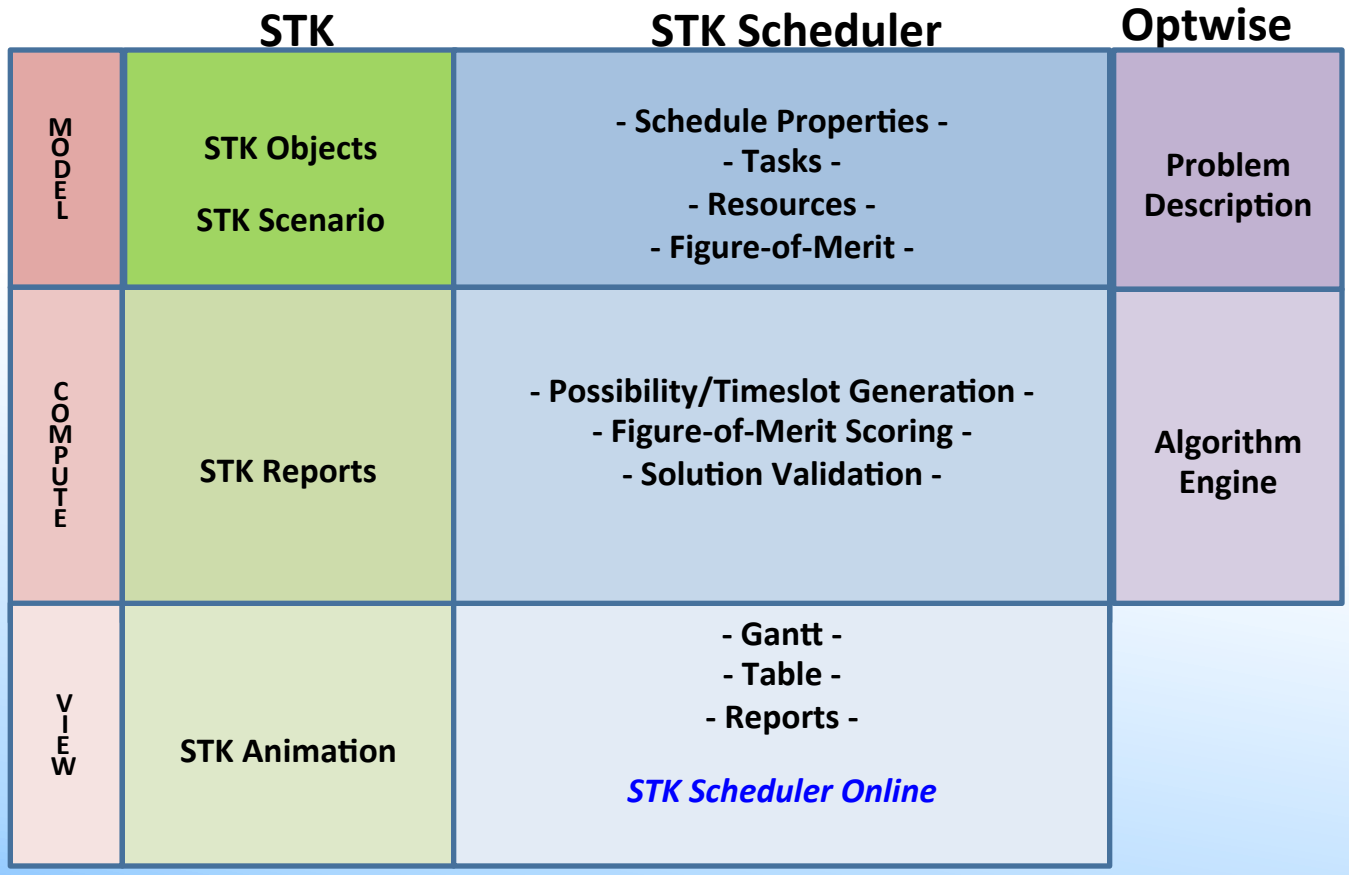
A Flexible Architecture for Creating Scheduling Algorithms as used in STK Scheduler

Dr. William Fisher, Optwise Corporation

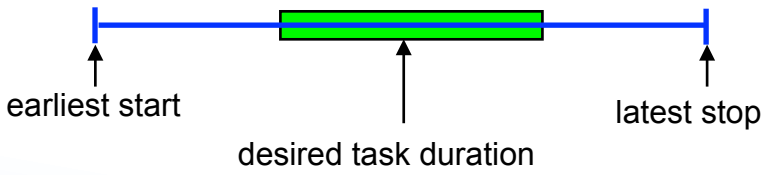
Ella Herz Orbit Logic Inc.

STK Scheduler Architecture

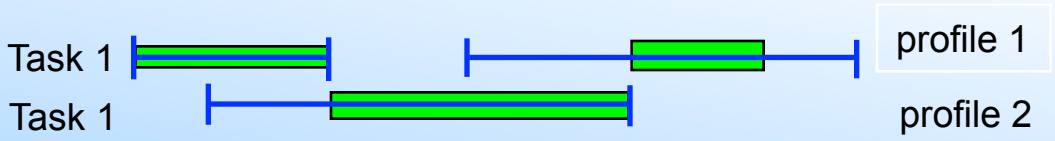


STK Scheduler Constraints

Tasks and Time Slots

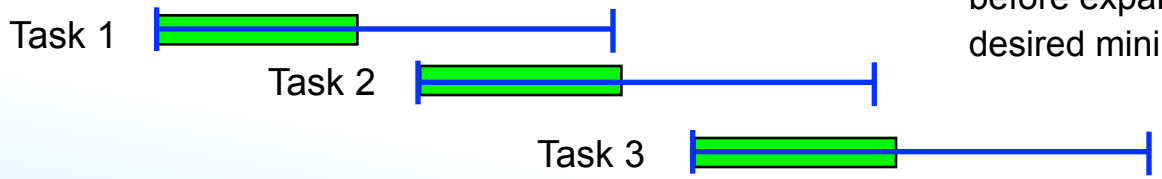


Tasks are scheduled onto time slots which represent the time during which a particular resource or combination of resources (a solution profile) may be used to satisfy the task.

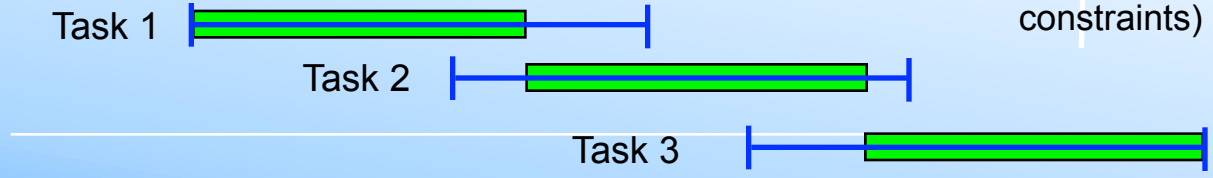


Handover tasks may use multiple time slots and different profiles

Task (duration) Expansion



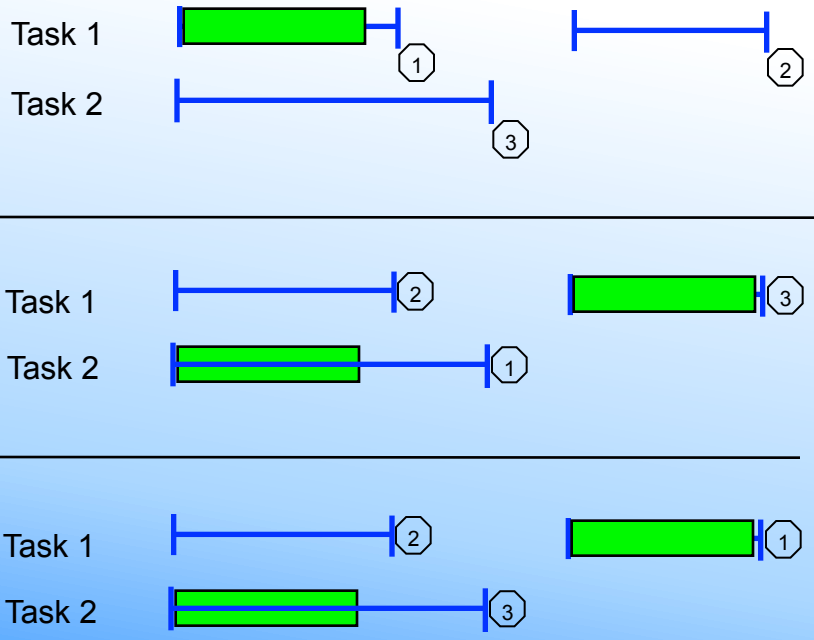
before expand each task is assigned the desired minimum duration



after expand each task is assigned up to the maximum desired duration (subject to constraints)

Algorithms from 10,000 feet

Most algorithms search and tasks slots in an ORDERED fashion. Changing the order can change the result. (Custom Algorithms allow the end user to change the order of slot consideration.)



The neural algorithm uses a constraint driven global competition model to solve the schedule (The Neural Algorithm is not customizable)

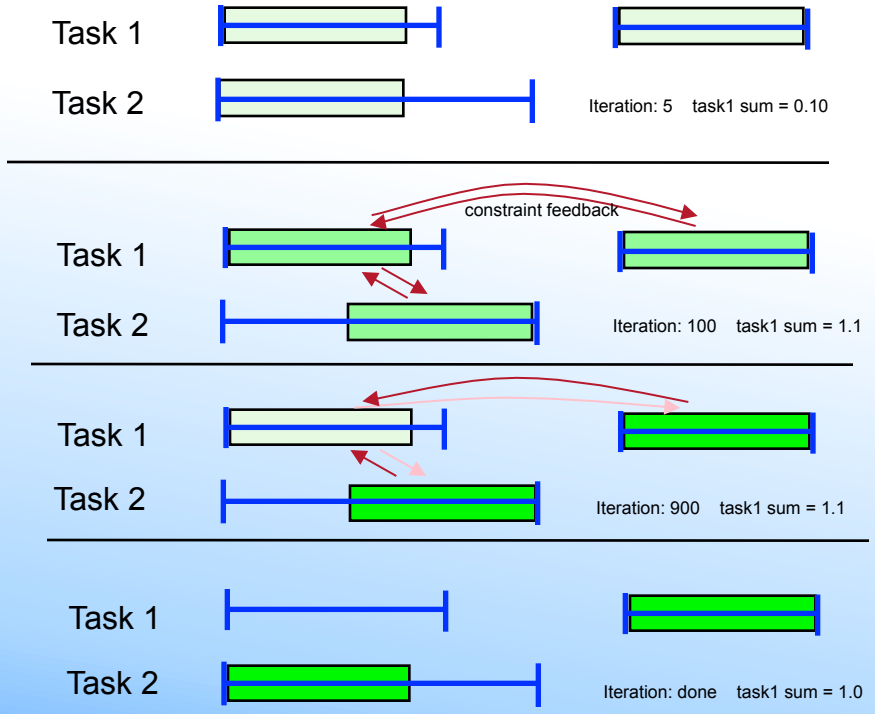


Figure of Merit Equation

$$FOM = \sum_{task=i} \mathrm{Pri}_i \left(\begin{array}{l} K_{assign}(Ua_i) + K_{dur}(AssignDur_i) + K_{desire}(SlotDesire_i) \\ + K_{early}(EarlyBonus_i) + K_{max}(MaxBonus_i) \\ + K_{userstart}(|DesiredStart_i - Start_i|) \end{array} \right)$$

$$+ UserFunc$$

$$Ua_i = 1 : assigned, \quad 0 : unassigned$$

K_{assign} is used to adjust the relative weight for assignment of at least minimum duration. This term is complementary to the next term.

K_{dur} is used to adjust the relative weight of assignment times the duration. Thus tasks with longer durations will affect the FOM more than shorter ones.

K_{desire} is used to adjust the relative weight of desirability calculated for the final position of the task.

K_{early} is used to adjust the relative weight of the early bonus. The early bonus is 1.0 if the task was scheduled as early as possible and zero if as late as possible for that task. (linear)

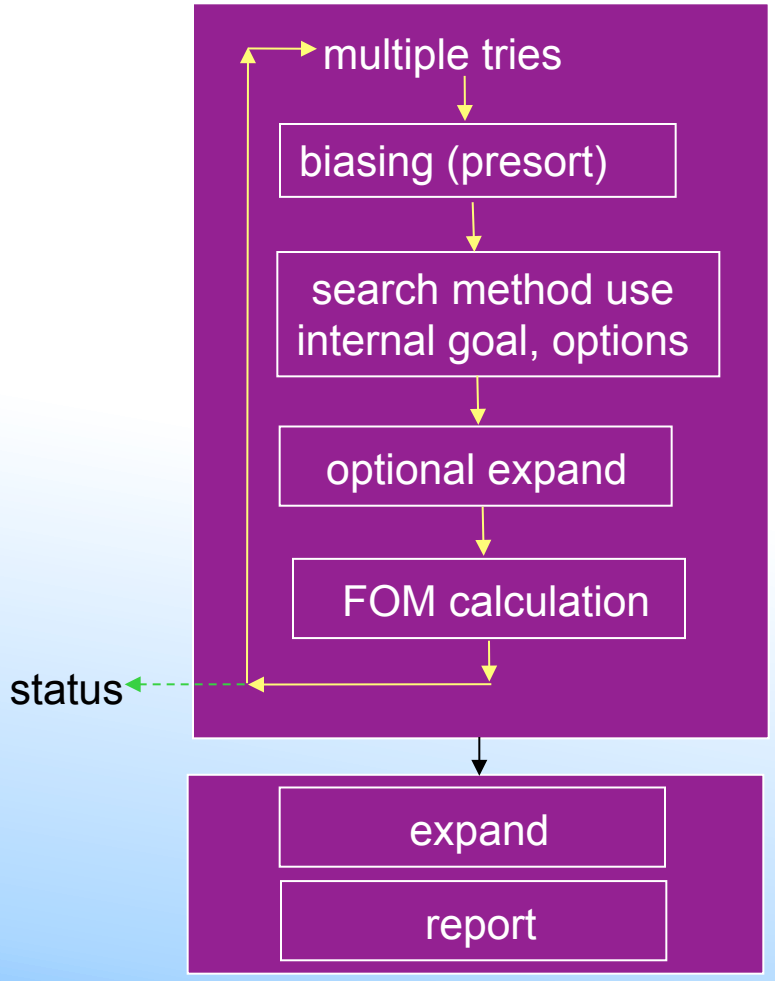
K_{Max} is used to adjust the relative weight of the Max duration bonus. It is 1.0 if maximum duration is scheduled and 0 if minimum duration is scheduled. (linear)

$K_{userstart}$ is used to adjust the relative weight of the userstart bonus. The userstart bonus is 1.0 if the task was scheduled as close to the desired user start and zero if far away as possible for that task.

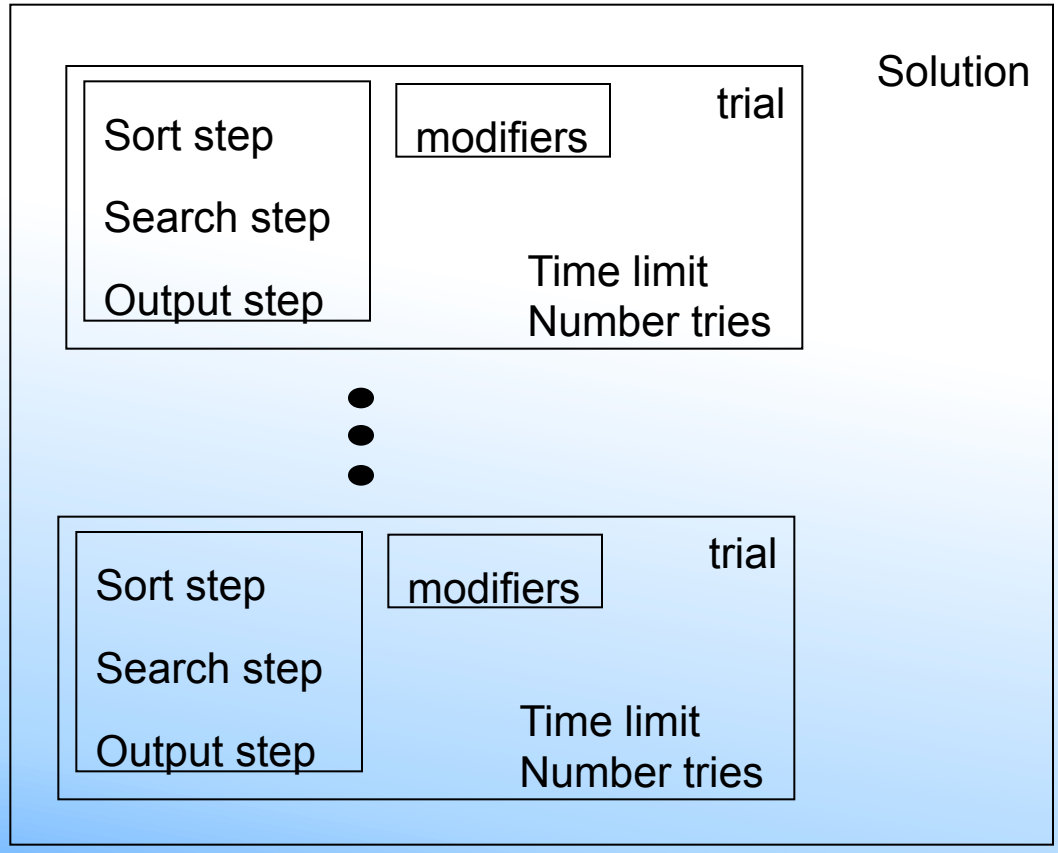
UserFunc link to external user function. (dll)

Custom Algorithm Process

- The algorithm interface allows the user to control the presort, the randomization process for multiple runs, and the search process.
- There are over 10 different sort criteria.
- There are many options for controlling the search phase.



Algorithm Description Strings

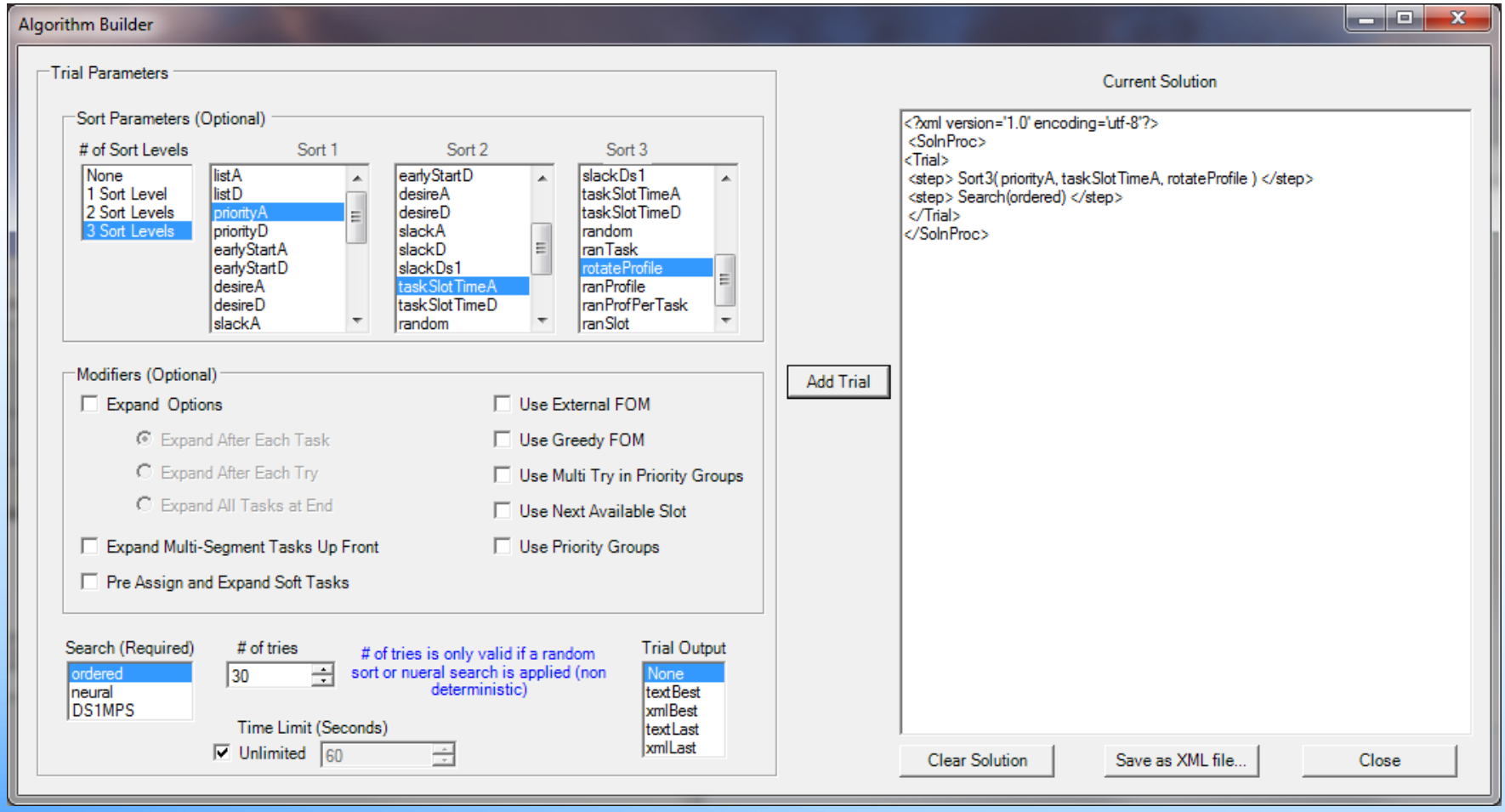


Each trial defines a complete algorithm

Only a search step is required, but normally has a sort step.

Multiple trials can be chained to create "multi-algorithm" searches

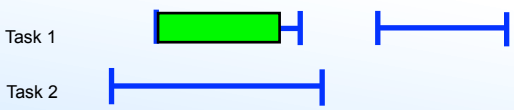
Building an Algorithm



Example Solution Strings

One Pass

```
<?xml version='1.0' encoding='utf-8'?>
  <SolnProc>
<Trial>
<modifier> usePriGrps </modifier>
<step> Sort1(desireA) </step>
<step> Search(ordered) </step>
</Trial>
</SolnProc>
```



Sequential

```
<?xml version='1.0' encoding='utf-8'?>
  <SolnProc>
<Trial>
<modifier> usePriGrps </modifier>
<step> Sort1(earlyStartA) </step>
<step> Search(ordered) </step>
</Trial>
</SolnProc>
```

Custom1

```
<?xml version='1.0' encoding='utf-8'?>
  <SolnProc>
<Trial numTries= '10' >
<modifier> usePriGrps </modifier>
<step> Sort2(desireA, ranTask) </step>
<step> Search(ordered) </step>
</Trial>
</SolnProc>
```

In this custom algorithm the Sequential algorithm is modified to search a different (randomly) generated task order for 10 tries. There is a 50% probability of full assignment on one try

Regression Test Data

name	#tasks	#resources	#slots
15T2R	15	2	20
OL-Ex10	120	80	363
Cust1	6510	11	6720
Cust2	619	163	26850
Cust3	553	160	24580
Cust4	526	58	4006

15T2r

Algorithm	# assigned	FOM
OPS	13	221.8
Sequential	13	221.8
MPS	14	230.3
Neural(1)	15	243.3
Neural(100)	15	243.8
Random(1)	15	238.5
Random(100)	15	244.2

OL-Ex10

Algorithm	# assigned	FOM
OPS	116	195.9
Sequential	120	239.1
MPS	116	218.2
Neural(1)	120	202.1
Neural(100)	120	204.0
Random(1)	120	196.0
Random(100)	120	199.8

customer sample assignments

Cust	OPS	Seq	MPS	Neural (1)	Neural (5)	Ran (1)	Ran (5)
1	37	37	37	44	44	48	53
2	577	598	615	573	578	607	609
3	544	540	550	526	527	543	549
4	255	251	267	259	263	254	253

customer samples solution times

Cust	OPS	Seq	MPS	Neural (1)	Neural (5)	Ran (1)	Ran (5)
1	13	13	31	15	26	14	15
2	9	8	13	171	823	8	11
3	8	7	12	170	818	7	9
4	2	2	2	7	28	2	2

Largest problem to date was a 29,745 task, 33 resource, 24 hr. problem. It had 1,710,462 slots a 20,706 assignment solution was found in 161 seconds.

Example Case Study

2 week, 4452 task problem, 158,069 time slots, 41 unique priority levels

Primary goal: Assign highest priority tasks first if at all possible

Secondary goal: maximize assignment and match target resource usage. This goal is measured in a "User Grade spreadsheet calculation".

User Grade = 1 + (100* A -1000*B - C)/D, where

A = ∑(number of events over desired), (over tasks)

B = ∑number of events under desired), (over tasks)

C = ∑(individual resource usage penalty) , (over resources) and

D = ∑((501-pri)*number desired events). (a constant)

User Grade = 1 –(1000* (number unassigned) – off-target resource penalty)/D

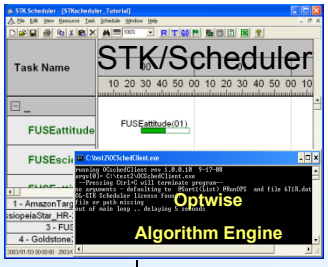
(Notice that the user grade does not include a term to award priority assignments)

Case Study Strategies

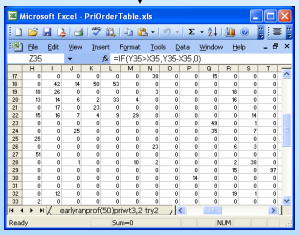
Optwise Corporation

ORBIT LOGIC

Try suitable standard algorithms and calculate an external User Grade after the fact

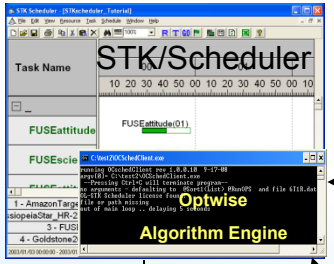


STK/Scheduler export macro

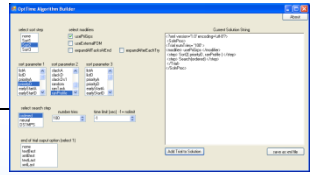


Calculate User Grade

Create and use a custom algorithms Calculate an external User Grade after the fact

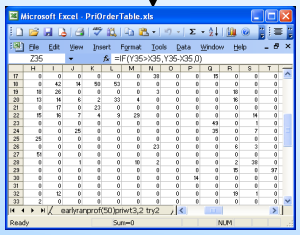


Algorithm Builder



add randomize profiles to sequential

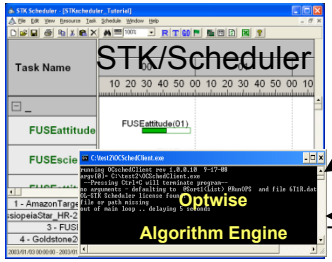
STK/Scheduler export macro

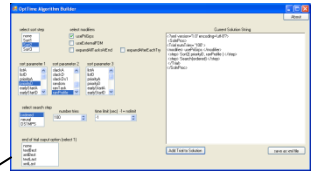


FOM = pri range/pri

Calculate User Grade

Use a custom Algorithm and or external FOM incorporating all or part of User grade to improve result

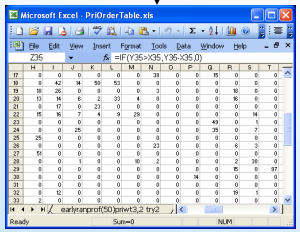




add randomize profiles to sequential

External FOM Calc

STK/Scheduler export macro



FOM= UG - 10*(501-pri)

Calculate User Grade

User Grade = 1000* (number unassigned) – custom resource penalty

Results Summary

Algorithm	Assigned	FOM	Penalty	# missed relative best at pri=
Sequential	3608	0.5592	0.050	-32 at pri=16
Sequential with random profile internal FOM 50 tries	3636	0.5558	0.067	-7 at pri=16
Sequential with random profile external FOM 200 tries	3644	0.5645	0.062	Best case primary criteria
Greedy External FOM 30 tries	3584	0.5775	0.002	-2 at pri=6

Solve times: 7 sec per try case1,2, 11 sec per try case 3 , case 4, greedy was over 300 sec/try. (64 bit Intel Core 2 2.13 GHz processor.)

Take- away

More than 10 years of operational experience has shown:

Users value predictability and speed of solution – Sequential, One Pass

Still tend to use total assignment instead of figure of merit

A flexible architecture is key to provide quick customer updates

For more information contact:

Dr. William Fisher
fisher@optwise.com

Ella Herz
ella.herz@orbitlogic.com