses of the Interface Classes

```
class lilcoDesigner : public Designer {
    public:
    Skill* getSkill() const { return _skill; }
    void assignJob(Job* job);
    private:
    lilcoSkill* _skill;
    .....
};
```

Engine: Concrete ILOG-aware Classes

```
class ilogDesigner  {
   public:
   IlcBool  does(ilogJob* job);   // may fail
   ....
   private:
   Designer*              _designer;
   IlcUnaryResource  _resource;
   ....
};
```

Pattern "Weighted Selection"

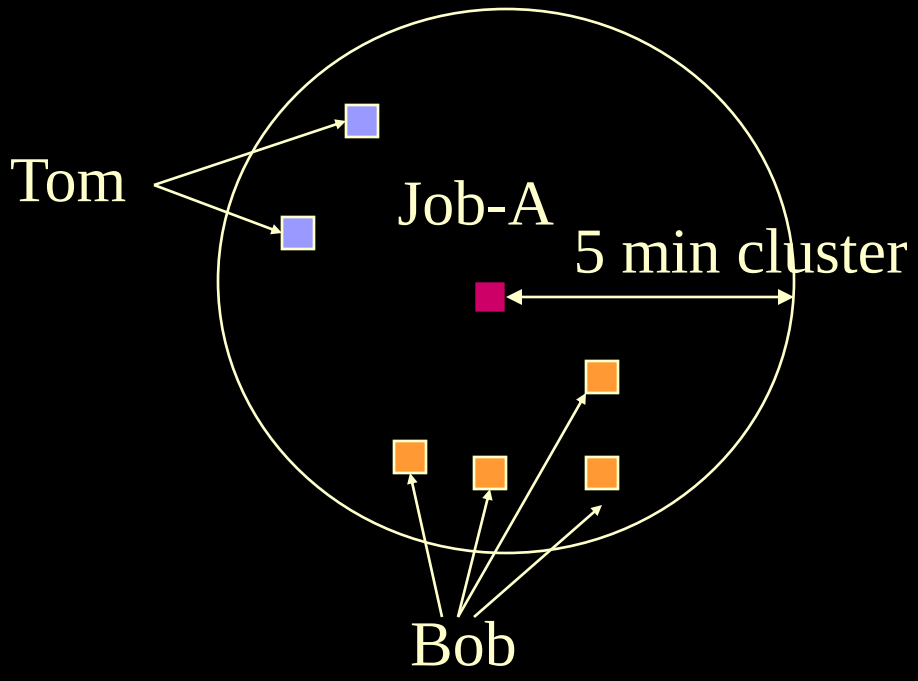
- Job selection cost
  - Weight of Job priority
  - Weight of LFD
- Resource selection cost
  - Weight of travel
  - Weight of skill matching, etc.
- Configuration files and run-time parameters

Switching Scheduling Strategies

- Parameterized strategies
  - Assign Resources to Jobs
    - Select the next most important job
    - Choose the cheapest resource for this job
  - Assign Jobs to Resources
    - For the latest scheduled crew of resources find the most important job this crew can do next
- Use the pattern “Strategy”

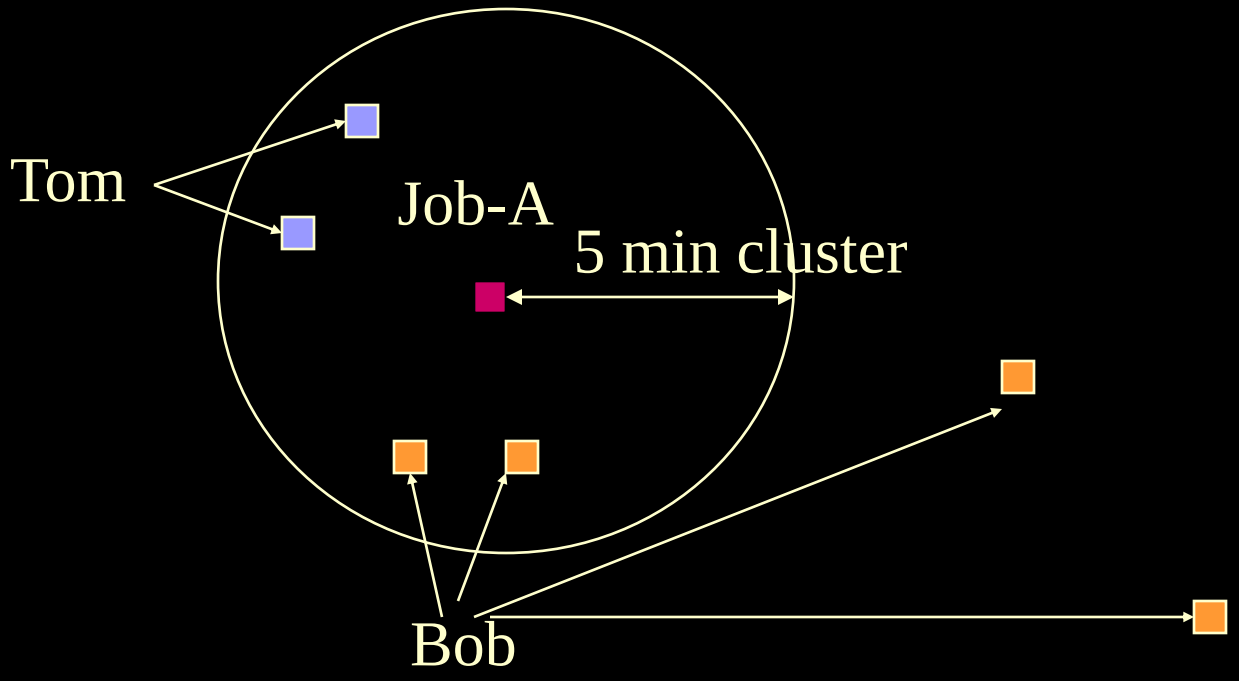
Resources Inside Job Clusters

- Prefer a designer with max number of "nearby" jobs (hours)



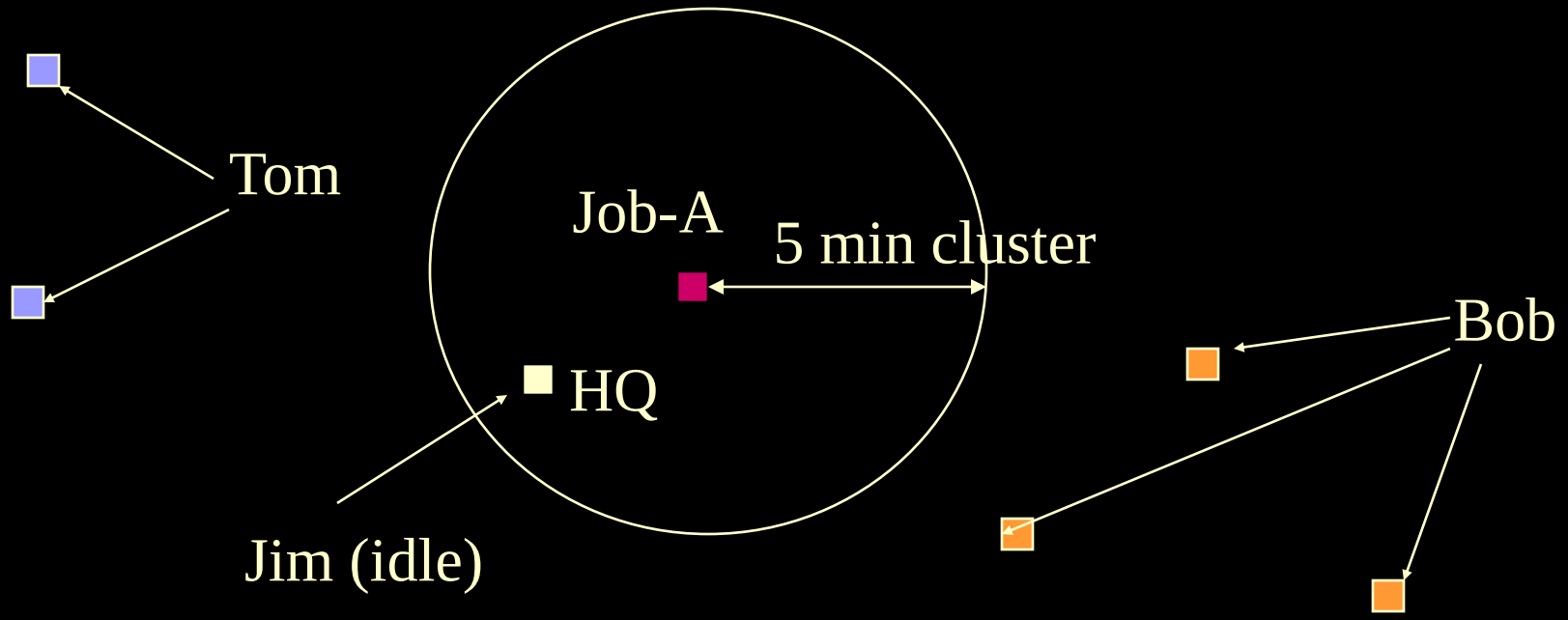
Resources Outside Job Clusters

- "Outside time" is a tiebreaker for designers with the same "inside time"



Idle Resources

- Idle resources are sitting in headquarters
- “Idle time” is an another tiebreaker



Unavailability Tolerance

- Actual job duration is far different from the original estimate
- First, the Engine tries to allocate 100% available resources
- If fails, it tries again tolerating partial unavailability but no more the a certain percent of the job duration

Resource Levelization and Travel

- Question: utilize All resources at 60% or utilize only 60% of all resources at 100%?
- Initially, it serves as a good tiebreaker for resources with the same travel cost
- Later, when resources are more utilized, may lead to too many new clusters
- Tweaking weight of levelization vs. weight of travel

Skill Matching and Travel

- Skills are hierarchical:
  Skill-C can be replaced by Skill-B
  Skill-B can be replaced by Skill-A
  A has a qualification distance 2 from C
- Replacement C by A will increase the designer selection cost by

  2 * WEIGHT_OF_SKILL_MATCHING
- Tweaking weight of skill matching vs. weight of travel

Minimize Travel Vs. Start ASAP

- If the “closest” designer is busy during week-1 and ASAP-strategy is set, the Engine will assign another designer.
- If the TRAVEL-strategy is set, the Engine will assign the “closest” designer but will schedule start date to week-2

Incremental Clustering

Strategy "Assign Jobs to Resources"

- Daily job clustering
  - Assign a crew to a job with small duration
  - What else this crew can do nearby today
- Resource pattern hierarchy
  - Use over-staffed crews for nearby jobs
  - Use this strategy for jobs far away from HQ
- Solving travelling salesperson problem

User Involvement

- Scheduling parameters defined via GUI
- What-if analysis
  - change weights and re-run the Engine
  - set frozen assignments and re-run the Engine
- Self-explanatory Engines
  - Engine Logs
  - Help from ILOG?

Dynamic Weighting

- Provocative conclusion: each data set requires a customized scheduling strategy?!
- Computing scheduling metrics up front
  - Dynamic selection of scheduling weights
  - Dynamic selections of scheduling strategies

Conclusion

- Consider not one but a Family of multi-objective intelligent engines
- Keep users involved during the entire system life-cycle.
- Use common ILOG design patterns