ＬＰとＣＰを併用した
スケジューリング

Dash　Optimization㈱
石村　猛

目次

- はじめに
- CP→MIP逐次解法
  例：Project Scheduling
  Ｑ１の定式化、実装、結果
  Ｑ２の定式化、実装、結果
- parallel MIP－CP解法
  例：装置割当てと順序付け
  定式化、実装、結果
- まとめ
- おわりに

はじめに

　複雑な問題では複数の解法を組合わせて使う必要がある。

　ここでは以下の２例について紹介する

- CP→MIP逐次解法
  - ＣＰの制約伝播をＬＰのpresolveとして使用
  - ＭＩＰでの最適解の探索
- parallel MIP－CP解法
  - ＣＰを、ＭＩＰのＢ－Ｂ過程におけるcut generation　として使用

CP→MIP逐次解法

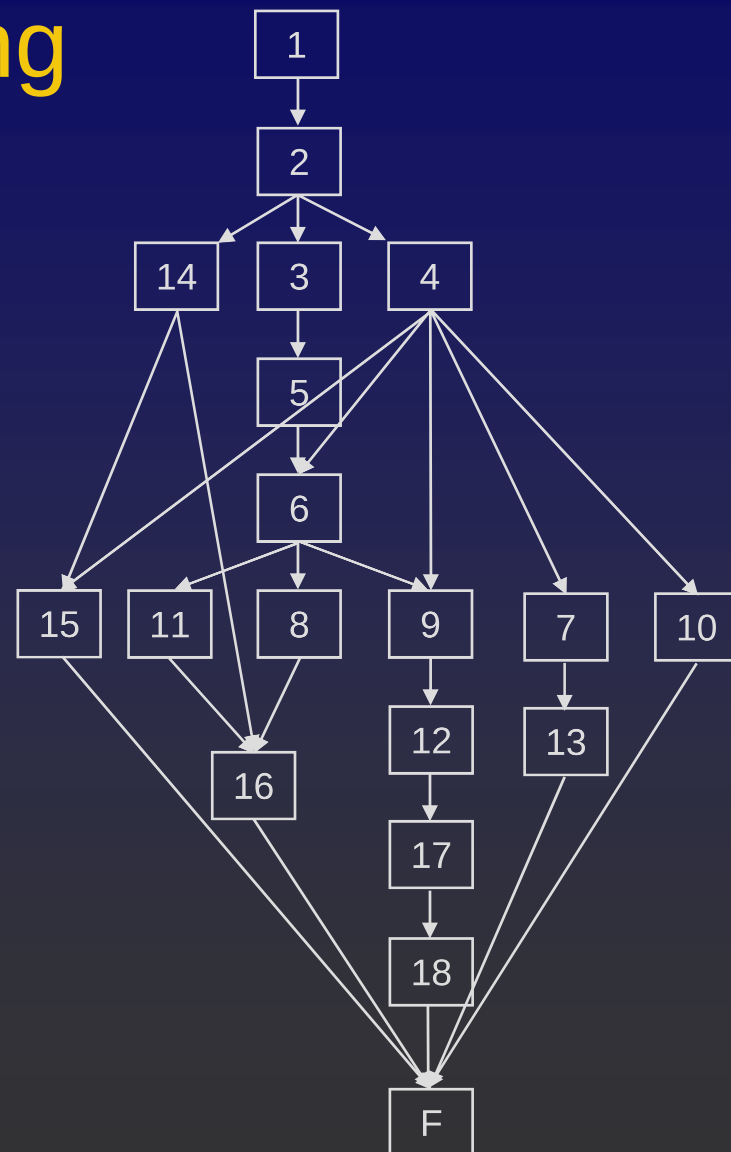
- ＣＰの制約伝播をＬＰのpresolveとして使用
- ＭＩＰでの最適解の探索

本問題は
‘Applications of optimization with Xpress-MP’
の Sec 7.1 を題材としております。

CP→MIP逐次解法

例：Project Scheduling

作業	工数	前作業	最大削減 可能工数	工数削減時 追加コスト
1	2	-	0	-
2	16	1	3	30
3	9	2	1	26
4	8	2	2	12
5	10	3	2	17
6	6	4,5	1	15
7	2	4	1	8
8	2	6	0	-
9	9	4,6	2	42
10	5	4	1	21
11	3	6	1	18
12	2	9	0	-
13	1	7	0	-
14	7	2	2	22
15	4	4,14	2	12
16	3	8,11,14	1	6
17	9	12	3	16
18	1	17	0	-



CP→MIP逐次解法

例：Project Scheduling

作業	工数	前作業	最大削減可能工数	工数削減時追加コスト
1	2	-	0	-
2	16	1	3	30
3	9	2	1	26
4	8	2	2	12
5	10	3	2	17
6	6	4,5	1	15
7	2	4	1	8
8	2	6	0	-
9	9	4,6	2	42
10	5	4	1	21
11	3	6	1	18
12	2	9	0	-
13	1	7	0	-
14	7	2	2	22
15	4	4,14	2	12
16	3	8,11,14	1	6
17	9	12	3	16
18	1	17	0	-

- Q1：
  最速完成は？

- Q2：
  追加コストを払って完成を早くすると、
  1単位あたり30のボーナス

  最も有利なスケジュールは？

dash optimization

CP→MIP逐次解法
Q1の定式化

$TASKS = \{1,\ldots,N\}$　　Nは完成を示す仮想タスク

$ARCS:$　$arc(i,j) \in ARCS$はタスクjを始める前にタスクiが完成

$DUR_i:$　タスクiの工期

$HORIZON:$工期の最大値$\left(=\sum_i DUR_i\right)$

$start_i:$　タスクiの開始時期

$\forall i \in TASKS : start_i \in \{0,\ldots,HORIZON\}$

$\forall (i,j) \in ARCS : start_i + DUR_i \le start_j$

dash optimization

CP→MIP逐次解法
Q1の実装-1

```
model "B-1 Stadium construction (CP)"
  uses "kalis"

  declarations
    N = 19 ! Number of tasks in the project
           ! (last = fictitious end task)
    TASKS = 1..N
    ARC: array(range,range) of integer ! Matrix of the adjacency graph
    DUR: array(TASKS) of integer ! Duration of tasks
    HORIZON : integer ! Time horizon

    start: array(TASKS) of cpvar ! Start dates of tasks
    bestend: integer
  end-declarations

  initializations from 'Data/b1stadium.dat'
    DUR ARC
  end-initializations

  HORIZON:= sum(j in TASKS) DUR(j)
```

CP→MIP逐次解法 Q1の実装-2

```
forall(j in TASKS) do
  0 <= start(j); start(j) <= HORIZON
end-do

! Task i precedes task j
forall(i, j in TASKS | exists(ARC(i, j))) do
  Prec(i,j):= start(i) + DUR(i) <= start(j)
  if not cp_post(Prec(i,j)) then
    writeln("Posting precedence ", i, "-", j, " failed")
    exit(1)
  end-if
end-do

! Since there are no side-constraints, the earliest possible completion
! time is the earliest start of the fictitiuous task N
bestend:= getlb(start(N))
start(N) <= bestend
writeln("Earliest possible completion time: ", bestend)

! For tasks on the critical path the start/completion times have been fixed
! by setting the bound on the last task. For all other tasks the range of
! possible start/completion times gets displayed.
forall(j in TASKS) writeln(j, ": ", start(j))

end-model
```

CP→MIP逐次解法

Q1の結果

Earliest possible completion time: 64
1: V001[0]
2: V002[2]
3: V003[18]
4: V004[18..29]
5: V005[27]
6: V006[37]
7: V007[26..61]
8: V008[43..59]
9: V009[43]
10: V010[26..59]
11: V011[43..58]
12: V012[52]
13: V013[28..63]
14: V014[18..53]
15: V015[26..60]
16: V016[46..61]
17: V017[54]
18: V018[63]
19: V019[64]

CP→MIP逐次解法

Q2の定式化　CPモデル

$TASKS = \{1,...,N\}$　　Nは完成を示す仮想タスク

$ARCS:$　$arc(i,j) \in ARCS$はタスクjを始める前にタスクiが完成

$DUR_i, MAXW_i:$　　タスクiの工期および最大削減期間

$start_i, duration_i:$　　タスクiの開始時期および工期

$$\forall i \in TASKS : start_i \in \{0,...,bestend\}$$

$$\forall i \in TASKS : duration_i \in \{DUR_i - MAXW_i,...,DUR_i\}$$

$$\forall (i,j) \in ARCS : start_i + duration_i \le start_j$$

dash optimization

CP→MIP逐次解法

Q2の定式化　MIPモデル

$start_i, save_i :$　　タスクiの開始時期および工期削減数

$$\forall (i, j) \in ARCS : start_i + DUR_i - save_i \le start_j$$

$$\forall i \in TASKS \setminus \{N\} : save_i \le MAXW_i$$

$$strat_N = bestend - save_N$$

$$\max imize \quad BOUNUS \bullet save_N - \sum_{i \in TASKS \setminus \{N\}} COST_i \bullet save_i$$

CP→MIP逐次解法
Ｑ２の実装

Ｑ２は以下の手順で実装される

- Ｑ１を解き、その解をbestendに保存
- Ｑ２のＣＰを解き、可能な開始･終了時期を取得
- 可能な開始･終了時期およびbestendもちいてＬＰモデルを作成　→　最適解を取得

CP→MIP逐次解法
Ｑ２の実装　ＣＰモデル-1

```
model "B-1 Stadium construction (CP submodel)"
  uses "kalis", "mmjobs"

  parameters
    MODE = 1                        ! Model version: 1 - fixed durations
                                    ! 2 - variable dur.
    HORIZON = 100                   ! Time horizon
  end-parameters

  declarations
    N = 19                          ! Number of tasks in the project
                                    ! (last = fictitious end task)
    TASKS = 1..N
    ARC: array(range,range) of integer ! Matrix of the adjacency graph
    DUR: array(TASKS) of integer       ! Duration of tasks
    MAXW: array(TASKS) of integer      ! Max. reduction of tasks (in weeks)

    start: array(TASKS) of cpvar       ! Start dates of tasks
    duration: array(TASKS) of cpvar    ! Durations of tasks
    lbstart,ubstart: array(TASKS) of integer ! Bounds on start dates of tasks
    EVENT_FAILED=2                     ! Event code sent by submodel
  end-declarations

  initializations from 'Data/b1stadium.dat'
    DUR ARC
  end-initializations
```

dash optimization

CP→MIP逐次解法
Ｑ２の実装　ＣＰモデル-2

```
forall(j in TASKS) setdomain(start(j), 0, HORIZON)

if MODE = 1 then                     ! **** Fixed durations
  ! Precedence relations between tasks
  forall(i, j in TASKS | exists(ARC(i, j))) do
    Prec(i,j):= start(i) + DUR(i) <= start(j)
    if not cp_post(Prec(i,j)) then
      send(EVENT_FAILED,0)
    end-if
  end-do

  ! Earliest poss. completion time = earliest start of the fictitiuous task N
  start(N) <= getlb(start(N))
else                                 ! **** Durations are variables
  initializations from 'Data/b1stadium.dat'
    MAXW
  end-initializations

  forall(j in TASKS) setdomain(duration(j), DUR(j)-MAXW(j), DUR(j))
```

- CP→MIP逐次解法
-
- Ｑ２の実装　ＣＰモデル-3

```
  ! Precedence relations between tasks
  forall(i, j in TASKS | exists(ARC(i, j))) do
    Prec(i,j):= start(i) + duration(i) <= start(j)
    if not cp_post(Prec(i,j)) then
      send(EVENT_FAILED,0)
    end-if
  end-do
 end-if

 ! Pass solution data to the master model
 forall(i in TASKS) do
  lbstart(i):= getlb(start(i)); ubstart(i):= getub(start(i))
 end-do

 initializations to "raw:"
  lbstart as "shmem:lbstart" ubstart as "shmem:ubstart"
 end-initializations

end-model
```

CP→MIP逐次解法
Ｑ２の実装　MIPモデル-1

```
model "B-1 Stadium construction (CP + LP) master model"
 uses "mmxprs", "mmjobs"

 forward procedure print_CP_solution(version: integer)

 declarations
  N = 19                          ! Number of tasks in the project (last = fictitious end task)
  TASKS = 1..N
  ARC: array(range,range) of integer ! Matrix of the adjacency graph
  DUR: array(TASKS) of integer       ! Duration of tasks
  BONUS: integer                     ! Bonus per week finished earlier
  MAXW: array(TASKS) of integer      ! Max. reduction of tasks (in weeks)
  COST: array(TASKS) of real         ! Cost of reducing tasks by a week
  lbstart,ubstart: array(TASKS) of integer ! Bounds on start dates of tasks
  HORIZON: integer                   ! Time horizon
  bestend: integer                   ! CP solution value

  CPmodel: Model                     ! Reference to the CP model
  msg: Event                         ! Termination message sent by submodel
 end-declarations

 initializations from ’Data/b1stadium.dat’
  DUR ARC MAXW BONUS COST
 end-initializations

 HORIZON:= sum(o in TASKS) DUR(o)
```

dash optimization

CP→MIP逐次解法

Ｑ２の実装　ＭＩＰモデル-2

```
! **** First CP model ****

  res:= compile("b1stadium_sub.mos")   ! Compile the CP model
  load(CPmodel, "b1stadium_sub.bim")   ! Load the CP model
  run(CPmodel, "MODE=1,HORIZON=" + HORIZON) ! Solve first version of CP model
  wait                                 ! Wait until subproblem finishes
  msg:= getnextevent                   ! Get the termination event message
  if getclass(msg)<>EVENT_END then     ! Check message type
    writeln("Submodel 1 is infeasible")
    exit(1)
  end-if

  initializations from "raw:"
    lbstart as "shmem:lbstart" ubstart as "shmem:ubstart"
  end-initializations

  bestend:= lbstart(N)
  print_CP_solution(1)
```

CP→MIP逐次解法
Q2の実装 MIPモデル-3

```
! **** Second CP model ****

  run(CPmodel, "MODE=2,HORIZON=" + bestend) ! Solve second version of CP
     model
  wait                                 ! Wait until subproblem finishes
  msg:= getnextevent                   ! Get the termination event message
  if getclass(msg)<>EVENT_END then     ! Check message type
    writeln("Submodel 2 is infeasible")
    exit(2)
  end-if

  ! Retrieve solution from memory
  initializations from "raw:"
    lbstart as "shmem:lbstart" ubstart as "shmem:ubstart"
  end-initializations

  print_CP_solution(2)
```

- CP→MIP逐次解法
-
- Ｑ２の実装　ＭＩＰモデル-4

```
! **** LP model for second problem ****
 declarations
  start: array(TASKS) of mpvar       ! Start times of tasks
  save: array(TASKS) of mpvar        ! Number of weeks finished early
 end-declarations

 ! Objective function: total profit
 Profit:= BONUS*save(N) - sum(i in 1..N-1) COST(i)*save(i)

 ! Precedence relations between tasks
 forall(i,j in TASKS | exists(ARC(i,j)))
  Precm(i,j):= start(i) + DUR(i) - save(i) <= start(j)

 ! Total duration
 start(N) + save(N) = bestend

 ! Limit on number of weeks that may be saved
 forall(i in 1..N-1) save(i) <= MAXW(i)

 ! Bounds on start times deduced by constraint propagation
 forall(i in 1..N-1) do
  lbstart(i) <= start(i); start(i) <= ubstart(i)
 end-do
```

dash optimization

CP→MIP逐次解法

Ｑ２の実装　ＭＩＰモデル-5

```
  ! Solve the second problem: maximize the total profit
  setparam("XPRS_VERBOSE", true)
  setparam("XPRS_PRESOLVE", 0)        ! We use constraint propagation as preprocessor
  maximize(Profit)

  ! Solution printing
  writeln("Total profit: ", getsol(Profit))
  writeln("Total duration: ", getsol(start(N)), " weeks")
  forall(i in 1..N-1)
    write(strfmt(i,2), ": ", strfmt(getsol(start(i)),-3),
  if(i mod 6 = 0,"¥n",""))
  writeln

!********************************************************************
procedure print_CP_solution(version: integer)
  writeln("CP solution (version ", version, "):")
  writeln("Earliest possible completion time: ", lbstart(N), " weeks")
  forall(i in 1..N-1)
    write(i, ": ", lbstart(i), if(lbstart(i)<ubstart(i), "-"+ubstart(i), ""),
    if(i mod 6 = 0, "¥n", ", "))
end-procedure

end-model
```

CP→MIP逐次解法

Q2の結果-1

CP solution (version 1):
Earliest possible completion time: 64 weeks
1: 0, 2: 2, 3: 18, 4: 18-29, 5: 27, 6: 37
7: 26-61, 8: 43-59, 9: 43, 10: 26-59, 11: 43-58, 12: 52
13: 28-63, 14: 18-53, 15: 26-60, 16: 46-61, 17: 54, 18: 63
CP solution (version 2):
Earliest possible completion time: 52 weeks
1: 0-12, 2: 2-14, 3: 15-27, 4: 15-37, 5: 23-35, 6: 31-43
7: 21-62, 8: 36-60, 9: 36-48, 10: 21-60, 11: 36-60, 12: 43-55
13: 22-63, 14: 15-57, 15: 21-62, 16: 38-62, 17: 45-57, 18: 51-63

CP→MIP逐次解法

Ｑ２の結果-2

```
Reading Problem ¥xprs_f1c_e30008
Problem Statistics
          28 (      0 spare) rows
          38 (      0 spare) structural columns
          83 (      0 spare) non-zero elements
Global Statistics
           0 entities        0 sets        0 set members

   Its         Obj Value      S   Ninf  Nneg        Sum Inf  Time
     0        360.000300      D     17      0      29.000010     0
    16         87.000000      D      0      0        .000000     0
Optimal solution found
Total profit: 87
Total duration: 54 weeks
 1: 0   2: 2   3: 15  4: 15  5: 23  6: 31
 7: 23  8: 36  9: 36 10: 23 11: 36 12: 45
13: 25 14: 15 15: 23 16: 39 17: 47 18: 53
```

parallel MIP－CP解法

- ＣＰを、ＭＩＰの Branch-and-Bound 過程における cut generation として使用

本問題は

V. Jain and I.E. Grossmann. Algorithms for hybrid MILP/CLP models for a class of optimization problems.

INFORMS J. Computing, 13(4):258—276, 2001.

を題材としております。

- parallel MIP－CP解法
- 例：装置割当てと順序付け

製品	製造コスト			製造時間			作成開始可能日	納期
	装置1	装置2	装置3	装置1	装置2	装置3		
1	12	6	7	10	14	13	2	32
2	13	6	10	7	9	8	4	33
3	10	4	6	11	17	15	5	36
4	8	4	5	6	9	12	7	37
5	12	6	7	4	6	10	9	39
6	10	5	6	2	3	4	0	34
7	7	4	5	10	15	16	3	30
8	9	5	5	8	11	12	6	26
9	10	5	7	10	14	13	11	36
10	8	4	5	8	11	14	2	38
11	15	8	9	9	12	16	3	31
12	13	7	7	3	5	6	4	22

製造時間、期間等の制約を満たした上で
コスト最小となる計画を作成

parallel MIP－CP解法

MIPモデルの定式化

$PRODS = \{1,\ldots,12\}, MACHS = \{1,2,3\}$　　それぞれ製品と装置の集合

$COST_{pm}, DUR_{pm}$:　　製品ｐ装置ｍの製造コストおよび製造期間

use_{pm} :　　製品ｐを装置ｍで製造した場合１、さもなくば０

$$\forall p \in PRODS : \sum_{m \in MACH} use_{pm} = 1$$

$$\min imize \quad \sum_{\substack{p \in PRODS \\ m \in MACH}} \mathrm{COST}_{pm} \bullet use_{pm}$$

parallel MIP－CP解法
ＣＰモデルの定式化

$PRODS = \{1,\ldots,12\}, MACHS = \{1,2,3\}$　それぞれ製品と装置の集合

$\mathrm{Pr}\,odMach_m \left(\mathrm{Pr}\,odMach_m \subseteq PRODS\right)$　装置mに割り当てられた製造

DUR_{pm} :　製品ｐを装置mで製造する際の製造期間

REL_p, DUE_p :　製品ｐの製造開始可能日および納期

$start_p$:　製品ｐの製造開始日

$$\forall p \in \mathrm{Pr}\,odMach_m : start_p \in \{REL_p,\ldots,DUE_p - DUR_{pm}\}$$

$$\forall p,q \in \mathrm{Pr}\,odMach_m, p < q : start_p + DUR_{pm} \le start_q \vee start_q + DUR_{qm} \le start_p$$

parallel MIP－CP解法 実装

```
Define the MIP machine assignment problem.
Define the operations of the CP model.
Start the MIP Branch-and-Bound search.
At every node of the MIP search:
    while function generate_cuts returns true
        re-solve the LP-relaxation

Function generate_cuts
    for all machines m call generate_cut_machine(m)
    if at least one cut has been generated
        Return true
    otherwise
        Return false

Function generate_cut_machine(m)
    Collect all operations assigned to machine m
    if more than one operation assigned to m
        Solve the CP sequencing problem for m
        if sequencing succeeds
            Save the solution
        otherwise
            Add an infeasibility cut for machine m to the MIP
```

parallel MIP－CP解法
実装　MIPモデル-1

```
model "Schedule (MIP + CP) master problem"
  uses "mmsystem", "mmxprs", "mmjobs"

  parameters
    DATAFILE = "Data/sched_3_12.dat"
    VERBOSE = 1
  end-parameters

  forward procedure define_MIP_model
  forward procedure setup_cutmanager
  forward public function generate_cuts: boolean
  forward public procedure print_solution

  declarations
    NP: integer                        ! Number of operations (products)
    NM: integer                        ! Number of machines
  end-declarations

!  initializations from DATAFILE
!    NP NM
!  end-initializations
   NP := 12
   NM :=  3

  declarations
    PRODS = 1..NP                      ! Set of products
    MACH = 1..NM                       ! Set of machines
```

dash optimization

parallel MIP－CP解法
実装　MIPモデル-2

```
  REL: array(PRODS) of integer       ! Release dates of orders
  DUE: array(PRODS) of integer       ! Due dates of orders
  MAX_LOAD: integer                  ! max_p DUE(p) - min_p REL(p)
  COST: array(PRODS,MACH) of integer ! Processing cost of products
  DUR: array(PRODS,MACH) of integer  ! Processing times of products
  starttime: real                    ! Measure program execution time
  ctcut: integer                     ! Counter for cuts
  solstart: array(PRODS) of integer
                                     ! **** MIP model:
  use: array(PRODS,MACH) of mpvar    ! 1 if p uses machine m, otherwise 0
  Cost: linctr ! Objective function

  totsolve,totCP: real               ! Time measurement
  ctrun: integer                     ! Counter of CP runs
  CPmodel: Model                     ! Reference to the CP sequencing model
  ev: Event                          ! Event
  EVENT_SOLVED=2                     ! Event codes sent by submodels
  EVENT_FAILED=3
end-declarations

! Read data from file
initializations from DATAFILE
  REL DUE COST DUR
end-initializations
```

parallel MIP－CP解法
実装　MIPモデル-3

```
! **** Problem definition ****
define_MIP_model ! Definition of the MIP model
res:=compile("sched_sub.mos")        ! Compile the CP model
load(CPmodel, "sched_sub.bim")       ! Load the CP model

! **** Solution algorithm ****
starttime:= gettime
setup_cutmanager                     ! Settings for the MIP search

totsolve:= 0.0
initializations to "raw:"
  totsolve as "shmem:solvetime"
  REL as "shmem:REL" DUE as "shmem:DUE"
end-initializations

minimize(Cost)                       ! Solve the problem

writeln("Number of cuts generated: ", ctcut)
writeln("(", gettime-starttime, "sec) Best solution value: ", getobjval)
initializations from "raw:"
  totsolve as "shmem:solvetime"
end-initializations
writeln("Total CP solve time: ", totsolve)
writeln("Total CP time: ", totCP)
writeln("CP runs: ", ctrun)
```

parallel MIP－CP解法
実装　MIPモデルの定義

```
procedure define_MIP_model

  ! Objective: total processing cost
  Cost:= sum(p in PRODS, m in MACH) COST(p,m) * use(p,m)

  ! Each order needs exactly one machine for processing
  forall(p in PRODS) sum(m in MACH) use(p,m) = 1

  ! Valid inequalities for strengthening the LP relaxation
  MAX_LOAD:= max(p in PRODS) DUE(p) - min(p in PRODS) REL(p)
  forall(m in MACH) sum(p in PRODS) DUR(p,m) * use(p,m) <= MAX_LOAD

  forall(p in PRODS, m in MACH) use(p,m) is_binary

end-procedure
```

parallel MIP－CP解法

実装　MIPモデル(Cut Generation-1)

```
!--------------------------------------------------------------
! Cut generation callback function
public function generate_cuts: boolean
  returned:=false; ctcutold:=ctcut

  setparam("XPRS_solutionfile", 0)
  forall(m in MACH) do
    if generate_cut_machine(m) then
      returned:=true ! Call function again for this node
      ctcut+=1
    end-if
  end-do
  setparam("XPRS_solutionfile", 1)
  if returned and VERBOSE>1 then
    writeln("Node ", getparam("XPRS_NODES"), ": ", ctcut-ctcutold,
            " cut(s) added")
  end-if

end-function
```

parallel MIP－CP解法
実装　MIPモデル(Cut Generation-2)

```
!---------------------------------------------------------------
! Generate a cut for machine m if the sequencing subproblem is infeasible
function generate_cut_machine(m: integer): boolean
  declarations
    ProdMach: set of integer
  end-declarations

  ! Collect the operations assigned to machine m
  products_on_machine(m, ProdMach)

  ! Solve the sequencing problem (CP model): if solved, save the solution,
  ! otherwise add an infeasibility cut to the MIP problem
  size:= getsize(ProdMach)
  returned:= false
  if (size>1) then
    if not solve_CP_problem(m, ProdMach, 1) then
      Cut:= sum(p in ProdMach) use(p,m) - (size-1)
      if VERBOSE > 2 then
        writeln(m,": ", ProdMach, " <= ", size-1)
      end-if
      addcut(1, CT_LEQ, Cut)
      returned:= true
    end-if
  end-if

end-function
```

parallel MIP－CP解法

実装　MIPモデル(Cut Generation-3)

```
procedure products_on_machine(m: integer, ProdMach: set of integer)

  forall(p in PRODS) do
    val:=getsol(use(p,m))
    if (val > 0 and val < 1) then
      ProdMach:={}
      break
    elif val>0.5 then
      ProdMach+={p}
    end-if
  end-do
end-procedure
```

parallel MIP－CP解法
実装 MIPモデル(CP call -1)

```
function solve_CP_problem(m: integer, ProdMach: set of integer,
                          mode: integer): boolean
  declarations
    DURm: array(range) of integer
    sol: array(range) of integer
    solvetime: real
  end-declarations

  ! Data for CP model
  forall(p in ProdMach) DURm(p):= DUR(p,m)
  initializations to "raw:"
    ProdMach as "shmem:ProdMach"
    DURm as "shmem:DURm"
  end-initializations

  ! Solve the problem and retrieve the solution if it is feasible
  startsolve:= gettime
  returned:= false
  if(getstatus(CPmodel)=RT_RUNNING) then
    fflush
    writeln("CPmodel is running")
    fflush
    exit(1)
  end-if
```

parallel MIP－CP解法
実装　MIPモデル(CP call -2)

```
ctrun+=1
run(CPmodel, "NP=" + NP + ",VERBOSE=" + VERBOSE + ",MODE=" + mode)
wait                              ! Wait for a message from the submodel
ev:= getnextevent                 ! Retrieve the event
if getclass(ev)=EVENT_SOLVED then
  returned:= true
  if mode = 2 then
    initializations from "raw:"
      sol as "shmem:solstart"
    end-initializations
    forall(p in ProdMach) solstart(p):=sol(p)
  end-if
elif getclass(ev)<>EVENT_FAILED then
  writeln("Problem with Kalis")
  exit(2)
end-if
wait
dropnextevent                     ! Ignore "submodel finished" event
totCP+= (gettime-startsolve)

end-function
```

parallel MIP－CP解法
実装　MIPモデル(Cut Managerの定義)

```
procedure setup_cutmanager
  setparam("XPRS_CUTSTRATEGY", 0) ! Disable automatic cuts
  feastol:= getparam("XPRS_FEASTOL") ! Get Optimizer zero tolerance
  setparam("zerotol", feastol * 10) ! Set comparison tolerance of Mosel
  setparam("XPRS_PRESOLVE", 0) ! Disable presolve
  setparam("XPRS_MIPPRESOLVE", 0) ! Disable MIP presolve
  command("KEEPARTIFICIALS=0") ! No global red. cost fixing
  setparam("XPRS_SBBEST", 0) ! Turn strong branching off
  setparam("XPRS_HEURSTRATEGY", 0) ! Disable MIP heuristics
  setparam("XPRS_EXTRAROWS", 10000) ! Reserve space for cuts
  setparam("XPRS_CPMAXELEMS", NP*30000) ! ... and cut coefficients
  setcallback(XPRS_CB_CM, "generate_cuts") ! Define the cut manager callback
  setcallback(XPRS_CB_UIS, "print_solution")! Define the integer solution cb.
  setparam("XPRS_COLORDER", 2)
  case VERBOSE of
  1: do
    setparam("XPRS_VERBOSE", true)
    setparam("XPRS_MIPLOG", -200)
    end-do
  2: do
    setparam("XPRS_VERBOSE", true)
    setparam("XPRS_MIPLOG", -100)
    end-do
  3: do ! Detailed MIP output
    setparam("XPRS_VERBOSE", true)
    setparam("XPRS_MIPLOG", 3)
    end-do
  end-case
end-procedure
```

parallel MIP－CP解法

実装　MIPモデル (結果出力-1)

```
public procedure print_solution
  declarations
    ProdMach: set of integer
  end-declarations

  writeln("(",gettime-starttime, "sec) Solution ",
          getparam("XPRS_MIPSOLS"), ": Cost: ", getsol(Cost))

  if VERBOSE > 1 then
    forall(p in PRODS) do
      forall(m in MACH) write(getsol(use(p,m))," ")
      writeln
    end-do
  end-if
```

parallel MIP－CP解法
実装　MIPモデル (結果出力-2)

```
if VERBOSE > 0 then
  forall(m in MACH) do
    ProdMach:= {}
    ! Collect the operations assigned to machine m
    products_on_machine(m, ProdMach)
    Size:= getsize(ProdMach)
    if Size > 1 then
      ! (Re)solve the CP sequencing problem
      if not solve_CP_problem(m, ProdMach, 2) then
        writeln("Something wrong here: ", m, ProdMach)
      end-if
    elif Size=1 then
      elem:=min(p in ProdMach) p
      solstart(elem):=REL(elem)
    end-if
  end-do

  ! Print out the result
  forall(p in PRODS) do
    msol:=sum(m in MACH) m*getsol(use(p,m))
    writeln(p, " -> ", msol,": ", strfmt(solstart(p),2), " - ",strfmt(DUR(p,round(msol))+solstart(p),2), " [",
            REL(p), ", ", DUE(p), "]")
  end-do
  writeln("Time: ", gettime - starttime, "sec")
  writeln
  fflush
end-if
end-procedure
end-model
```

parallel MIP－CP解法
実装　ＣＰモデル -1

```
model "Schedule (MIP + CP) CP subproblem"
  uses "kalis", "mmjobs" , "mmsystem"

  parameters
    VERBOSE = 1
    NP = 12                              ! Number of products
    MODE = 1                                ! 1 - decide feasibility
                                         ! 2 - return complete solution
  end-parameters

  startsolve:= gettime

  declarations
    PRODS = 1..NP ! Set of products
    ProdMach: set of integer
  end-declarations

  initializations from "raw:"
    ProdMach as "shmem:ProdMach"
  end-initializations

  finalize(ProdMach)
```

parallel MIP－CP解法
実装　CPモデル -2

```
declarations
  REL: array(PRODS) of integer          ! Release dates of orders
  DUE: array(PRODS) of integer          ! Due dates of orders
  DURm: array(ProdMach) of integer      ! Processing times on machine m
  solstart: array(ProdMach) of integer  ! Solution values for start times

  start: array(ProdMach) of cpvar       ! Start times of tasks
  Disj: array(range) of cpctr           ! Disjunctive constraints
  Strategy: array(range) of cpbranching ! Enumeration strategy
  EVENT_SOLVED=2                        ! Event codes sent by submodels
  EVENT_FAILED=3
  solvetime: real
end-declarations

initializations from "raw:"
  DURm as "shmem:DURm" REL as "shmem:REL" DUE as "shmem:DUE"
end-initializations

! Bounds on start times
forall(p in ProdMach) setdomain(start(p), REL(p), DUE(p)-DURm(p))
```

dash optimization

parallel MIP－CP解法
実装　ＣＰモデル -3

```
! Disjunctive constraint
ct:= 1
forall(p,q in ProdMach| p<q) do
  Disj(ct):= start(p) + DURm(p) <= start(q) or start(q) + DURm(q) <= start(p)
  ct+= 1
end-do

! Post disjunctions to the solver
nDisj:= ct; j:=1; res:= true
while (res and j<nDisj) do
  res:= cp_post(Disj(j))
  j+=1
end-do

! Solve the problem
if res then
  Strategy(1):= settle_disjunction(Disj)
  Strategy(2):= assign_and_forbid(KALIS_SMALLEST_DOMAIN, KALIS_MIN_TO_MAX,
                                start)
  cp_set_branching(Strategy)
  res:= cp_find_next_sol
end-if
```

parallel MIP－CP解法
実装　CPモデル -4

```
! Pass solution to master problem
if res then
  forall(p in ProdMach) solstart(p):= getsol(start(p))
  if MODE=2 then
    initializations to "raw:"
      solstart as "shmem:solstart"
    end-initializations
  end-if
  send(EVENT_SOLVED,0)
else
  send(EVENT_FAILED,0)
end-if

! Update total running time measurement
initializations from "raw:"
  solvetime as "shmem:solvetime"
end-initializations
solvetime+= gettime-startsolve
initializations to "raw:"
  solvetime as "shmem:solvetime"
end-initializations

end-model
```

parallel MIP－CP解法

結果

Solution 1: Cost: 98
1 -> 3: 2 - 15 [2, 32]
2 -> 3: 15 - 23 [4, 33]
3 -> 1: 25 - 36 [5, 36]
4 -> 2: 24 - 33 [7, 37]
5 -> 2: 33 - 39 [9, 39]
6 -> 1: 0 - 2 [0, 34]
7 -> 1: 3 - 13 [3, 30]
8 -> 2: 13 - 24 [6, 26]
9 -> 3: 23 - 36 [11, 36]
10 -> 2: 2 - 13 [2, 38]
11 -> 1: 16 - 25 [3, 31]
12 -> 1: 13 - 16 [4, 22]
Time: 2.684sec

Solution 2: Cost: 92
1 -> 3: 2 - 15 [2, 32]
2 -> 3: 15 - 23 [4, 33]
3 -> 2: 15 - 32 [5, 36]
4 -> 1: 24 - 30 [7, 37]
5 -> 2: 32 - 38 [9, 39]
6 -> 2: 0 - 3 [0, 34]
7 -> 1: 3 - 13 [3, 30]
8 -> 1: 16 - 24 [6, 26]
9 -> 3: 23 - 36 [11, 36]
10 -> 1: 30 - 38 [2, 38]
11 -> 2: 3 - 15 [3, 31]
12 -> 1: 13 - 16 [4, 22]
Time: 4.337sec

まとめ

ＬＰとＣＰを併用したスケジューリング例として
•CP→MIP逐次解法
•parallel MIP－CP解法
２つの事例を紹介

　このようなハイブリッドアルゴリズムは、個別のケース毎に、開発、実施、テストする必要がある。

↓

　開発努力に対する少なからぬ投資、そして解決手法や用いるツールに対する深い理解が必要である。

dash optimization

おわりに

本発表は

Hybrid MIP/CP solving
with Xpress-Optimizer and Xpress-Kalis

S. Heipcke

http://www.dashoptimization.com/home/products/products_kalis.html

を基に行いました。