Configurable Multi-Product Floorplanning

Qiang Ma, Martin D.F. Wong and Kai-Yuan Chao

Dept of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign

Intel Corporation
Hillsboro, OR

Jan 21, 2010

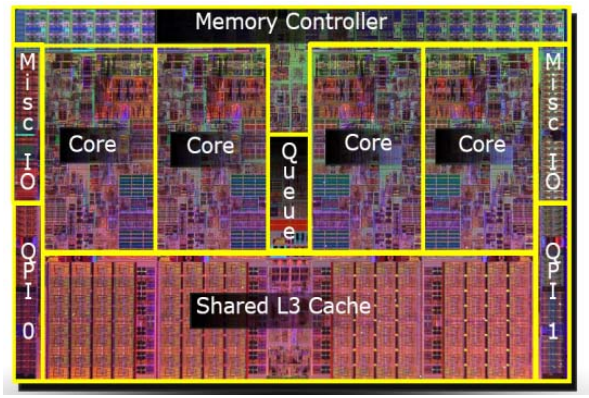
Outline

- Introduction
- Problem Formulation
- Algorithm
- Experimental Results
- Conclusion

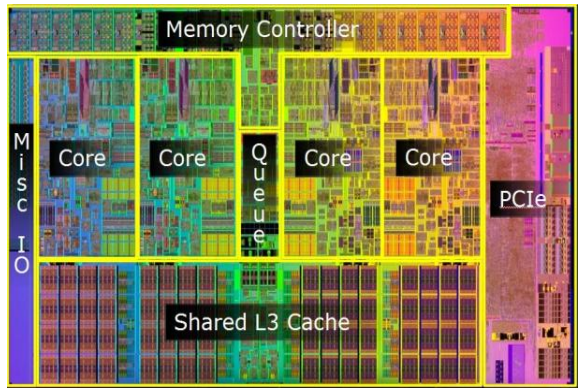
Motivation

- **CPU design**: modify original design to produce related products targeting different market segments (e.g., scientific computing, high-end server, low-end server, desktop, mobile etc.)
- **Intel's Nehalem CPUs**: 45nm, 10 configurations (cores, cache, integrated chipset, IO etc.), at least 21 CPU products planned
- Requires high ECO and design re-convergence efforts unless we plan for all products up front
- Important problem: *configurable multi-product floorplan design*

Intel's Nehalem family



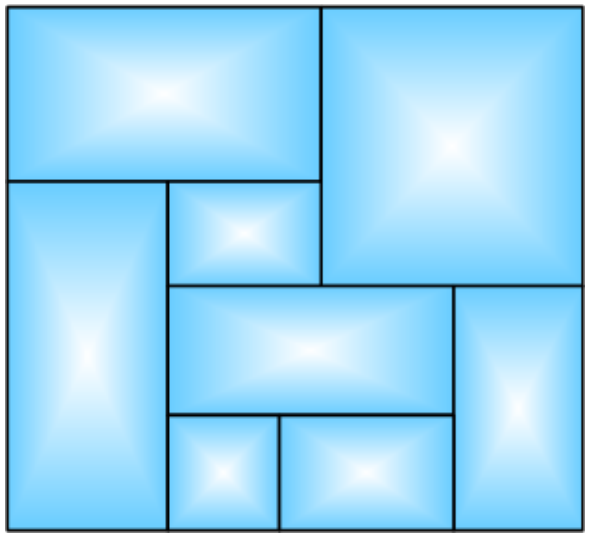
Bloomfield



Lynnfield

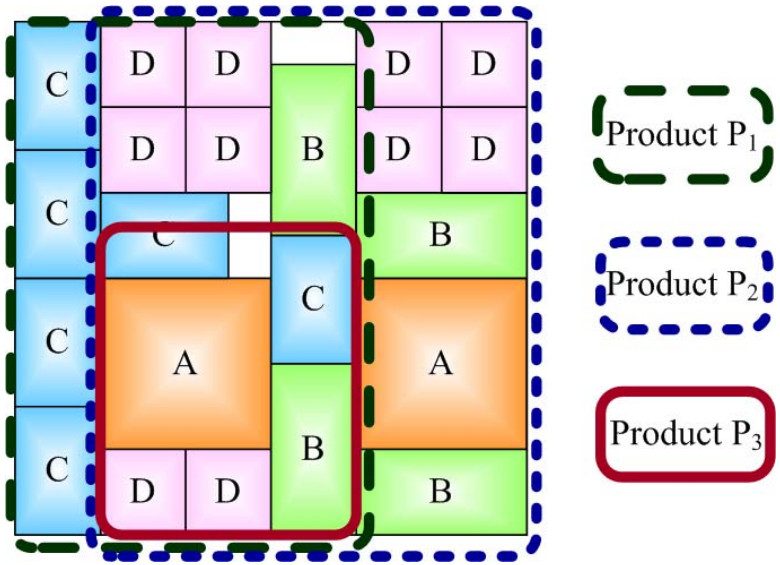
Traditional Floorplan Design

- Traditional floorplanning problem
  - Given a set of rectangles, each with a range of aspect ratio
  - Generate a packing of all the rectangles s.t.
    - the rectangles do not overlap
    - the total area and wire length are minimized



Multi-product Floorplan Design

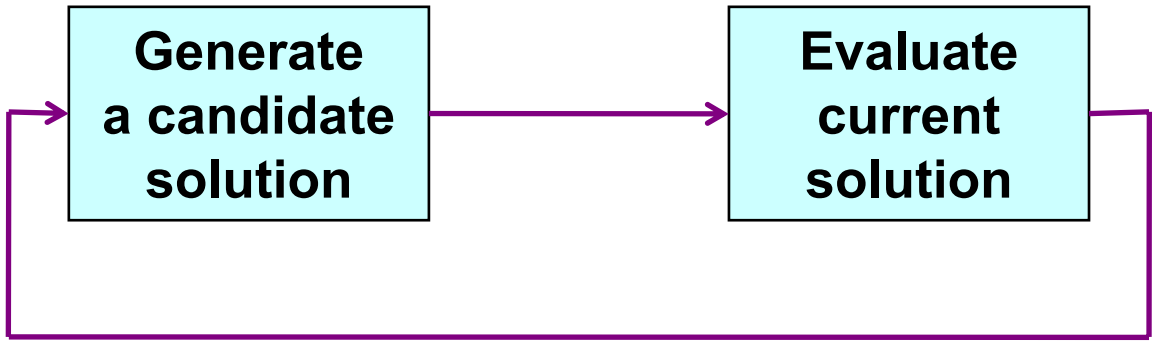
- *Multi-product Floorplanning*
  - Generate a floorplan of enough components (basic blocks) that is configurable for all the products
  - The rectangular region for each product is as small as possible



Requirement of Products:

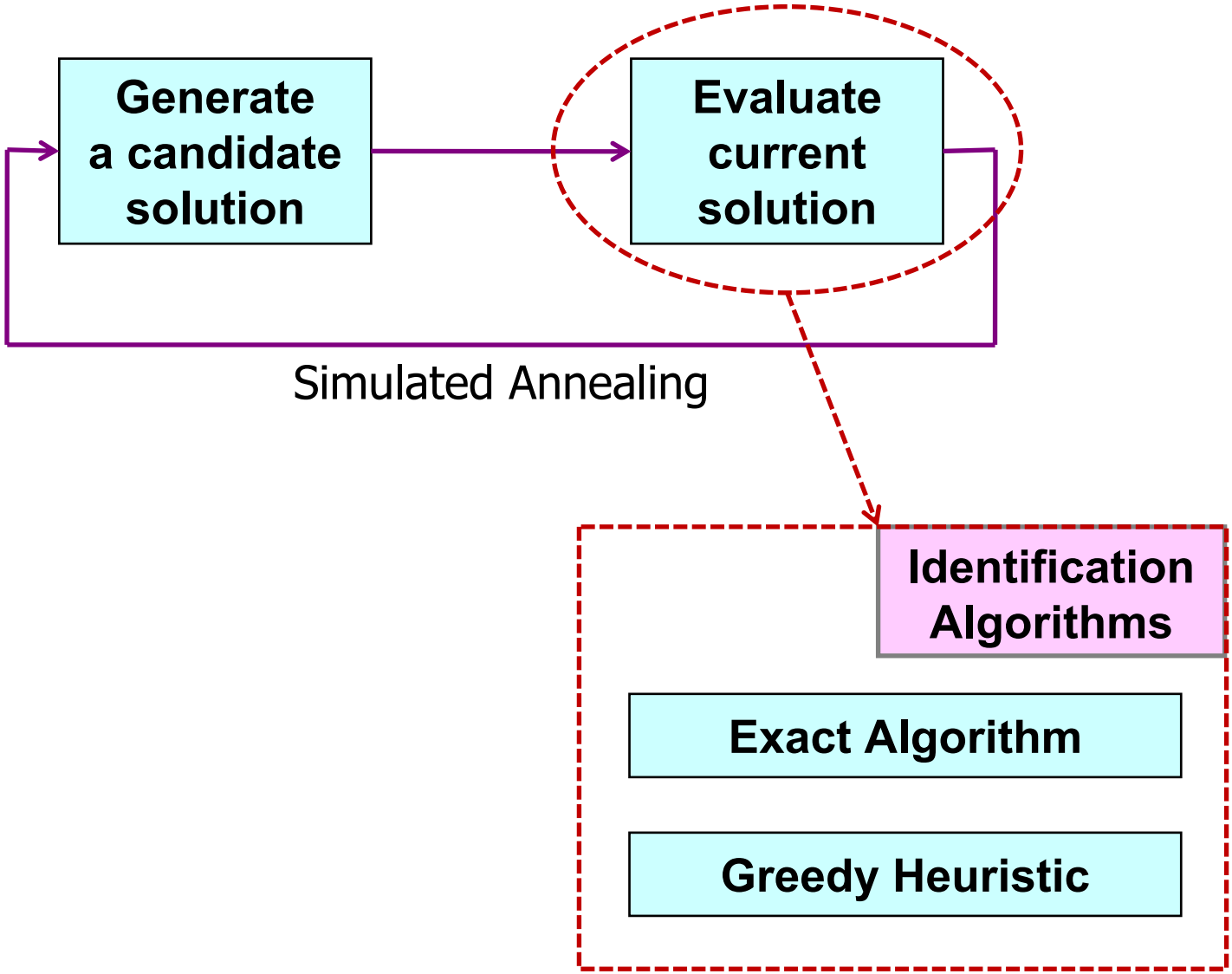
Products	Basic Blocks			
	A	*B*	*C*	*D*
P_1	1	2	6	6
P_2	2	4	2	10
P_3	1	1	1	2

Traditional Floorplan Algorithm



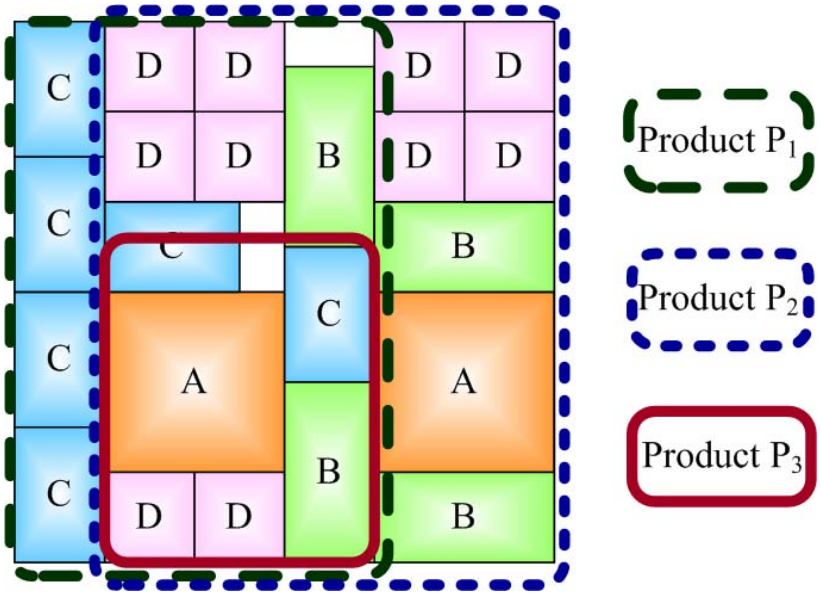
Simulated Annealing

Our Multi-product Floorplan Algorithm Flow



Notations

- Problem inputs
  - *m* types of basic blocks $\{b_1, b_2, \ldots, b_m\}$
  - *q* products $\{P_1, P_2, \ldots, P_q\}$
  - Demand Vector of P_i : $D^i = [D^i_1, D^i_2, \ldots, D^i_m]^T$

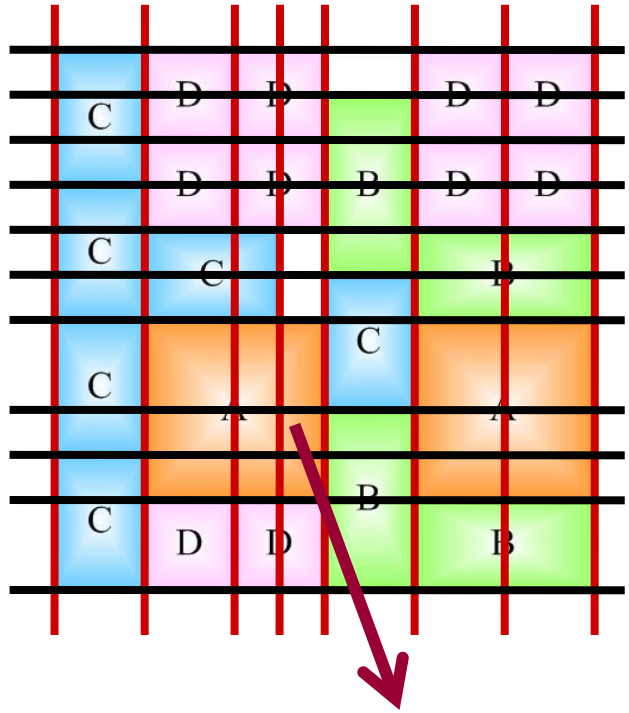


Requirement of Products:

Products	Basic Blocks			
	A	*B*	*C*	*D*
P_1	1	2	6	6
P_2	2	4	2	10
P_3	1	1	1	2

Notations

- Notations on a candidate floorplan
  - ***Cutline*** : a line obtained by extending the interval of a block boundary
  - ***Valid Region*** : A rectangular region *R* on a candidate floorplan formed by four Cutlines *l, r, t, b* (denoted by *R(l,r,t,b)*).
  - ***Capacity Vector*** : $C^R = [C^R_1, C^R_2, \ldots, C^R_m]^T$ of a Valid Region *R* , where C^R_j denotes # of basic block b_j lying within *R.*
  - ***Feasible Region*** : *R is Feasible for product* P_i *if and only if* $C^R \geq D^i$.
  - ***MFR*** : Minimum Feasible Region

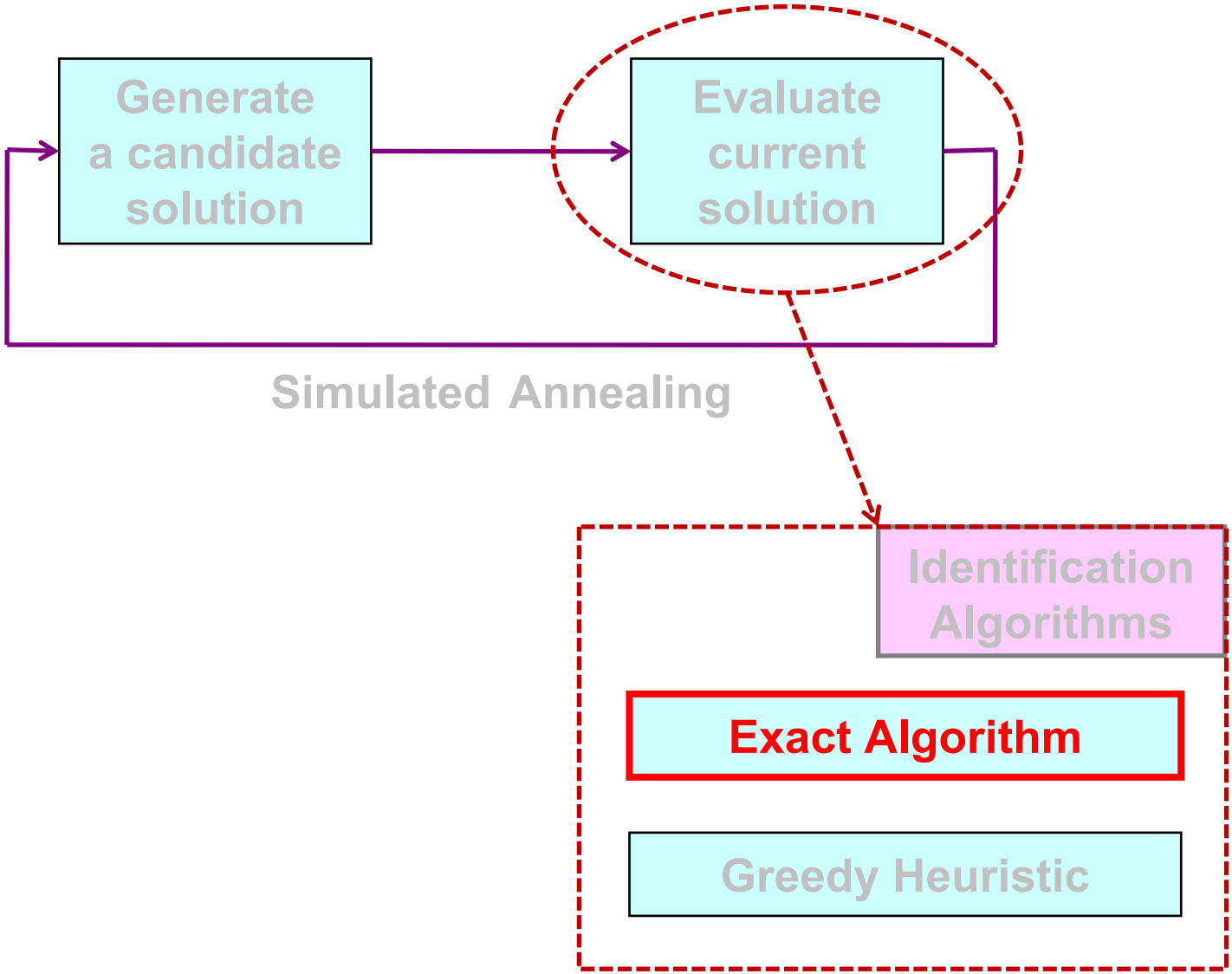


$C^R = [1,1,0,2]^T$

Problem Formulation

- ***Multi-product Floorplanning (MPF)***
  - Inputs:
    - A set of m basic blocks $\{b_1, b_2, \ldots, b_m\}$
    - A set of *q* products $\{P_1, P_2, \ldots, P_q\}$
    - Each P_i is associated with a Demand Vector D_i
  - Objective:
    - *Generate a floorplan of sufficient* basic blocks to accommodate all *q products*
    - *The total* area of the MFR for each product on the resultant floorplan *is minimized*

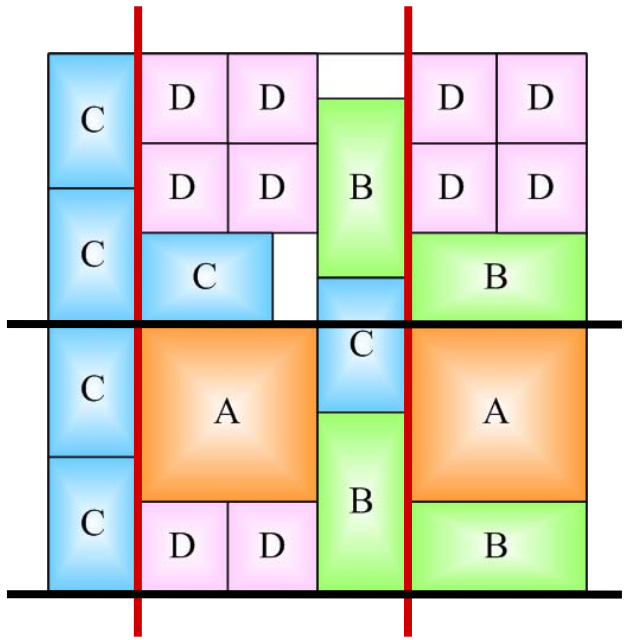
Our Multi-product Floorplan Algorithm Flow



Product Identification

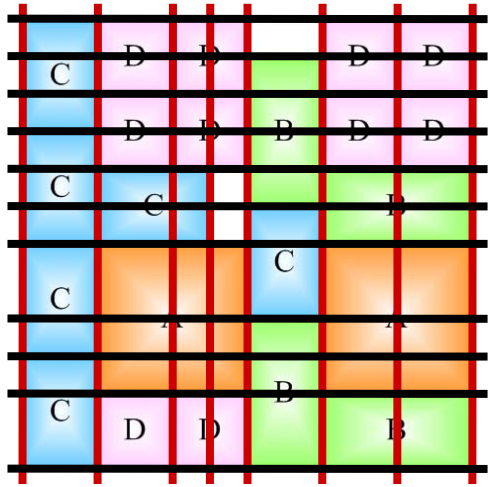
- Naïve identification of the Minimum Feasible Region (MFR) for a product is expensive
  - O(n) horizontal cutlines
  - O(n) vertical cutlines
  - $O(n^4)$ candidate rectangular regions to check
  - O(n) operations to check if a region is feasible or not
  - Time complexity is $O(n^5)$

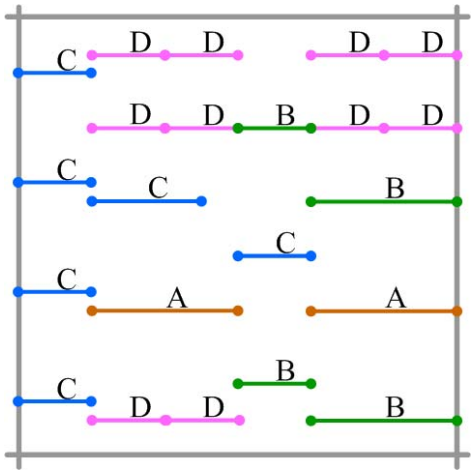
Can we do better?



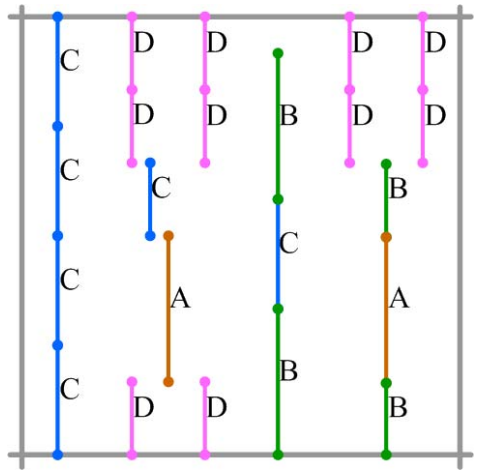
Preprocessing

- Sorting all the cutlines
  - *Left-Cutlines [n]*
  - *Right-Cutlines [n]*
  - *Top-Cutlines [n]*
  - *Bottom-Cutlines [n]*
- Sorting block intervals
  - Horizontal interval array *H[2n]*
  - Vertical interval array *V[2n]*





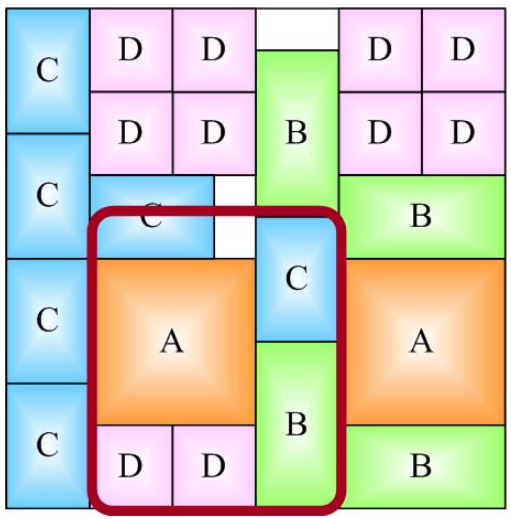
Horizontal intervals



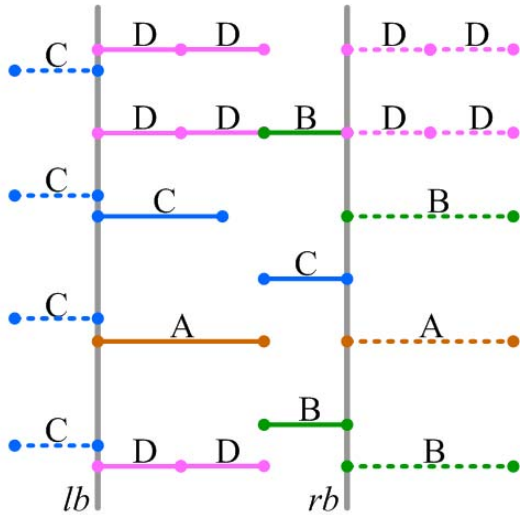
Vertical intervals

Exact Identification Algorithm

- Horizontal Scan (H-Scan)
  - Determine a Horizontal Feasible Region (HFR) for a product
  - Horizontal interval array H is scanned
  - Fix left boundary, shift right boundary until the current region contains sufficient basic blocks



(a) $D^3=[1,1,1,2]^T$



(b) $C^R=[1,2,2,6]^T$

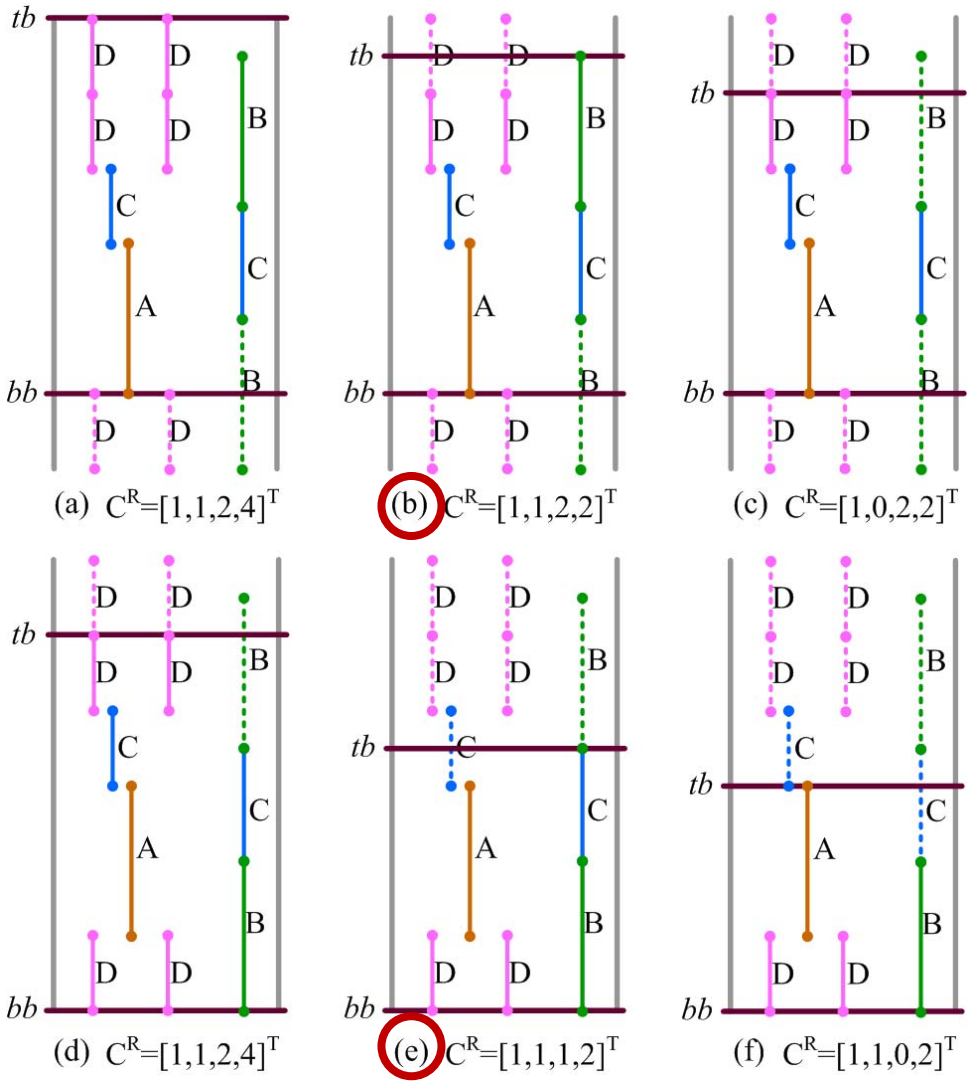
Exact Identification Algorithm

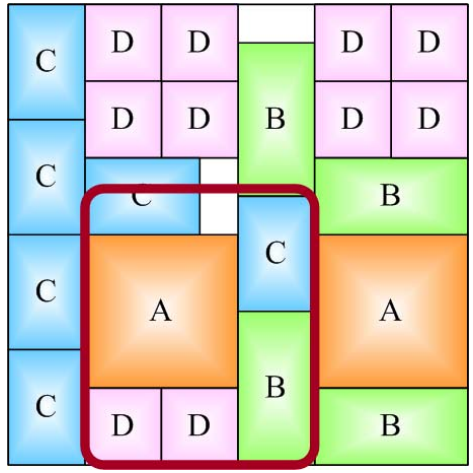
- Vertical Scan (V-Scan)
  - Given a *HFR(lb,rb)*, find *HFR*(lb,rb)* with minimum height
  - Find positions for *tb* and *bb* s.t. (*bb-tb)* is minimum
  - At the beginning, *tb* = *Top-Cutlines[1], bb= Bottom-Cutlines[1]*
  - Repeat
    - Shift *bb* down until *R(lb,rb,tb,bb)* is feasible
    - Shift *tb* down until *R(lb,rb,tb,bb)* is infeasible
    - Record the current height and update the best
  - Until *bb* is the last Bottom Cutline

- ***Lemma 1****: Given a HFR(lb,rb), the procedure V-Scan correctly returns the HFR*(lb,rb) in O(n) time.*

Exact Identification Algorithm

An example for V-Scan procedure



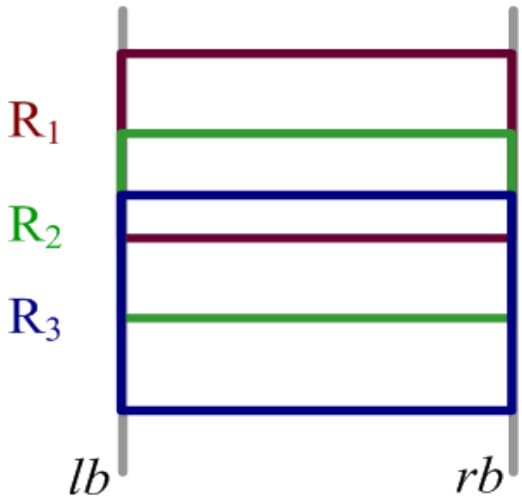


Demand vector:

$D^3=[1,1,1,2]^T$

Exact Identification Algorithm

- Each iteration of *V-Scan()* identifies a minimal HFR
- *V-Scan()* enumerates all minimal HFRs, and the one with minimum height is the HFR*
- Observe that the minimal HFRs are ***"increasing"***
  - i.e., tb(R_1) is above tb(R_2) iff bb(R_1) is above bb(R_2)
- So *V-Scan()* can correctly find the HFR*

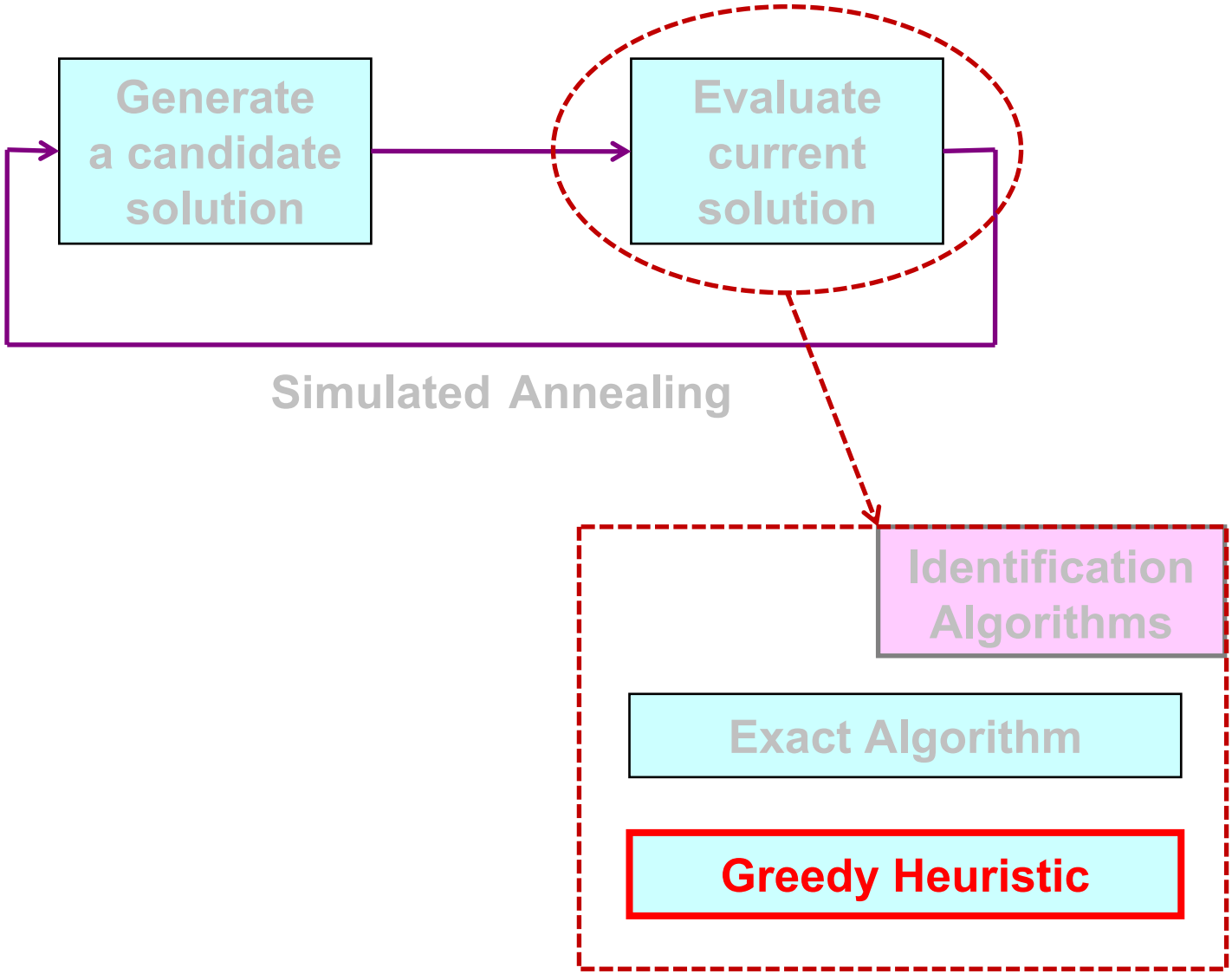


Exact Identification Algorithm

- The Algorithm
  - *H-Scan()* enumerates all the *HFR*s in an ordered fashion
  - *V-Scan()* captures the *HFR** of each *HFR*
  - Optimality: *MFR* must be the *HFR** of some *HFR,* and that all the $O(n^2)$ *HFR*s are enumerated

- ***Theorem 1****:* The Exact Identification Algorithm correctly returns the MFR for a product in $O(n^3)$ time

Our Multi-product Floorplan Algorithm Flow

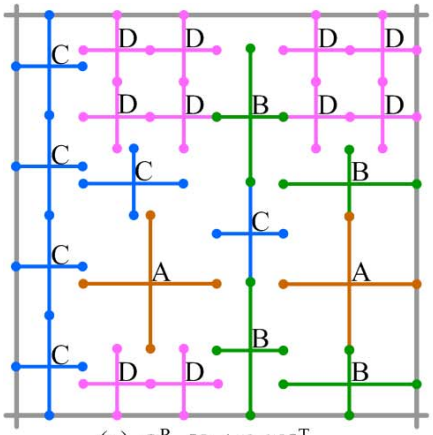


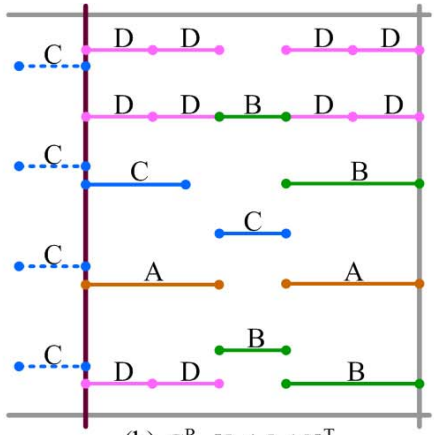
Greedy Identification Heuristic

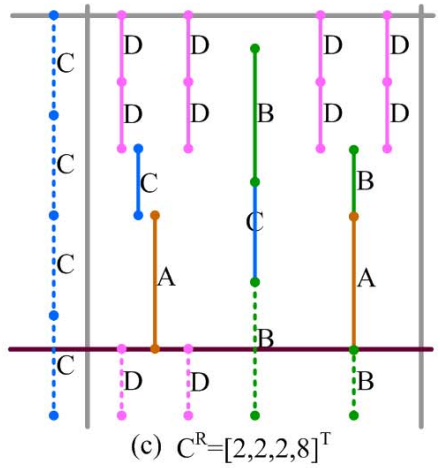
- Given a candidate floorplan and a product P_i's demand vector D^i, the greedy heuristic behaves as follows
  - *The feasible region R(lb,rb,tb,bb) starts with the whole floorplan*
  - *Greedily shrink the four boundaries*
    - Pick one boundary *b* and shift it one step to the center
    - Its shifting reduce *R* by the most
      - Break the tie by picking the least recently used boundary
    - The feasibility of *R* is maintained
    - Scan Horizontal interval array *H* and vertical interval array *V*
    - Stop when *R* can not be shrunk any more
    - The resultant region *R* is a Minimal Feasible Region
- ***Theorem 2***: The greedy heuristic returns a *Minimal Feasible Region* for a product in *O(n) time.*

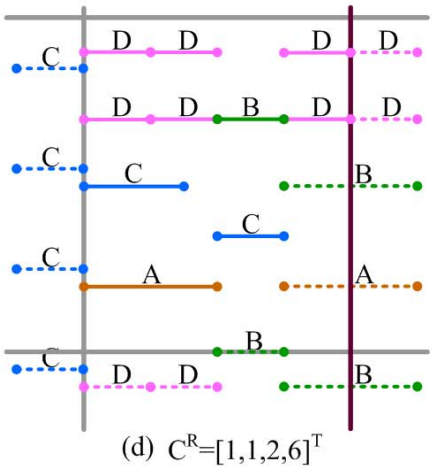
Greedy Identification Heuristic

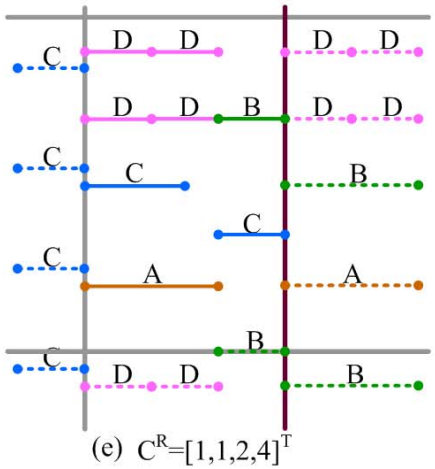
An example for Greedy Heuristic

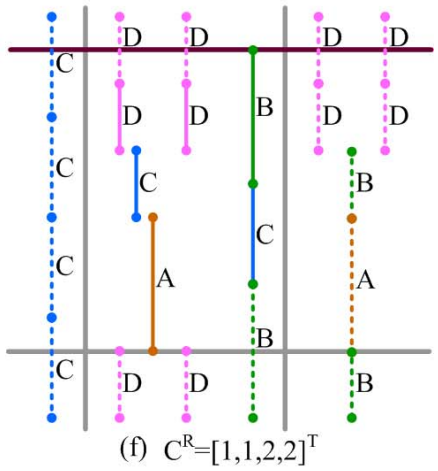












Demand vector:

$D^3=[1,1,1,2]^T$

Experimental Results

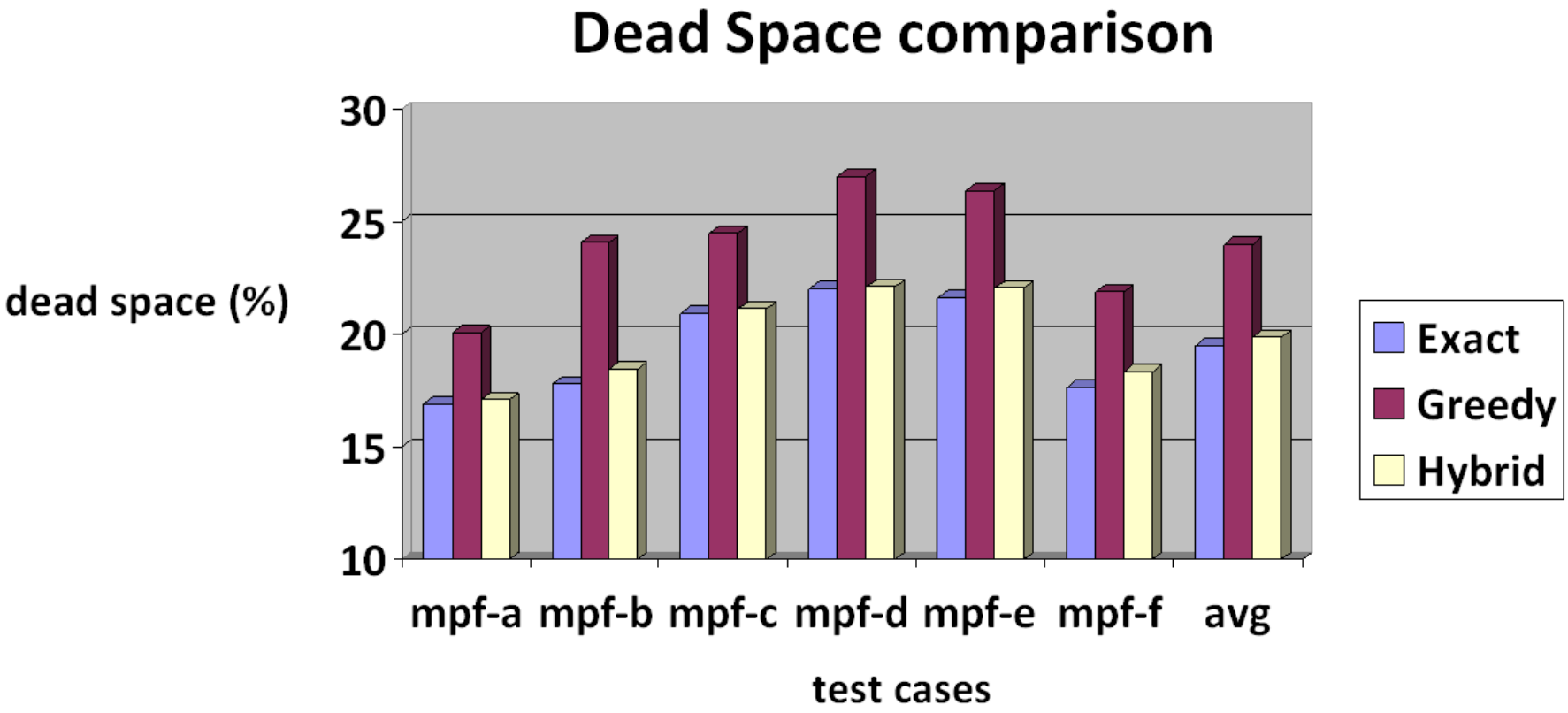
- Exact
  - Use the Exact Identification Algorithm
- Greedy
  - Use the Greedy Identification Heuristic
- Hybrid Method
  - Combine the two algorithms
  - Provide good tradeoff between solution quality and run time
  - Applying the greedy heuristic in the first stage
  - Switching to the exact algorithm in the second stage, when the acceptance ratio drops below a certain value

Experimental Results

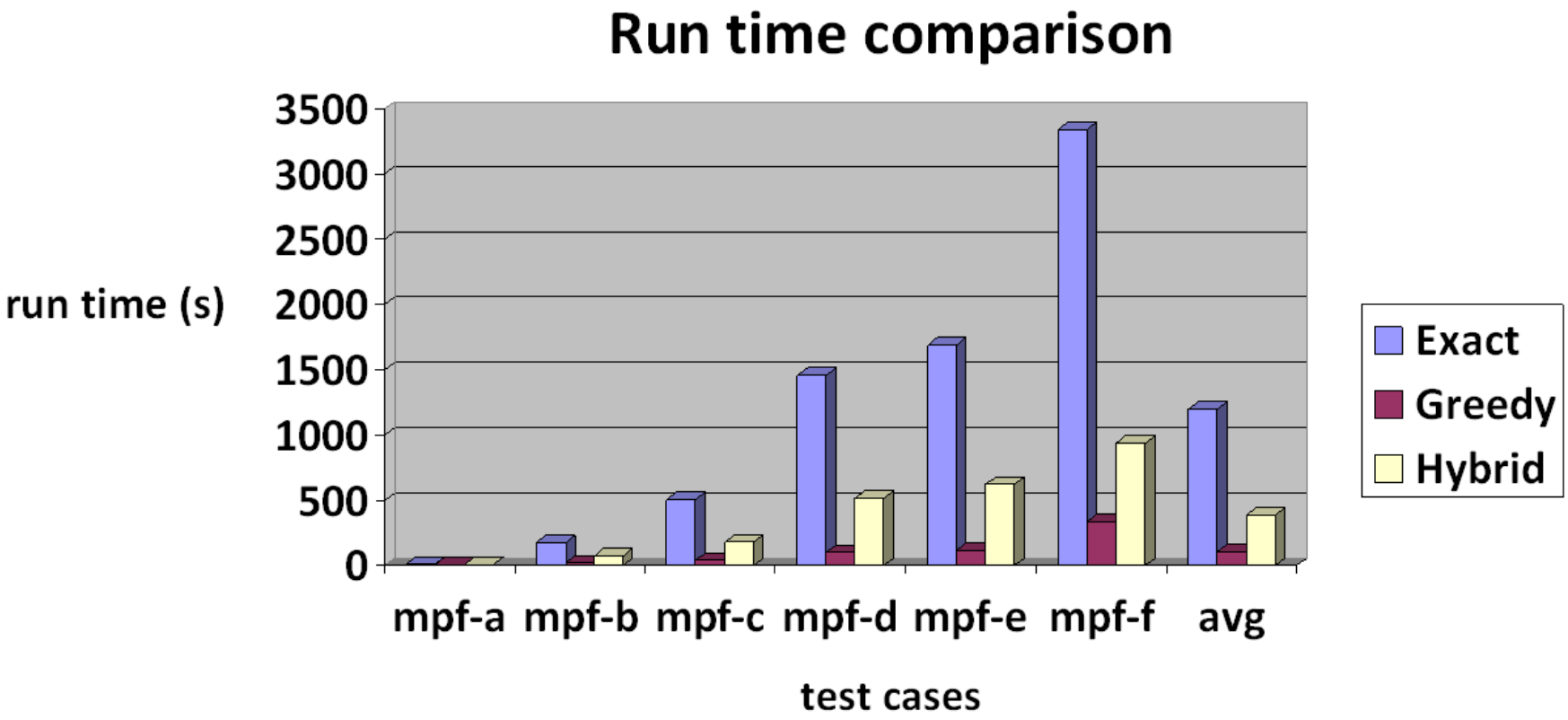
- The three method are tested and compared on a set of test cases derived from industrial data

Test Cases	# of Products	# of Bacis Blocks
mpf-a	6	32
mpf-b	8	75
mpf-c	10	90
mpf-d	12	140
mpf-e	15	150
mpf-f	20	190

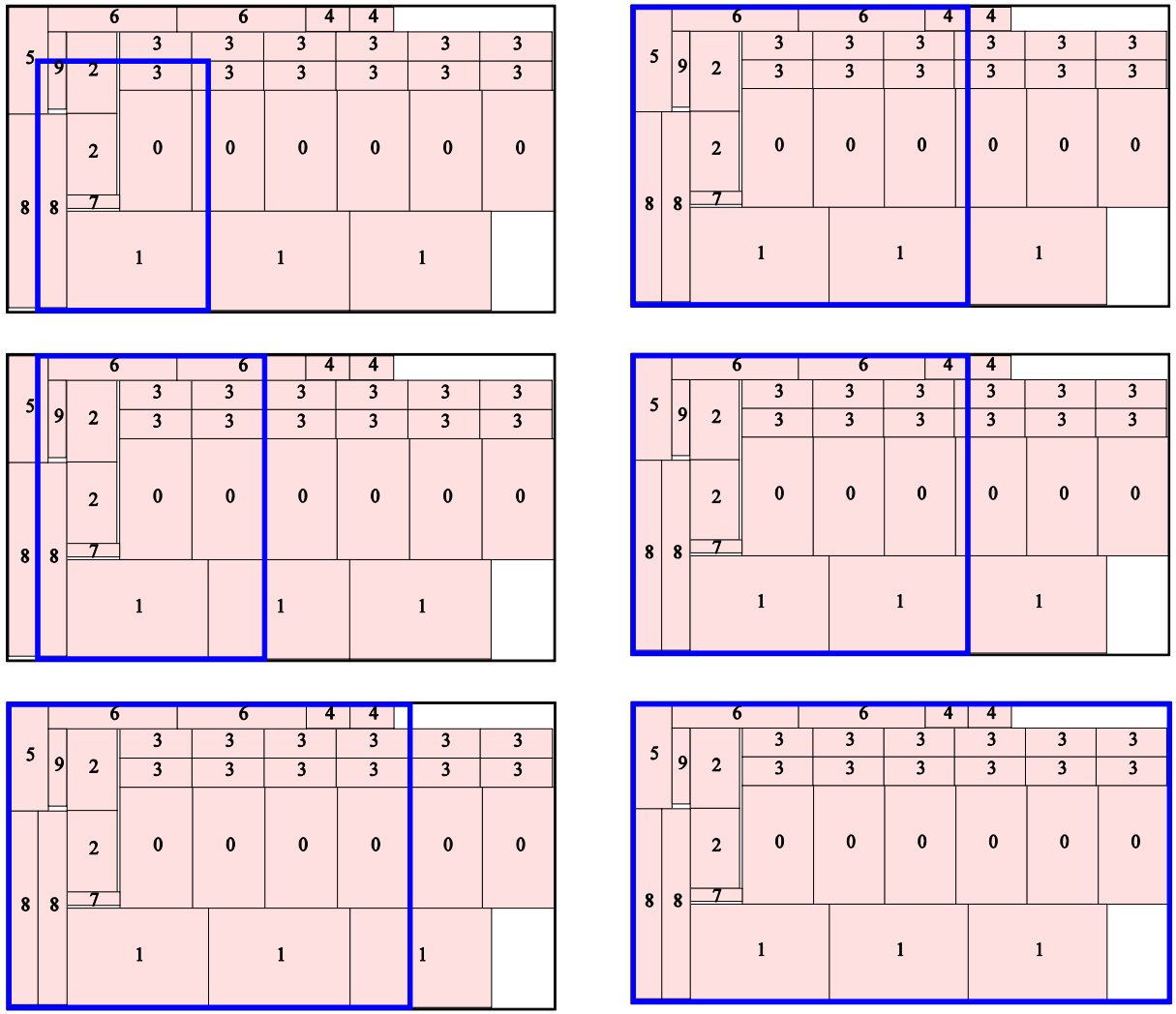
Experimental Results



Experimental Results



Experimental Results



Test case mpf-a with 6 products

Conclusion

- We introduced the Multi-product Floorplanning problem
  - Simultaneously design for a family of related products
  - Significantly reduce overall design time and cost
- First work in literature addressing this problem
- We proposed a SA based approach
  - An $O(n^3)$ exact product identification algorithm
  - An $O(n)$ greedy product identification heuristic
  - These two algorithms can be combined to provide a good tradeoff between solution quality and run time
- The effectiveness of our approach is validated by promising results on a set of test cases