Combinatorial Optimization Models in Automated Manufacturing

Yves Crama

HEC Management School
University of Liège

Francqui Lecture, KUL, March 2010

Based on joint work with Hakan Gultekin, Antoon Kolen, Alwin Oerlemans, Frits Spieksma, Joris van de Klundert, etc.

Outline

Automated manufacturing systems

Keywords: numerically controlled (NC, CNC) machines, flexible manufacturing systems (FMS), robots, ...

Main feature: flexibility

- variety of operations,
- rapid changes of tools,
- flexible material handling systems.

Particularly well adapted for production of small series.
But also used for automation of repetitive manufacturing.

Industrial applications

Applications include: laser, flame and plasma cutting, welding, bending, spinning, pinning, gluing, fabric cutting, sewing, tape and fiber placement, routing, picking and placing, sawing, etc.

Large variety of situations $\Rightarrow$ large variety of models.

Focus here on two types of situations:

Models:

- tool management for NC machines
- flexible assembly lines

- special, reasonably “clean” types of combinatorial machine scheduling problems
- connexions with many other problems and structures.

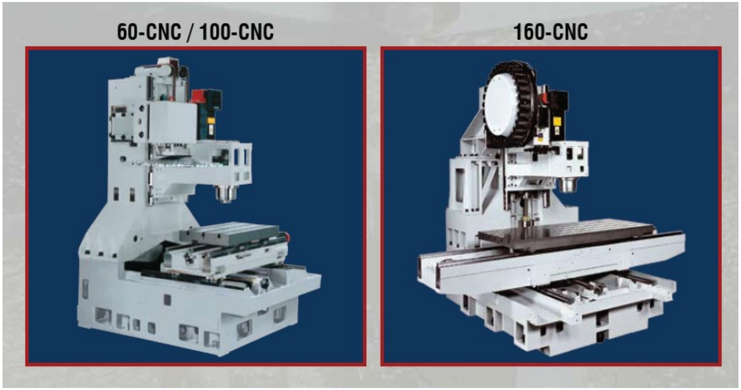
Tool management problems

Main features

- automated tool interchanging device
- can switch tools very fast between the tool magazine (carousel) and the workhead
- allows each machine to perform various operations with very small setup times.

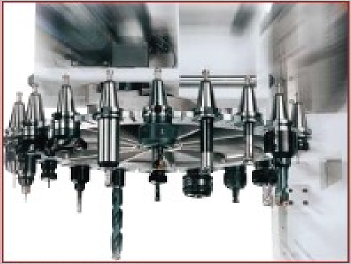
CNC machine

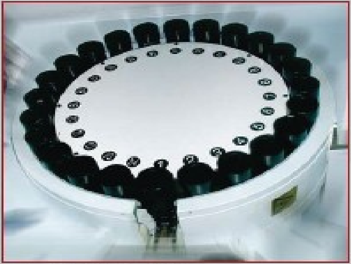
Vertical machining centers



Tool carousels

Changeur d'outils – *Werkzeugwechsler* – Tool changer





60-CNC / 100-CNC

Type carrousel 20 outils (Option 40 outils)
Karussell Typ 20 Werkzeuge (Option 40 Werkzeuge)
Carrousel type 20 tools (Option 40 tools)

Vertical – *Vertikaler* – Swing arm

	Standard	Option
60-CNC / 100-CNC:	-	24
160-CNC:	30	40

Tool changes

Videos:

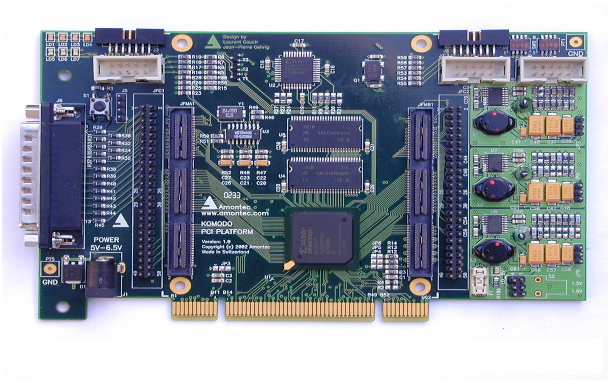
`http://www.metacafe.com/watch/2649895/limac_r4000atc_series_cnc_router/`

`http://www.youtube.com/watch?v=HSsKVHrMg2E`

`http://www.youtube.com/watch?v=ezTdFDD5db0`

PCB assembly

Similar situations arise in printed circuit board assembly.



`http://www.youtube.com/watch?v=g2TXhqCq9-c`

Setup minimization

Tool setup minimization:

- limited size of the tool magazine (say, 10 to 120 tools)
- many more tools may be stored in a central storage area
- transferred to the machines as required
- costly, slow, error-prone operations

One-machine scheduling with tooling decisions:

Simultaneously

- sequence parts to be processed and
- allocate tools required to the machine

so as to minimize tool setup costs.

Setup minimization

Various models for setup minimization.

Common data

- M: number of tools;
- N : number of parts;
- A : $M \times N$ tool-part matrix:
  $a_{ij} = 1$ if part j requires tool i, 0 otherwise;
- C : capacity of the tool magazine (= number of tool slots)

Feasible batch

A batch of parts is feasible if it can be carried out without tool switches, i.e., if it requires at most C tools.

Example

Capacity: $C = 3$

Parts

Tools

	P_1	P_2	P_3	P_4	P_5
T_1	1	0	1	0	1
T_2	1	0	0	1	0
T_3	0	1	1	1	0
T_4	0	1	0	0	1

A basic model

Batch selection

Find a feasible batch of maximum cardinality.

Equivalently:

Batch selection

Find a largest subset of columns of the tool-part incidence matrix such that the submatrix induced by this subset has at most C nonzero rows.

or...

Batch selection

Given a hypergraph, find a subset of C vertices that contains the largest possible number of hyperedges.

Example

Capacity: $C = 3$

Tools / **Parts**

	P_1	P_2	P_3	P_4	P_5
T_1	1	0	1	0	1
T_2	1	0	0	1	0
T_3	0	1	1	1	0
T_4	0	1	0	0	1

Complexity

Theorem

Batch selection is NP-hard.

Generalization of maximum clique.

Many papers on this problem:

- integer programming
- heuristics
- special “graphical” case
- subproblem for *part grouping problem*: partition parts into small number of feasible batches (minimize setups).

Heuristics

Problem is hard $\longrightarrow$ heuristics are appealing.

Mostly based on priority lists: selection rule is used iteratively to add parts to the current batch, as long as magazine capacity allows.

Empirically, most successful selection rules rest on variants of the Maximal Intersection rule:

Maximal Intersection rule

Among the parts not yet selected, select one that shares the maximum number of common tools with the parts already in the batch.

Worst-case ratio

From a theoretical point of view, one may ask:

Worst-case ratio

What is the (theoretical) worst-case ratio of heuristics for the batch selection problem, where:

$$wcr = \frac{\textit{optimal value}}{\textit{heuristic value}}?$$

Analysis

Observe:

- every heuristic puts at least one part in the batch
- every feasible batch contains at most $\binom{C}{[C/2]}$ parts.

Worst-case ratio

For every heuristic,

$$wcr = \frac{\textit{optimal value}}{\textit{heuristic value}} \leq \binom{C}{[C/2]} = \Omega(\frac{2^C}{\sqrt{C}})$$

Crama and van de Klundert (1999) proved:

Theorem

The worst-case ratio of the list-processing heuristic based on the Maximal Intersection selection rule is of the order of $\binom{C}{[C/2]}$.

Analysis

Several other heuristics have similar worst-case performance ratios.

Question

What is the best worst-case ratio attainable by a polynomial-time approximation algorithm for batch selection?

Analysis

Crama and van de Klundert (1999) proved:

Theorem

If there is a polynomial-time approximation algorithm with constant worst-case ratio for batch selection, then there is also a polynomial-time approximation scheme for this problem.

Conjecture

There exists no polynomial-time approximation algorithm with constant worst-case ratio for batch selection, unless P = NP.

Perhaps even true:

Conjecture

There exists no polynomial-time approximation algorithm with worst-case ratio $O(poly(C))$ for batch selection, unless P = NP.

Model

Recall:

- when a sequence of parts is processed, tools need to be loaded on the machine between certain pairs of successive operations
- (and other tools need to be removed in order to create space).

Tool switching problem

Determine a part input sequence and an associated sequence of tool loadings such that the total number of tool switches is minimized.

Example

Capacity: $C = 3$

Tools / **Parts**

	P_1	P_2	P_3	P_4	P_5
T_1	1	0	1	0	1
T_2	1	0	0	1	0
T_3	0	1	1	1	0
T_4	0	1	0	0	1

The part sequence $P_1, P_2, \ldots, P_5$ requires 3 tool switches.

Example

Capacity: $C = 3$

Tools / **Parts**

	P_1	P_2	P_3	P_4	P_5
T_1	1	0	1	0	1
T_2	1	0	0	1	0
T_3	0	1	1	1	0
T_4	0	1	0	0	1

The part sequence $P_1, P_2, \ldots, P_5$ requires 3 tool switches.

Analysis

Note: two distinct subproblems

- sequence the parts
- given the part sequence, optimize tool loadings.

Concentrate on

Tool loading problem

Given a part sequence, find the sequence of tool loadings that entails the smallest number of tool switches.

Note: Related problems in computer memory management ("paging problem") or Web caching applications.

Some old results

- An optimal tooling can be computed in time $O(MN)$ by the KTNS (Keep Tool Needed Soonest) algorithm (Belady 1966; Tang, Denardo 1988).
- Variant with tool-dependent setup times can be formulated as an integer programming problem whose constraint matrix has the *consecutive ones property* (Crama, Kolen, Oerlemans, Spieksma 1994)
  - reducible to a network max flow problem;
  - yields KTNS when all setup times are equal.
- More general version with tool-dependent switching times can be directly formulated as a network max flow problem (Finke, Privault 1993).

More recent results

Previous results apply when all tools occupy exactly one slot.
What if some tools require more slots?
Crama, Moonen, Spieksma and Talloen (2007):

Theorem

Tool loading with arbitrary tool sizes is strongly NP-hard, even with unit switching costs.

Theorem

Tool loading with arbitrary tool sizes can be expressed as a shortest path problem on a graph with $O(M^C C!)$ nodes. (Polynomial for fixed C.)

Back to tool switching

What if the part sequence is not fixed?

- The tool switching problem is NP-hard even when $C = 2$. (Reduction from *VLSI gate matrix permutation problem*.)
- Many heuristics, but very hard to solve exactly. (Difficult to compute tight lower bounds on the optimal value.)
- Very possible that, here again:

Conjecture

There exists no polynomial-time approximation algorithm with worst-case ratio $O(poly(C))$ for the tool switching problem, unless P = NP.

Industrial robots

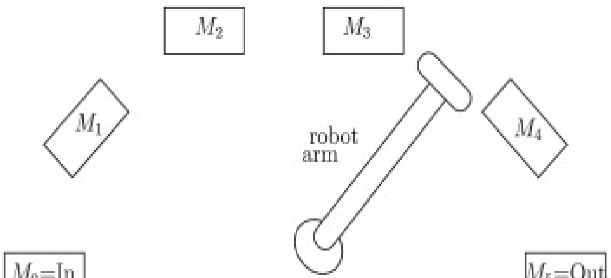
Main features

- can perform a broad variety of tasks
- can be used as flexible material handling devices

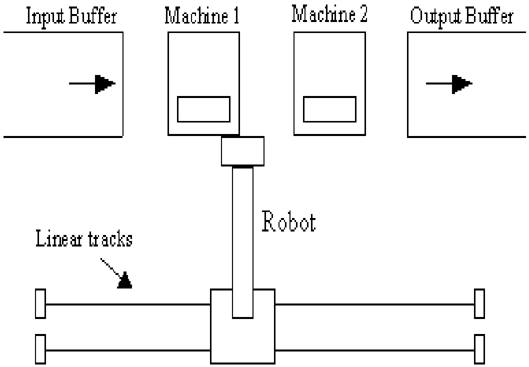
Industrial robots

Video: `http://www.youtube.com/watch?v=xJVhe2QXDUA&feature=player_embedded`

Circular robotic cell



Linear robotic cell



Robotic cell flowshops

- m machines in line (or on a circle), without buffer space: $M_1, M_2, \ldots, M_m$
- loading station M_0 and unloading station M_{m+1}
- set of parts to be produced by the line
- a unique robot loads and unloads the parts

Robotic cell scheduling

Determine

- a sequence of parts,
- a robot activity sequence,
- a production schedule (start/end times),
- so as to minimize cycle time (maximize throughput).

Additional details

Here:

- cycle time = average time elapsed between successive outputs
- throughput = average number of parts per time unit

Number of parts:

- finite ($\longrightarrow$ minimize makespan)
- infinite: a given set of parts (Minimal Part Set, MPS) is to be produced repeatedly ($\longrightarrow$ minimize the asymptotic cycle time).

Additional details

Duration of operations:

- processing intervals : $[L_{kj}, U_{kj}]$
- classical (free) case : $U_{kj} = \infty$
- no-wait : $L_{kj} = U_{kj}$

Robot travel time: d_{ij} = travel time from M_i to M_j.

- symmetry: $d_{ij} = d_{ji}$
- triangular inequalities: $d_{ij} \leq d_{i\ell} + d_{\ell j}$
- additivity: $d_{ij} = d_{i\ell} + d_{\ell j}$

Many other variants: multiple robots, finite buffers, reentrant flowshops, parallel machines, different optimality criteria, etc.

Publications

Many publications over the last 20 years.

- Structural properties of optimal schedules
- Algorithmic complexity
- Approximation algorithms
- etc.

See:

Dawande, M., Geismar, H.N., Sethi, S.P. and Sriskandarajah, C., *Throughput Optimization in Robotic Cells*, Springer, NY, 2007.

Brauner N., Identical part production in cyclic robotic cells: Concepts, overview and open questions, *Discrete Applied Mathematics* 156 (2008) 2480-2492.

Today's assumptions

We mostly concentrate here on:

- repetitive production of identical parts
- cycle time minimization
- free-processing case : $U_{kj} = \infty$
- no intermediate buffers
- additive robot travel times.

Structure of robot move sequences

In traditional scheduling models, we must take care of the part input sequence.

Here, we need to consider robot move sequences.

A robot activity A_i is a sequence of robot moves in which the robot:

1. unloads machine M_i
2. travels from M_i to M_{i+1}
3. loads machine M_{i+1}

Every sequence of robot moves can be decomposed into a sequence of robot activities.

1-Unit cycles

1-Unit cycles

A 1-unit cycle is a sequence of activities which unloads exactly one part in the output buffer and which returns the cell to its initial state.

(In particular, every activity is performed exactly once and the cycle can be repeated indefinitely.)

Simulation: `http://www.ie.bilkent.edu.tr/~robot/`

Theorem

Every permutation of $A_0, \ldots, A_m$ is a 1-unit cycle (and there are exactly $m!$ 1-unit cycles).

1-Unit Cycles

1-unit cycles are attractive for practical and theoretical reasons:

- simplicity of implementation
- the optimal 1-unit cycle can be computed efficiently
- 1-unit cycle conjecture

Optimal 1-Unit Cycles

Complexity

What is the complexity of computing an optimal 1-unit cycle?

Note:

- remember: a 1-unit cycle corresponds to a permutation of the activities $A_0, \ldots, A_m$;
- for fixed m: enumerate all $m!$ 1-unit cycles;
- what about general m?

Pyramidal Cycles

Crama and van de Klundert (1997) proved:

Theorem

There is a pyramidal cycle that minimizes the average cycle time among all 1-unit cycles.

What is a pyramidal cycle?

- a cycle $\pi = (A_0, A_{i_1}, A_{i_2}, \ldots, A_{i_m})$ is *pyramidal* if the indices $i_1, i_2, \ldots, i_m$ are first increasing, then decreasing.

Examples

- $\pi_U = (A_0, A_1, A_2, \ldots, A_{m-1}, A_m)$ (push, uphill);
- $\pi_D = (A_m, A_{m-1}, \ldots, A_2, A_1, A_0)$ (pull, downhill);
- $\pi = (A_0, A_2, A_4, A_5, A_7, A_6, A_3, A_1)$.

Pyramidal Cycles

- there are 2^{m-1} pyramidal permutations on m indices;
- pyramidal permutations have been first investigated in connection with the traveling salesman problem;
- for the TSP, an optimal pyramidal permutation can be computed in polynomial time;
- this result *does not* extend in a straightforward way to robotic cells; but...

Crama and van de Klundert (1997) proved:

Theorem

For a robotic cell with m machines, a pyramidal cycle that minimizes the average cycle time can be computed in $O(m^3)$ time.

Note:

- discussion was restricted to identical parts and additive travel times
- other versions of the problem are NP-hard.

We mentioned earlier: 1-unit cycles are attractive for practical and theoretical reasons:

- simplicity of implementation
- the optimal 1-unit cycle can be computed efficiently
- 1-unit cycle conjecture

1-Unit Cycle Conjecture

There is a 1-unit cycle that minimizes the average cycle time over all possible production cycles.

Reformulation...

A *k*-unit cycle is a sequence of activities which unloads exactly *k* parts in the output buffer and which returns the cell to its initial state.

1-Unit Cycle Conjecture

There is a 1-unit cycle that minimizes the average cycle time over all possible *k*-unit cycles.

1-Unit Cycle Conjecture

Proof of the conjecture:

- 2 machines: Sethi et al. (1992)
- 3 machines: Crama and van de Klundert (1999)

Disproof of the conjecture:

- The conjecture fails in general: when $m > 3$, the best 1-unit cycle may be suboptimal (Brauner and Finke 2001).

Extensions

Since 1-unit cycles do not necessarily minimize cycle time,

Complexity

What is the complexity of computing an optimal (cyclic) robot move sequence?

Problem has been open for 20 years.
More restrictive version:

Complexity of k-unit cycles

Given $k \geq 2$, what is the complexity of computing an optimal k-unit cycle?

Also open.

Approximation properties

Alternatively: how good are 1-unit cycles in general?

Theorem

- *(Crama and van de Klundert 1997) The cycle time of the downhill permutation* $\pi_D = (A_m, A_{m-1}, \dots, A_2, A_1, A_0)$ *is at most twice the optimal cycle time.*
- *(Geismar, Dawande and Sriskandarajah 2005) There is a pyramidal permutation whose cycle time is at most* 1.5 *times the optimal cycle time.*
- *(Geismar, Dawande and Sriskandarajah 2007) There is a pyramidal permutation whose cycle time is at most* $10/7 = 1.43$ *times the optimal cycle time.*
- *Better yet??*

Open questions

Many other variants and results.
Main open questions:

Complexity

What is the complexity of computing an optimal (cyclic) robot move sequence?

Complexity of *k*-unit cycles

Given $k \geq 2$, what is the complexity of computing an optimal k-unit cycle?

Smallest optimal *k*-unit cycles

For an m-machine cell, what is the smallest value of $k = k(m)$ such that k-unit cycles are optimal?

Two-Machine Flowshops with Flexible Operations

Crama and Gulktekin (2010) investigate

- Flowshop/assembly line with two flexible machines.
- Identical parts require three operations.
- Fixed operations can only be processed on a specific machine: operation o_i requires f_i time units on machine M_i.
- One flexible operation with processing time s can be processed by either one of the machines.

Flexible assembly scheduling

Determine the assignment of the flexible operation to one of the machines (for each part) and compute a schedule that maximizes the throughput rate.

Variations

Different problems depending on:

- Buffer capacity between machines: 0, finite or infinite
- Number of parts: finite or infinite

A main difficulty:

A problem instance is defined by 5 numbers: f_1, f_2, s, buffer capacity b and number of parts n.
(Similar to robotic cell scheduling with 2 machines).

$\longrightarrow$ High-multiplicity scheduling problem

Main results

- Infinite buffer case and the no-buffer case: the optimal makespan (or cycle time) and the starting time for each part can be computed in polynomial (constant) time.
- Finite buffer case: polynomial-delay algorithm; the algorithm proceeds sequentially, part after part; it requires $O(l)$ computing time to determine the assignment of the flexible operation for the next part and $O(l)$ time to determine the optimal makespan (or cycle time).
- Never more than 3 parts in the buffer.

Open questions

- Can the complexity be improved in the finite buffer case?
- Extensions to more complex situations: non-identical machines, more machines, etc.

Some references

Brauner N., Identical part production in cyclic robotic cells: Concepts, overview and open questions, *Discrete Applied Mathematics* 156 (2008) 2480-2492.

N. Brauner, G. Finke, and W. Kubiak. Complexity of one-cycle robotic flow-shops. Journal of Scheduling, 6(4):355-371, 2003.

N. Brauner and G. Finke. Cycles and permutations in robotic cells. Mathematical and Computer Modelling, 34(5-6):565-591, 2001.

N. Brauner and G. Finke. Optimal moves of the material handling system in a robotic cell. International Journal of Production Economics, 74(1-3):269-277, 2001.

N. Brauner and G. Finke. On a conjecture about robotic cells: new simplified proof for the three-machine case. Journal of Information Systems and Operational Research - INFOR: Scheduling in Computer and Manufacturing Systems, 37(1):20-36, 1999.

Some references

Y. Crama, Combinatorial optimization models for production scheduling in automated manufacturing systems, European Journal of Operational Research 99 (1997) 136-153.

Y. Crama, A.G. Oerlemans and F.C.R. Spieksma, Production Planning in Automated Manufacturing, Springer, Berlin, 1996, Second, revised and enlarged edition.

Y. Crama, A.W.J. Kolen, A.G. Oerlemans and F.C.R. Spieksma, Minimizing the number of tool switches on a flexible machine, International Journal of Flexible Manufacturing Systems 6 (1994) 33-54.

Y. Crama and J. van de Klundert, Cyclic scheduling of identical parts in a robotic cell, Operations Research 45 (1997) 952-965.

Y. Crama and J. van de Klundert, Robotic flowshop scheduling is strongly NP-complete, in: Ten Years LNMB, W.K. Klein Haneveld, O.J. Vrieze and L.C.M. Kallenberg (eds.), CWI Tract 122, Amsterdam, The Netherlands, 1997, pp. 277-286

Some references

Y. Crama and J. van de Klundert, Cyclic scheduling in 3-machine robotic flow shops, Journal of Scheduling 2 (1999) 35-54.

Y. Crama and J. van de Klundert, Worst-case performance of approximation algorithms for tool management problems, Naval Research Logistics 46 (1999) 445-462.

Y. Crama, L.S. Moonen, F.C.R. Spieksma and E. Talloen, The tool switching problem revisited, European Journal of Operational Research 182 (2007) 952-957.

Y. Crama, V. Kats, J. van de Klundert and E. Levner, “Cyclic scheduling in robotic flowshops”, Annals of Operations Research, v96, pp. 97-124, 2000.

Y. Crama and H. Gultekin, Throughput optimization in flowshops with flexible operations, to appear in Journal of Scheduling.

Some references

Dawande, M., Pinedo, M., and Sriskandarajah, C. 2009. Multiple Part-Type Production in Robotic Cells: Equivalence of Two Real-World Models. Manufacturing and Service Operations Management 11, 2 (Apr. 2009), 210-228.

M. Dawande , H.N. Geismar , S.P. Sethi , Chelliah Sriskandarajah, Sequencing and Scheduling in Robotic Cells: Recent Developments, Journal of Scheduling, v.8 n.5, p.387-426, October 2005

Dawande, M., Geismar, H. N., Sethi, S. P. and Sriskandarajah, C., Throughput Optimization in Robotic Cells, Springer, New York, 2007.

Dawande, M., M.A. Chan, N. Geismar and C. Sriskandarajah, Approximation Algorithms for Optimal *k*-Unit Cycles in Dual-Gripper Robotic Cells, Production and Operations Management Vol. 17, No. 5, 551-563, 2008.

Geismar, H.N., M. Dawande, and C. Sriskandarajah, “Approximation algorithms for *k*-unit cyclic solutions in robotic cells”, European Journal of Operational Research, 162, 291-309 (2005).