Model

Variables

$tasks$	The number of tasks to schedule
$machines$	The number of machines available
$tools$	The number of tools available
$cycles$	The number of cycles performed
$uniqueTasks$	The number of unique tasks
$trays$	The number of trays available
$fixtures$	The number of fixtures available
$components$	The number of components available
$nbrConcurrentGroups$	The number of concurrent groups used
$nbrOrderedGroups$	The number of ordered groups used
$T = \{1 \dots tasks\}$	The tasks
$M = \{1 \dots machines\}$	The machines
$To = \{1 \dots tools\}$	The tools
$Tr = \{1 \dots trays\}$	The trays
$F = \{1 \dots fixtures\}$	The fixtures
$C = \{1 \dots components\}$	The components
$Cy = \{1 \dots cycles\}$	The cycles
$concurrentGroups = \{1 \dots nbrConcurrentGroups\}$	The concurrent groups
$orderedGroups = \{1 \dots nbrOrderedGroups\}$	The ordered groups
$d_t \in \{0 \dots 2^{63} - 1\},\ t \in T$	The duration of t
$s_t \in \{0 \dots 2^{63} - 1\},\ t \in T$	The start time of t

$e_t = s_t + d_t,\ t \in T$	The end time of t
$m_t \in M, \quad t \in T$	The machine t uses
$tray_t \in Tr,\ t \in T$	The tray t uses. If none is used $tray_t = 0$
$fixture_t \in F,\ t \in T$	The fixture t uses. If none is used $fixture_t = 0$
$it_{t_1,t_2} \in \{0,1\},\ t_1, t_2 \in T$	If t_1 and t_2 are equal, then 1, otherwise 0
$cycle_t \in Cy,\ t \in T$	The cycle t is in
$identicalTasks_k \subset T,\ k \in \{1 \dots uniqueTasks\}$	Set of tasks that perform the same operation, but on different components
$componentsUsed_t \subset C,\ t \in T$	The components used at t
$componentCreated_t \in C,\ t \in T$	The component created at t
$mounting \subset T$	Set of tasks that mounts a component
$taking \subset T$	Set of tasks that takes a component
$moving \subset T$	Set of tasks that moves a component somewhere
$putting_c \subset putting,\ c \in C$	Set of tasks that puts component c somewhere
$mounting_c \subset mounting,\ c \in C$	Set of tasks that mounts component c
$taking_c \subset taking,\ c \in C$	Set of tasks that takes component c
$moving_c \subset moving,\ c \in C$	Set of tasks that moves component c somewhere
$concurrentTasks_k \subset T,\ k \in concurrentGroups$	Set of tasks needing concurrent execution

$order_k \subset T,\ k \in orderedGroups$	Array of tasks needed to be performed in a certain order on a single machine
$toolNeeded_t \in To,\ t \in T$	The tool needed for t
$changesTool \subset T$	A set of tasks that perform tool changes
$changesToTool_t \in To,\ t \in changesTool$	The tool that a task changes to
$defaultTool_m \in To,\ m \in M$	The tool set to each machine at the start.
$time_{t1,t2} \in \{0 \dots 2^{63} - 1\},\ t1, t2 \in T$	The time it takes to move from having performed $t1$ to be ready to perform $t2$
$usingMachine_t \in M,\ t \in T$	The machine t uses
$pred_t \in T,\ t \in T$	The task that is the predecessor to t
$moveS_t \in \{0 \dots 2^{63} - 1\},\ t \in T$	The start time for the move from $pred_t$ to t
$moveD_t = time_{pred_t,t},\ t \in T$	The duration for the move from $pred_t$ to t
$moveE_t = moveS_t + moveD_t,\ t \in T$	The end time for the move from $pred_t$ to t
$End = \{0 \dots D\},\ D = \sum\{d_t : \forall t \in T\}$	Marks the end of the schedule, i.e. makespan. The variable that is minimized.

Predicates

$noOverlap(t_1, t_2),\ t_1, t_2 \in T$ $\wedge\ noOverlap2(s_{t_1}, e_{t_1}, s_{t_2}, e_{t_2})$	t_1 does not overlap t_2
$noOverlap2(startTime_1, endTime_1, startTime_2, endTime_2),$ $startTime_1, startTime_2, endTime_1, endTime_2$ $\in \{s_t, d_t, e_t, moveS_t, moveD_t, moveE_t \mid \forall t \in T\}$ $\wedge\ endTime_1 \leq startTime_2 \vee startTime_1 \geq endTime_2$	The time interval $startTime_1$ to $endTime_1$ is not overlapped by the interval $startTime_2$ to $endTime_2$
$t_1 \prec t_2,\ t_1, t_2 \in T$ $\wedge\ e_{t_1} \leq moveS_{t_2} \wedge pred_{t_1} \neq t_2$	t_1 precedes t_2, i.e. t_1 has to end before t_2 starts
$task1PredecessorOfTask2(t_1, t_2),\ t_1, t_2 \in T$ $\wedge\ pred_{t_2} = t_1 \wedge pred_{t_1} \neq t_2$	t_1 is the predecessor of t_2, i.e. t_2 comes directly after t_1 on the same machine, no other task can com inbetween

Constraints

$End = max(\{e_t : \forall t \in T\})$	The end is where the last task ends
$\forall c \in C($ $\forall mountTask \in mounting_c($ $\forall putTask \in putting_c($ $putTask \prec mountTask)))$	If a set of tasks on a component involves a mount task and a put task, the put task has to come before the mount task
$\forall c \in \{c : c \in C \wedge putting_c = \emptyset \wedge moving_c = \emptyset\}($ $\forall mountTask \in mounting_c($ $\forall takeTask \in taking_c($ $task1PredecessorOfTask2(takeTask, mountTask))))$	If a set of tasks on a component involves a mount and a take task, but no move tasks or put tasks, the take task is the predecessor of the mount task
$\forall c \in \{c : c \in C \wedge putting_c = \emptyset \wedge moving_c \neq \emptyset\}($ $\forall mountTask \in mounting_c($	If a set of tasks on a component involves a mount task, a take task, and one or several moving tasks, but no put task, the take task has to come before the mount task, and the move tasks come in between

$\forall takeTaskintaking_c($ $takeTask \prec mountTask \wedge$ $pred_{mountTask} \neq takeTask \wedge$ $usingMachine_{takeTask} = usingMachine_{mountTask} \wedge$ $\forall moveTask \in moving_c($ $takeTask \prec moveTask \wedge moveTask \prec mountTask) \wedge$ $\forall t \in \{t : t \in T \setminus \{mountTask, takeTask\} \wedge t \notin moving\}($ $usingMachine_{takeTask} = usingMachine_{mountTask}$ $\rightarrow noOverlap2(s_t, e_t, s_{takeTask}, e_{mountTask}))))$	
$\forall c \in C($ $\forall putTask \in \{task : task \in putting_c \wedge tray_{putTask} = 0\}($ $\forall takeTask \in taking_c($ $task1PredecessorOfTask2(takeTask, putTask))))$	If a set of tasks on a component involves a put task in a tray and a take task, the take task ha to be the predecessor of the put task
$\forall group \in concurrentGroups($ $\forall t_1 \in concurrentTasks_{group}($ $\forall t_2 \in concurrentTasks_{group} \setminus \{t_1\}($ $s_{t_1} = s_{t_2} \wedge usingMachine_{t_1} \neq usingMachine_{t_2}$ $\wedge pred_{t_2} \neq t_1 \wedge pred_{t_1} \neq t_2)))$	Concurrent groups
$\forall t_1 \in \{t : t \in T \wedge componentCreated_t > 0\}($ $\forall t_2 \in \{t : t \in T \wedge componentCreated_{t_1} \in componentsUsed_{t_2}\}($ $t_1 \prec t_2))$	Components cannot be used before they are created
$\forall group \in orderedGroups($ $)$	Order FIXA
$\forall f \in fixtures($ $\forall t_1 \in \{t : t \in T \wedge fixture_t = f\}($	Two tasks on the same fixture can't overlap

$\forall t_2 \in \{t : t \in T \wedge fixture_t = f\} \setminus \{t_1\}($ $noOverlap(t_1, t_2))))$	
$\forall tr \in trays($ $\forall t_1 \in \{t : t \in T \wedge tray_t = tr\}($ $\forall t_2 \in \{t : t \in T \wedge tray_t = tr\} \setminus \{t_1\}($ $noOverlap(t_1, t_2))))$	Two tasks on the same tray can't overlap
$\forall t_1 \in T($ $\forall t_2 \in T \setminus \{t_1\}($ $usingMachine_{t_1} = usingMachine_{t_2} \rightarrow noOverlap(t_1, t_2)))$	Two tasks using the same machine cannot overlap
$cumulative(s, d, [1 : t \in T], nbrMachines)$	There can only be as many tasks performed simultaneously as there are machines
$\forall t \in T(pred_t \neq t)$	A task cannot be a predecessor to itself
$\forall t \in T(s_t \geq moveE_t)$	A task can only start after the move to it
$\forall t \in T(pred_t \neq 0 \rightarrow pred_t \prec t)$	Apart from the first tasks, a task has to start after its predecessor
$\forall t \in T(pred_t \neq 0 \rightarrow pred_{pred_t} \neq t)$	A task cannot be a predecessor to its predecessor
$alldifferent\_except\_0(pred)$	No two tasks can have the same predecessor. The exception is the start tasks. Since there are multiple machines, there can be multiple start tasks
$\forall t_1 \in T(\forall t_2 \in T \setminus \{t_1\}($ $usingMachine_{t_1} = usingMachine_{t_2} \rightarrow pred_{t_1} \neq pred_{t_2}))$	Tasks cannot have the same predecessors, except for start task which have 0 as predecessor. This together with $alldifferent\_except\_0$ helps set that constraint. But since the solver first determines on what machine a task will be performed, this constraint speeds up the solving process since the constraint is based on what machine is used
$\forall t \in T(pred_t = 0 \rightarrow moveS_t = 0)$	The start tasks have predecessor 0 and starts at time 0
$\forall t_1 \in T(\forall t_2 \in T \setminus \{t_1\}($	Two tasks using different machines cannot be each other predecessors

$usingMachine_{t_1} \neq usingMachine_{t_2} \rightarrow pred_{t_1} \neq t_2 \wedge pred_{t_2} \neq t_1))$	
$\forall t \in T(pred_t \neq 0 \rightarrow usingMachine_t = usingMachine_{pred_t})$	A task and its predecessor has to use the same machine
$member(pred, 0)$	There has to be at least one starting task
$\forall t \in T(pred_t \neq 0 \rightarrow \forall t_2 \in T($ $usingMachine_{t_2} = usingMachine_t \rightarrow noOverlap2(s_{t_2}, e_{t_2}, e_{pred_t}, s_t)))$	No other task on the same machine can come between t and its predecessor $pred_t$, i.e. no other task can overlap the timeslot inbetween
$\forall t \in T(moveD_t = times_{pred_t,t})$	Assigns the duration for the move between tasks
$\forall t_1 \in taking, \forall t_2 \in changesTool(pred_{t_2} \neq t_1)$	A tool-change cannot happen right after a take task, i.e. a tool-change cannot have a take task as it predecessor
$\forall t_1 \in \{t : t \in T \wedge toolsNeeded_{t_1} \neq 0\}($ $\exists t_2 \in \{t : t \in changesTool \wedge changesToTool_t = toolsNeeded_{t_1}\}($ $usingMachine_{t_1} = usignMachine_{t_2} \wedge t_2 \prec t_1) \vee$ $\exists m \in \{m : m \in M \wedge defaultTool_m = toolsNeeded_{t_1}\}($ $usingMachine_{t_1} = m \wedge s_{t_1} \geq 0))$	Tasks needing a tool has to use the same machine as a task switching to that tool
$\forall t_1 \in \{t : t \in T \wedge toolsNeeded_t \neq 0 \wedge$ $toolsChangedTo \setminus \{toolsNeeded_t\} \neq \emptyset\}($ $\exists t_2 \in \{t : t \in changesTool \wedge changesToTool_t \neq toolsNeeded_{t_1}\}($ $usingMachine_{t_1} = usingMachine_{t_2} \rightarrow t_1 \prec t_2))$	A task that needs a tool needs to happen before the change to another tool on the machine it is using
$\forall it \in identicalTasks($ $\forall t_1 \in it($ $\forall t_2 \in \{t : t \in it \wedge cycleNbr_{t_1} < cycleNbr_{t_2}\}($ $t_1 \prec t_2)))$	Identical tasks must come in the order of their cycles.