

**SCHEDULING ENGINE**  
**FOR**  
**LONG ISLAND LIGHTING COMPANY (LILCO)**

**Jacob Feldman**  
Technical Director  
Solution Dynamics Inc.  
981 US Hwy. 22, Suite 2000  
Bridgewater, NJ 08807  
(908) 725-5445

**Alex A. Hoyos**  
Sr. Systems Specialist  
Long Island Lighting Co.  
175 East Old Country Road  
Hicksville, NY 11801  
(516) 545-5204

**Nicholas P. Sekas**  
Sr. Management Consultant  
Long Island Lighting Co.  
100 East Old Country Road  
Hicksville, NY 11801  
(516) 545-3288

**Didier Vergamini**  
Technical Director  
ILOG, Inc.  
2105 Landings Drive  
Mountain View, CA 94043  
(415) 390-9000

# SCHEDULING ENGINE

## FOR

### LONG ISLAND LIGHTING COMPANY (LILCO)

ILOG constraint programming tools have been used to develop a scheduling engine (SE) for LILCO's Corporate Resource Management System (CRMS). The SE assigns work crews (i.e. resources: made up from thousands of humans, equipment, and vehicles) to various utility construction and maintenance jobs (hundreds per day). In short, the SE develops a "smart" work schedule given a limited number of resources while respecting user-defined scheduling rules and preferences. The SE builds teams by considering: skill and equipment requirements, availability, possible overtime allowances, travel time between job locations, employee-to-employee and driver-to-vehicle preferences. It includes special constraints for team splitting, resource replacements, reservation of resources for emergency jobs, back-up jobs for high priority jobs, and more. A schedule could be based on user-defined optimization criteria: schedule jobs as soon as possible, maximize number of scheduled jobs and/or hours, utilize specific resources, minimize travel time, etc. The SE could assign different teams to multi-shift jobs, verify and honor manual resource and date/time assignments, place jobs into jeopardy when they are missing one or more resources, and provide recommendations for the better scheduling. The system was developed under Windows 3.1 and ported to UNIX.

## Introduction

Imagine a small company that has 50 jobs to complete during one week. This represents the backlog. This company has 20 employees to do this work as well as other resources such as vehicles and equipment. How do you produce a daily schedule for the work force so that it gradually chips away at the backlog each work day? What jobs do you do first? Who does what job? In what order do you perform the jobs? Jim prefers to work with Joe, but never with Rob. Should we take this into consideration together with the fact that Bob prefers to drive his truck only. What happens if the original work scope changes? Are you scheduling efficiently? How do you evaluate your performance?

Today's management has direct control of scheduling. Scheduling involves intelligently selecting which jobs to work first while maximizing the utilization of all of your resources. Working efficiently means working smarter, not necessarily harder. Smart scheduling can help improve efficiency and lower cost. Just think of the last time you saw a group of workers sitting idle on a job only because they did not have the appropriate tools, equipment, or material.

And now, let's switch from the hypothetical example above to the real-world problems that Long Island Lighting Company (LILCO) management faces every day: several *thousand* limited *resources* made up from employees, equipment, and vehicles must be assigned to several *thousand* *jobs* with durations ranging from two hours to two months while respecting different scheduling rules and preferences!

ILOG Solver/Schedule libraries were selected as a foundation for the Scheduling Engine (SE) that is now the core of LILCO's Corporate Resource Management System (CRMS). CRMS allows the scheduling analyst to first schedule the jobs that bring the Company the most value. Given LILCO's huge backlog, we should know exactly what we need to perform each one of these jobs. Once we know this and a job's relative "value" to the Company, we can produce the final product of CRMS: the schedule. Employees are the most critical resource to the Company. Taking into consideration a continuously changing job backlog and resource availability, CRMS must frequently perform its rescheduling process. This paper will highlight the many things that the SE considers or "thinks about" before it produces a schedule. Read on!

## **LILCO Scheduling**

### **Terms and Concepts**

The SE considers different business requirements and defines them as constraints when it generates a schedule. Below is a brief list of LILCO-specific terms that represent some of these constraints that will be mentioned in this paper:

Area: The SE uses "areas" to identify individual pools of resources and jobs for scheduling (employees, vehicles, and equipment). In general, work groups are assigned to specific areas by organization and type of work performed within that organization. For example, in the Hicksville Operations Center, the Electric organization can have two areas: Overhead Electric and Underground Electric.

Job Priority: Job priority is the single most important piece of job information to the SE when it assesses a value to each job. The higher the priority (lower number), the more valuable the job is to the Company. LILCO uses a corporate priority system which is used consistently across the entire company.

Resource Pattern: A resource pattern is used to generically express the required resources of a job. It can include any number of resource types from: employees, vehicles, and/or equipment. The sole requirement is that at least one employee must be included on each resource pattern. From this information, the SE then searches for the "demanded" resources of each job and tries to satisfy them so that a job can be "fully" scheduled.

Worker Skill: A worker skill represents the employee component of a job's resource pattern. Some examples include: Lineman, Senior Lineman, Working Foreman A, Welder, Mechanic B, etc.

Crew: All of the workers required to perform a job constitutes a crew.

Vehicle Type: Examples of vehicle types include: 50-ft. bucket truck, 30-ft. bucket truck, compressor truck, passenger vehicle, etc.

Equipment Type: Examples of equipment types include: trailer, Dig-it, backhoe, etc.

Job Duration: Job duration is the "clock" time required to complete a job. The SE calculates this as: (Job estimate + Additional hours)/crew size. If a job has an estimate of 20 man-hours, 4 additional hours added, and a crew size of 3 workers, then the job duration equals 8 hours.

Early Start Date (ESD): The earliest date that a job can be scheduled to start. The Early Start Time (EST) is the earliest time a job can start on the scheduled start date.

Late Finish Date (LFD): The latest date that a job can be completed by. Unless a job was delayed after it already started (i.e. job in progress), this date cannot be violated. The Late Finish Time (LFT) is the latest time a job can be completed by on the scheduled end date. EST and LFT are usually defined for small jobs that should be started and completed within the same day.

Scheduling Origin: This is the start of the scheduling window for a given schedule. It defaults to "today."

Scheduling Horizon: This is the end of the scheduling window for a given schedule. If the user wants a two week schedule, this value will equal two weeks from today.

Grid Number: The grid number represents the geographic location of a job. The SE uses this to determine job proximities.

Travel Time: Travel time is a job specific value that represents the time required to travel to a job and to return to headquarters, if necessary. The SE uses this value to minimize travel time expended throughout the work day.

Scheduling Status of Jobs: A job is considered "scheduled" if its start time is defined and it has all of its resources (as defined by its resource pattern) assigned to it. A job is considered "jeopardized" if its start time is defined and it has at least one employee assigned to it, but is missing at least one other required resource. An "unscheduled" job has zero employees assigned to it and no start time defined. It could be either a new job or a job which the SE failed to schedule or at least put into jeopardy (to find at least one available employee with the required skills).

### **Scheduling Methodology**

The scheduling engine is the core of where all of the decisions are made for a schedule. The SE defines start time for all jobs and assigns limited resources to jobs if possible. The same resource cannot be assigned to two or more different jobs at the same time. The SE receives both job requirements and resource availability information and attempts to create a schedule with one basic premise: schedule the most valuable jobs first while utilizing the given resources as efficiently as possible. Job requirements and resource availability are defined in the Company's jobs database (i.e. area, job priority, due date, resource pattern, etc.). A user can define specific parameters via the CRMS graphical user interface (GUI). The SE should try to "honor" user requests while searching for the best solution using basic scheduling algorithms.

Some of the scheduling requirements are rules and must be satisfied for a job to be scheduled while others are preferences and *should* be satisfied only if possible. An example of a “strong” user preference is when the user “freezes” a low priority job so that it will be scheduled tomorrow (as opposed to next month since it was promised to the customer that the work be done by then). Here, it is important to remember that the user always “has the last say”. The user may have timely information that the system is unaware of. As such, the SE allows the user to manually override any scheduling solution at any time. In this case, the SE will give this job a much higher value and try to schedule it much earlier than usual; thus, increasing its chances of becoming a scheduled job.

An example of a weaker preference is when the user defines (via the GUI) an employee preference relationship where one employee prefers to work with another. Here, with all of the other more critical constraints to scheduling that the SE must consider, the SE will try to satisfy this preference only after it satisfies the more important constraints.

When the user runs a schedule, he/she must define its origin and horizon. Although we have jobs that go out for years, the user will most often define the scheduling horizon to two weeks. Here, the SE will try to produce two weeks worth of work for its given resources. The scheduling horizon is an example of a “dialable” parameter.

Once the scheduling data are defined, the SE will apply a series of algorithms and constraints which will generate a “good quality” schedule in the user defined time. The user has some special “dialable” parameters which will affect the scheduling algorithms, and thus, the resulting schedule. For instance, the user may define that a schedule be prepared which maximizes the number of jobs to be scheduled; as opposed to the number of hours to be scheduled.

The user will use the GUI to analyze the results of the scheduling. The user may make several adjustments or “manual overrides” and ask that scheduling be re-run. The processes of “manual overrides” and “re-scheduling” gives the user flexibility to control and customize a given schedule. It should be noted that the user views the GUI as the entire scheduling system. Although many users understand how the SE “thinks” or processes each schedule, to most users the SE is a “black box.” In a future release, the SE and GUI will be combined inside the same constrained environment, and all of the user scheduling actions will be under the control of the same constraint “demons” that now control the SE.

### **Basic Scheduling Logic**

The SE is given information about each job that assists it in producing a schedule. With the limited resources, its main challenge is to schedule those jobs that are most valuable to the company first. The SE uses a set of simple rules to assist it with basic scheduling processing. How it matches resources to these jobs is where the real challenge lies.

The SE has a long list of scheduling rules and the ILOG-based implementation allows additions to this list without changes in the search algorithms. The list below shows the

four independent scheduling processes (in order) that the SE uses every time it runs a new schedule:

- 1) Schedule Frozen Jobs
- 2) Schedule Interruptible Work Jobs
- 3) Schedule Other Jobs
- 4) Try to Schedule (Jeopardize) Unscheduled Jobs.

For all of these four scheduling processes, the SE values different jobs based on the following characteristics (in order of value):

- 1) Highest Job Priority (lowest number)
- 2) Minimum Late Finish Date (earliest)
- 3) Minimum Early Start Date (earliest).

Based on the user requirements, other characteristics (like "Job Duration", "Use of the Given Resource Skills") could have different values.

More scheduling rules can be added and their order could be defined by the user. In addition, resource utilization levels can serve as a "preference" to be included within the above rules. Here, the SE would schedule a more consistent amount of work for each employee (i.e. maintaining average work levels throughout the work force, of say, approximately 7 scheduled hours of work per man per day). The SE would "prefer" employees with lower levels of scheduled work hours when assigning employees to jobs. Throughout the SE, basic scheduling logic is very often returned to as a means to treat specific job comparison situations. The rules above are the backbone of logic that the SE uses when considering which to schedule first.

## **Jobs**

Jobs originate in the LILCO's legacy database. Each job includes specific information that helps the SE determine the level of importance that should be given to it. When a job gets passed to the SE, it combines this job level information with other resource level information and begins its process of scheduling.

### **Job Priority and Escalation Algorithm**

As mentioned earlier, each job has a priority. The higher the priority (lower number), the more valuable the job is to the Company. The Company uses priority "decades" to classify different types of jobs:

## JOB PRIORITY

| <u>Decade</u> | <u>Number</u> | <u>Type of Work</u> |
|---------------|---------------|---------------------|
| Decade 0:     | 0- 9          | Emergency           |
| Decade 1:     | 10-19         | Customer work       |
| Decade 2:     | 20-29         | Reserved for future |
| Decade 3:     | 30-39         | Project work        |
| Decade 4:     | 40-49         | Reserved for future |
| Decade 5:     | 50-59         | Normal backlog jobs |
| Decade 6:     | 60-69         | Normal backlog jobs |
| Decade 7:     | 70-79         | Normal backlog jobs |
| Decade 8:     | 80-89         | Normal backlog jobs |
| Decade 9:     | 90-99         | Interruptible Work  |

This classification is defined on the business level and can be modified without any changes in the SE, but the SE can escalate some priorities. The general purpose of the escalation algorithm is to ensure that lower priority jobs (higher numbers) get the attention of the SE as their individual late start dates approach "today" (the first day of the schedule). As a low priority job ages through time, it's probability of getting scheduled increases. Therefore, to satisfy this need, the SE actually escalates the priorities of jobs as each job's LSD approaches today.

### **Interruptible Work Jobs**

The SE provides the ability to "reserve" a specific amount of resources for any *anticipated but as yet unknown* emergency work that may occur during a work day. Because interruptible work jobs can be started and stopped easily during the day, they are good candidates for this "flexible" part of the schedule. The SE defines any job that has an original priority within 90 and 99 as an interruptible work job. It is important to note that reserved interruptible work resources are not idle. They are, in fact, scheduled to lower priority work so that if an emergency does occur, these resources will be available and the probability of interrupting more critical work is less.

### **Overtime**

The SE allows the user to specify which jobs are allowed overtime (OT), not which employees. By telling the SE which jobs can be worked on OT for how many OT hours and during what periods, the CRMS has direct control over overtime expenditures.

The SE understands two types of overtime. The first is defined by type of work. Here, the user tells the SE to allow OT for a specific type of work (i.e. setting overhead distribution transformers). Whenever the SE finds a job that falls into this category, it will schedule OT for it, if applicable. The second type of OT is defined for a specific job. Here, the user has the flexibility to allow OT for a specific job, regardless of its type. In both cases, the user has defined a job to be an overtime candidate. As the user does not know exactly when the SE will schedule a job, they are considered OT "candidates."

The SE schedules OT candidate jobs in exactly the same manner that it schedules other jobs. However, the SE does not schedule overtime just to "save a day of work". The SE assumes that total resource demand is greater than total resource availability. As such, it schedules OT allowances whenever it has the chance so that it can schedule more work and increase aggregate resource utilization levels.

### **Multi-Shift Jobs**

The general purpose of the multi-shift function is to assign more than one crew (including all of the required resources) to the same job at different shifts during the work day. The SE considers each of these "shift assignments" (one crew per shift) as a unique sub-job of the single job to be scheduled. The SE can then schedule each of these sub-jobs just like any other job while respecting the single job's characteristics (overtime, estimate, early start date, late finish date, etc.). As such, for a given sub-job shift, it only looks for resources that are available during that specific shift entirely. If an employee is available from 7:00 a.m. to 3:00 p.m. and the sub-job shift is defined from 8:00 a.m. to 4:00 p.m., then this employee is not considered. Thus, the shift is the dictating parameter when assigning resources to jobs. Although it takes longer for the SE "to think" about more complex multi-shift jobs, the SE has no special limitations on the scheduling of these jobs.

### **Frozen Jobs**

Jobs exist in two scheduling states: frozen and unfrozen. Frozen jobs constitute jobs which are "fully scheduled" and whose dates and resources are not likely to change. In other words, the scheduling engine will give its highest respect to frozen jobs and it will "try it's best" to honor their fully scheduled status. The scheduling system automatically makes jobs frozen as they approach their scheduled start date, but the user, via the GUI, always has the flexibility of "freezing" or "unfreezing" a job.

### **Travel Time and Job Clusters**

The general purpose of this function is to complete more work during a typical work day by minimizing unnecessary travel time. When certain conditions exist, the SE tries to assign a specific team to several "travel candidate" jobs within the same "job cluster." The SE dynamically builds each job cluster as it tries to schedule jobs. In general, for jobs of similar value (i.e. same priority decade), if several jobs are "close" in proximity and share the same resource pattern demands, they can belong to the same job cluster where travel time will play a significant role in the scheduling of each job.

### **Back-Up Jobs**

The SE creates a "base" schedule and then tries to build a "back-up" schedule by considering the remaining unscheduled and jeopardized jobs. Each job on the base schedule can only have one back-up job. The SE uses back-up jobs for two reasons:

- 1) as substitute work when a scheduled job cannot be worked, and
- 2) as additional work when the scheduled job is completed ahead of schedule.

The SE uses specific back-up criteria to determine if a job can serve as a back-up. For instance, back-up jobs should: have small durations, be located near the applicable base

job, and require the same or "smaller" resources pattern as the applicable base job. In general, the more unscheduled and jeopardized jobs a schedule has, the more likely the SE will be to satisfy back-up job requirements.

## **Resources**

Resources that the SE assigns to jobs for scheduling fall into 3 categories: employees, vehicles, and equipment. In the future, materials required for each job (i.e. utility poles, transformers, wire, pipe, etc.) will be defined as a fourth resource category.

### **Employee-Employee Preferences**

The general purpose of this function is to allow the user to define which employees "prefer" to work with one another. This does not represent a constraint to scheduling, but a preference that is honored whenever possible. The scheduling engine takes these preferences into consideration when scheduling but it does not consider their definitions a "must" for a correct schedule. In fact, given all of the other scheduling rules, this reference carries little weight in assigning resources to jobs.

The user defines specific employee preferences via the GUI. For example, if Bob prefers to work with Bill, one relationship is defined. The SE interprets this single relationship as "Bob prefers to work with Bill and Bill prefers to work with Bob". When the SE first begins to assign employees to a job, the first employee is assigned arbitrarily. Then, given the first employee assignment, the SE will try to honor any preferences that this first employee may have. In addition, the more preferences an employee has, the more "desirable" that employee will be. So, a worker with 3 employee preferences will be assigned to a job before an employee with less than 3 preferences (all else being equal). If the SE cannot "honor" an employee preference, it will then just try to assign any employee (as defined by the resource pattern) to the job.

### **Driver-Vehicle Preferences**

Like employee-employee preferences, the SE also considers driver-vehicle preferences for a given schedule. Again, the user defines specific driver-vehicle preferences via the GUI. For example, if employee A prefers to use vehicle #123, one relationship is defined. The SE interprets this single relationship as "Employee A prefers vehicle #123 and vehicle #123 prefers employee A".

Since resources are not assigned to jobs in any specific order, the reverse interpretation above must be understood by the SE. In addition, when the SE first begins to assign employees to a job, the first employee is assigned arbitrarily. Then, given the first employee assignment, the SE will try to honor any driver-vehicle preferences (as well as employee preferences) that this first employee may have (assuming a vehicle is needed for the job). Again, the more preferences an employee has, the more "desirable" that employee will be. If the SE is trying to assign a vehicle to a job which has employee(s)

already assigned to it and it cannot "honor" a driver-vehicle preference, it will then just try to assign any vehicle (as defined by the resource pattern) to the job.

### **Resource Replacements**

Skills are typically hierarchical in nature. For example, consider two types of "bucket" trucks used in LILCO's electric business: 50-ft and 30-ft. If a job requires a 50-ft bucket truck, the 30-ft bucket truck cannot be assigned as a vehicle resource to this job. However, if another job requires a 30-ft bucket truck, then the 50-ft bucket truck can be used as a replacement. Of course, in order to more efficiently utilize resources, assigning a 50-ft truck in place of a 30-ft truck should only be done if no 30-ft trucks are available. In the above example, the resource replacement logic defined is interpreted as "30-ft bucket truck can be replaced by 50-ft bucket truck."

The same resource replacement logic is applied to human skills and equipment. There are also several levels to the hierarchy of replacement logic (i.e. a 40-ft bucket truck can be embedded within the 30-ft & 50-ft replacement logic above).

### **Team Splitting**

Teams are assembled from a pool of resources from a specific site, which is called an area. Once a team is assigned to its first job, the SE then tries to assign this team to a second job which requires the same resource pattern. However, if no other job requires the same resource pattern for the day, the team may be split to satisfy other job demands. The best case scenario is when a team is split into exactly two or more teams.

The underlying team splitting rule is when a team is "formed" during the day, it must be formed solely from another team. As such, a team becomes a kind of "mini-pool" of resources.

For jobs which are not subject to these geographic restrictions (i.e. power plant jobs), the "team splitting" rule still applies. Here, we benefit in that if a team "overruns" its estimated hours, it creates less havoc in the operation. That is, since the team is usually re-assigned as a whole, the entire team is late or on time for subsequent jobs.

Consider an example of Job A with priority 20 and resource pattern equal to five Linemen and two 50-ft bucket trucks. If this job ends during the day and the SE finds a new job, say Job D, with the same resource pattern and priority 22, the SE will schedule this same team to Job D. However, if the SE finds two other jobs that are more "valuable" than Job D, then it will "split" Job A's team to schedule these jobs first. Assume Job B has a priority 21 and resource pattern equal to three Linemen and one 50-ft bucket truck. Also assume Job C has a priority 21 and resource pattern equal to two Linemen and one 50-ft bucket truck. Lastly, assume that the SE could not find any other resources to satisfy the demands of Job B & C. Therefore, for jobs B & C to be scheduled, Job A's team must be split. Here, it was more valuable for the SE to schedule Jobs B & C at the expense of Job D. The SE makes these kinds of decisions in every scheduling run!

## **Frozen Resources**

The frozen resources function allows the user to define or freeze specific resource(s) to a specific job. This user request only constitutes a preference; it is not a scheduling rule that must be honored. For instance, a frozen resource will be dishonored if the scheduling engine determines that the job can be scheduled earlier only if another resource can be used in the place of the frozen resource. In addition, if the SE needs the resource to accomplish a higher priority job, the frozen resource preference will be dishonored since it is more important (business decision) to get the higher priority jobs scheduled first as opposed to honoring preferences.

## **Using Scheduling Engine**

The current release of the SE works as a stand alone process. It could be started by different authorized users via GUI and actually runs on the UNIX server. It gets data from the central database, runs the scheduling processes, and lastly, saves the results back to the same database.

## **Using Dialable Parameters**

Dialable parameters are user-defined variables that the SE uses to produce a schedule. These parameters can be “dialed” or changed for each new scheduling run. This allows the user to adjust the characteristics of the schedule and to use "what-if" analyses without changing any program code. Current dialable parameters include: user Id, scheduling origin and horizon, kickout factor, and several optimization criteria.

## **Scheduling Horizon**

LILCO jobs are spanned over several years. As such, we should only be interested in considering jobs to be scheduled over a specific "horizon." The scheduling horizon presents the latest date managed by the SE. Its value can range between one day and six months or more. A typical scheduling horizon is set by the user to equal two weeks. This means that the SE will consider any job that has an early start date falling on or before the scheduling latest date. If a job's late finish date falls after the scheduling latest date (overlapping scheduling window), the SE accepts the job and it attempts to schedule it. However, if the SE cannot find the resources to schedule the job to start on or before the scheduling latest date, the SE "stops thinking" and does not attempt to schedule the job any further.

## **Kickout Factor**

The kickout factor is a time window starting “today” and ending by an offset number of days defined by the user. For instance, if today is 6/6/95 and the user defines a kickout factor of 5 days, the kickout window is: 6/6/95 to 6/10/95. All the jobs that are scheduled within the kickout factor must have all of their resources assigned. It applies only to human resources; not vehicles or equipment. That is, no partially scheduled (jeopardized) jobs, will be found starting within the kickout factor window. In other words, a job with missing human resources that starts in this window, becomes

"unscheduled" to attract maximum attention of the user. This parameter allows the user to minimize the jeopardized jobs that are generated; and thus, prevents tying up some resources on jobs which have "voids" which may be difficult to fulfill.

### Optimization Criteria

There are several dialable parameters that allow the user change the "quality" of the schedule. From each individual user's perspective, using these parameters to fit one's scheduling objectives constitutes improving the schedule or "optimizing" it.

#### Maximize Number of Scheduled Jobs

If the user prefers to maximize the number of scheduled jobs for a given schedule, the SE will be instructed to choose small duration jobs when it has the chance. Although, the total numbers of aggregate hours scheduled may decrease, the number of jobs scheduled should increase.

#### Maximize Number of Scheduled Hours

If the user prefers to maximize the number of scheduled hours for a given schedule, the SE will be instructed to choose large duration jobs when it has the chance. This is the exact opposite of maximizing the number of jobs for a given schedule. Although, the total numbers of aggregate jobs scheduled may decrease, the number of hours scheduled should increase.

#### Maximize Use of a User-Defined Skill

If the user prefers to maximize the use of a specific skill, that skill must be included as the dialable parameter for the SE to include it in its scheduling rules. This tends to occur more often only with large schedules. When the SE has an opportunity to use one skill over another, it will use the skill defined here by the user. For example, if a Senior Lineman can do the work of a Lineman, the SE will "try to" select the Senior Lineman when this skill is defined as the one to be maximized.

#### Minimize Travel Time

If this criterion is defined, the SE will honor "job closeness" more than "job priority" for jobs inside the same priority decade. As a result, if the same team is assigned to several jobs on the same day, the SE will try to minimize the travel time expended by this team for that day. The next job to be assigned to the team is currently selected based on the shortest travel time. The SE uses a "maximum travel time" dialable parameter to make this function more flexible to the varying user needs. For example, if this parameter is set to 30 minutes, then the SE will consider travel time more than job priority for jobs in the same cluster that are less than or equal to a 30 minute "drive away." LILCO's legacy systems have varying speed tables for each area to compute this "time needed for travel." The next SE release will also solve the traveling salesperson task.

### Scheduling Results

The SE produces the resulting schedule in terms of data structures that will be saved in the central database. The quality of the SE is expressed both by the quality of its results, and by its performance.

### Metrics

The SE produces some metrics to evaluate the quality of the schedule. These metrics include: number and/or percentage of scheduled jobs of the same priority decade; number and/or percentage of scheduled hours by priority decades; number of used resources and resource skills; number of idle hours and more. The metrics could be presented via bar charts to help the user to evaluate the quality of the SE work.

### Reports

The SE produces the report which includes:

- 1) scheduling parameters
- 2) scheduling environment
- 3) results in form of the following job lists:
  - frozen jobs
  - interruptible work jobs
  - regular scheduled jobs
  - jeopardized jobs
  - unscheduled jobs.

All jobs are presented in the order the SE processed them. The scheduled and jeopardized jobs include along with a job definition (name, pattern, early start, late finish, duration, grid) the description the assigned teams and time intervals.

The SE also produces the reports with the explanations of positive or negative solutions it took while the scheduling. The list of rejected jobs and the reasons for the rejection is also provided.

### Graphical Schedule Representation

The current GUI does not directly interact with the SE, but rather display results produced by the SE and saved in the database. It allows to show the jobs over time intervals, selects jobs or resources based on different search criteria and modify the scheduling data for the further schedule runs. The GUI was developed using PowerBuilder for MS Windows.

ILOG Views has been used to present the SE results via Gantt charts. It gives the user an ability to see and evaluate the entire schedule or the selected part of it. The user can use the "point-and-click" technology to navigate through the time intervals and select or modify the scheduling results. In the next phase we are planning to combine the SE and GUI in a way, when the same scheduling rules and constraints will be applied to both the SE and the user actions.

### Efficiency and Extendibility

In June 1995 the CRMS (with the SE) began being used in the limited production or "pilot" environment. The initial results have demonstrated the high efficiency of the SE: the average scheduling time is about 10 seconds on the UNIX machine and from 30 seconds to one minute on the PC. Therefore, the user can perform "what-if" analysis since iterative rescheduling is no longer an expensive process.

The use of ILOG libraries for constraint-based programming allowed LILCO to develop the SE as an extendible system core that can be enriched by new business constraints and adjusted to fit different user requirements without changing the basic scheduling algorithms. Only the real-world experience can prove the usefulness of different scheduling concepts and strategies. Even during the development cycle several important concepts were introduced in the late phases: it would be enough to mention only multi-shift jobs! Only the flexible ILOG-based object-oriented environment with clear demarcation between problem definition and problem solving parts allowed us to perform real iterative development. As the scheduling system is in "pilot" phase production, we are planning to add new concepts and business constraints dictated by additional real business needs.

### **Benefits to the Company**

LILCO spends over \$112 million annually on scheduled work that consumes about 2.7 million hours of employee labor. An Anderson Consulting study of the utility industry stated 5-20 percent gains in productivity when a company installs a new planning and scheduling system. These gains come from improved efficiency and utilization of all resources. Conservatively assuming just 5 percent in savings equates to \$5.6 million cost reduction per year for LILCO!

Other intangible benefits include substantial improvements in customer service. Since the new, automated scheduling system will not allow "bad data" (i.e. late finish dates prior to today, missing resource patterns, etc.), the sheer *purification* of job data will increase of likelihood of meeting customer commitment dates. In addition, the scheduling analyst will take on a more critical role as a "single point of contact." The scheduling analyst will have a tremendous amount of flexibility in maintaining job schedules and sharing resources across different "areas" and/or organizations at LILCO.

Furthermore, as LILCO maximizes its use of its own resources, its dependency on contractors should diminish. Big cost savings here! All in all, the ILOG's Scheduling Engine incorporated into LILCO's CRMS have set the stage for a new era of the planning and scheduling of thousands of resources. As LILCO moves into full production with CRMS, many benefits as well as significant savings are expected.