The Creation of a Cybernetic (Multi-Strategic Learning) Problem-Solver: Automatically Designed Algorithms for Logic Synthesis and Minimization

by

Karen M. Dill

Marek A. Perkowski

GRM Minimization – Origins of Problem

Two Approaches:

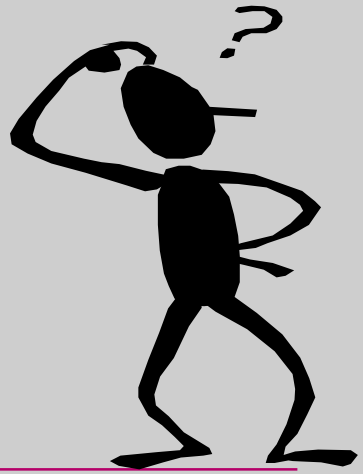
- **GRMIN/2: Debnath/Sasao**
  - – **Rule-Based: Human Design**
    - » **Extensive Development Time**
    - » **Smart, Often Exact Solutions**
    - » **Large/Small Benchmarks**
    - » **Fast Computation**
- **GA-GRM/MO: Dill/Perkowski**
  - – **Evolutionary Based: Auto Design**
    - » **Minimal Development Time**
    - » **Good, Competitive Solutions**
    - » **Small Benchmarks**
    - » **Slow Computation**

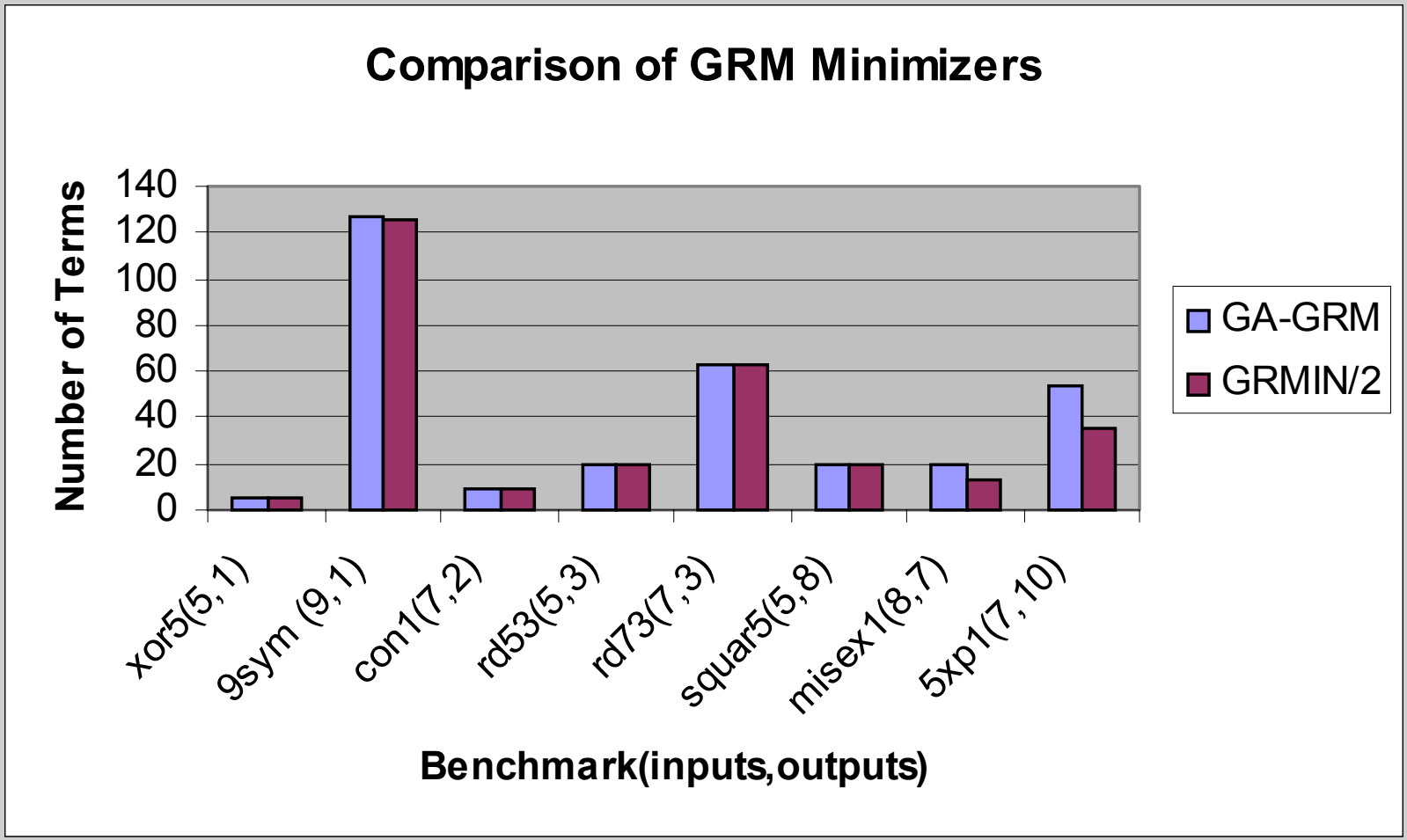
Run-Time
Dev-Time
Results

Which is cheaper?

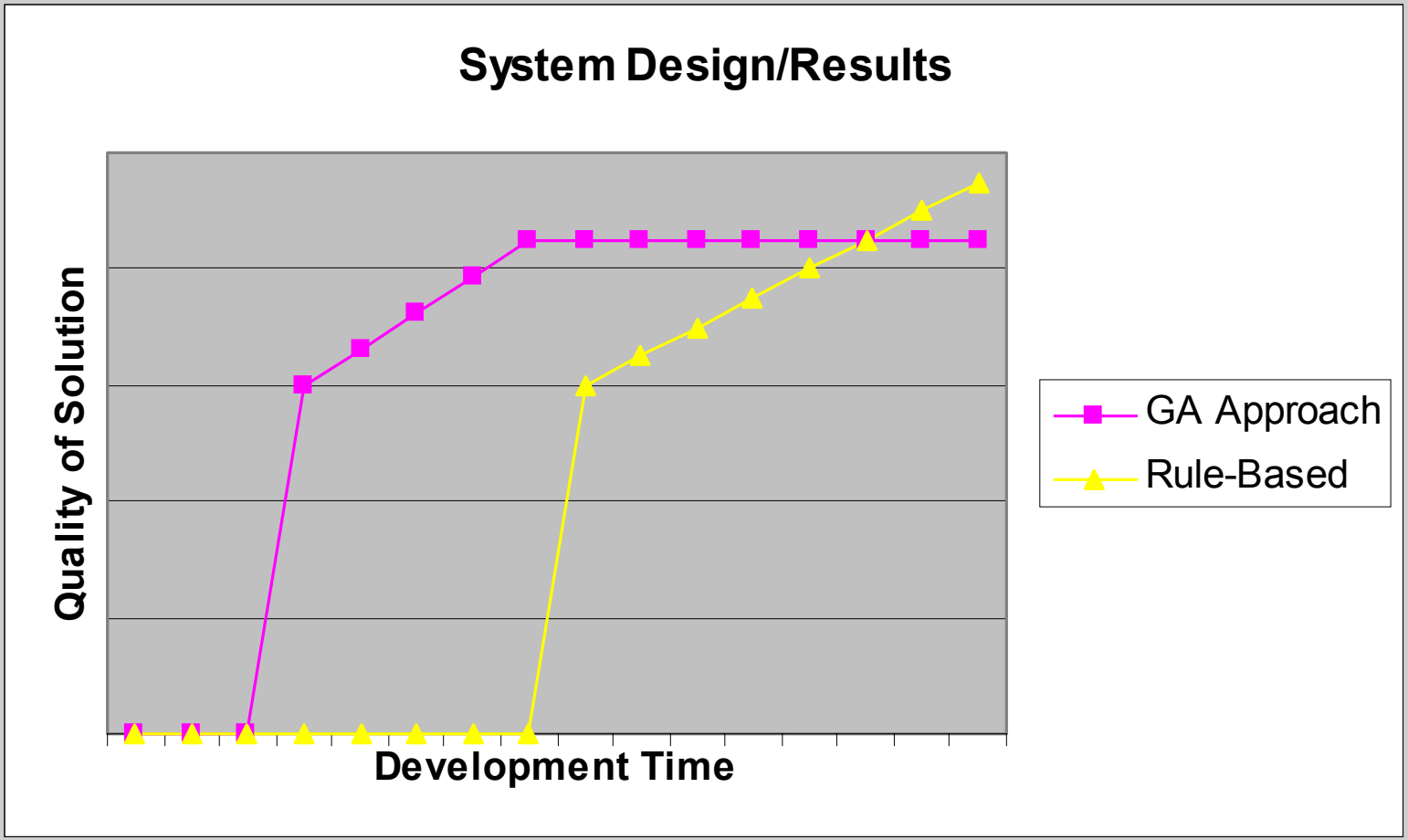
More practical?

SW Skills
No. Apps.
End-User




Comparison of GRM Minimizers
Number of Terms
140
120
100
80
60
40
20
0
xor5(5,1)
9sym (9,1)
con1(7,2)
rd53(5,3)
rd73(7,3)
squar5(5,8)
misex1(8,7)
5xp1(7,10)
Benchmark(inputs,outputs)
GA-GRM
GRMIN/2

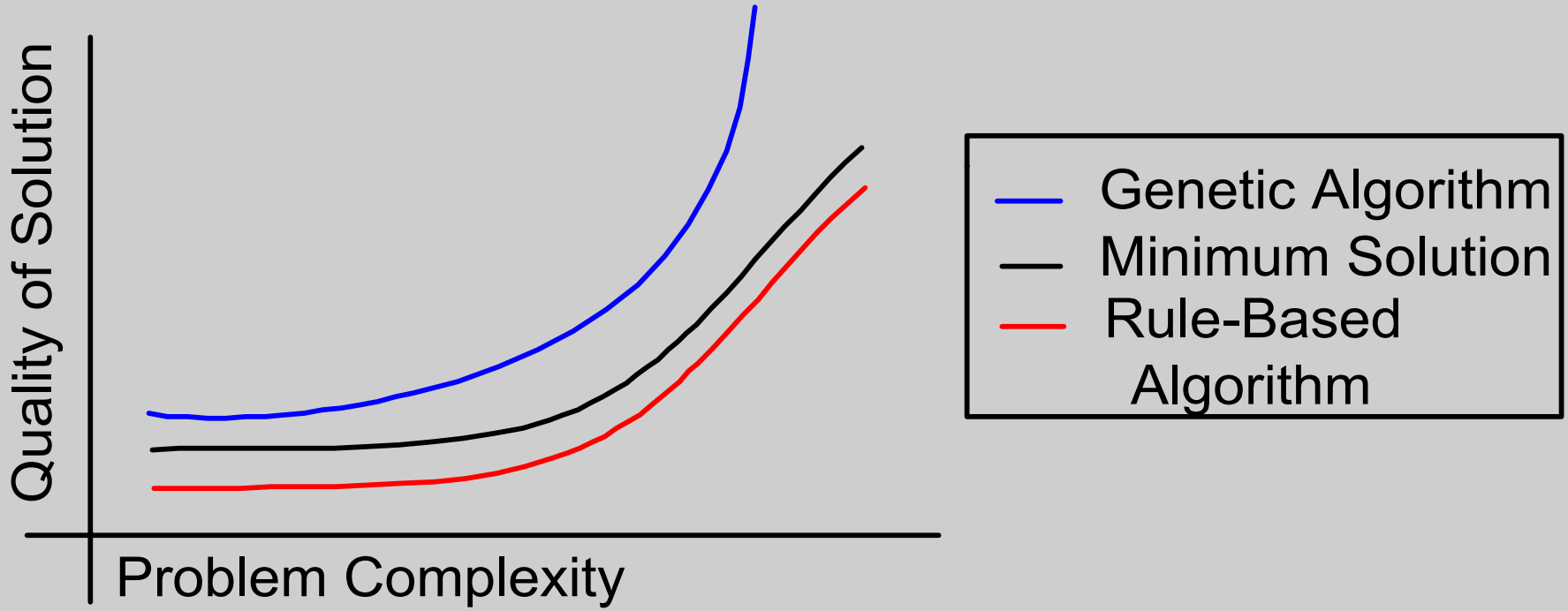
Our Experience with Evolutionary Methods… an Intuitive Portrayal



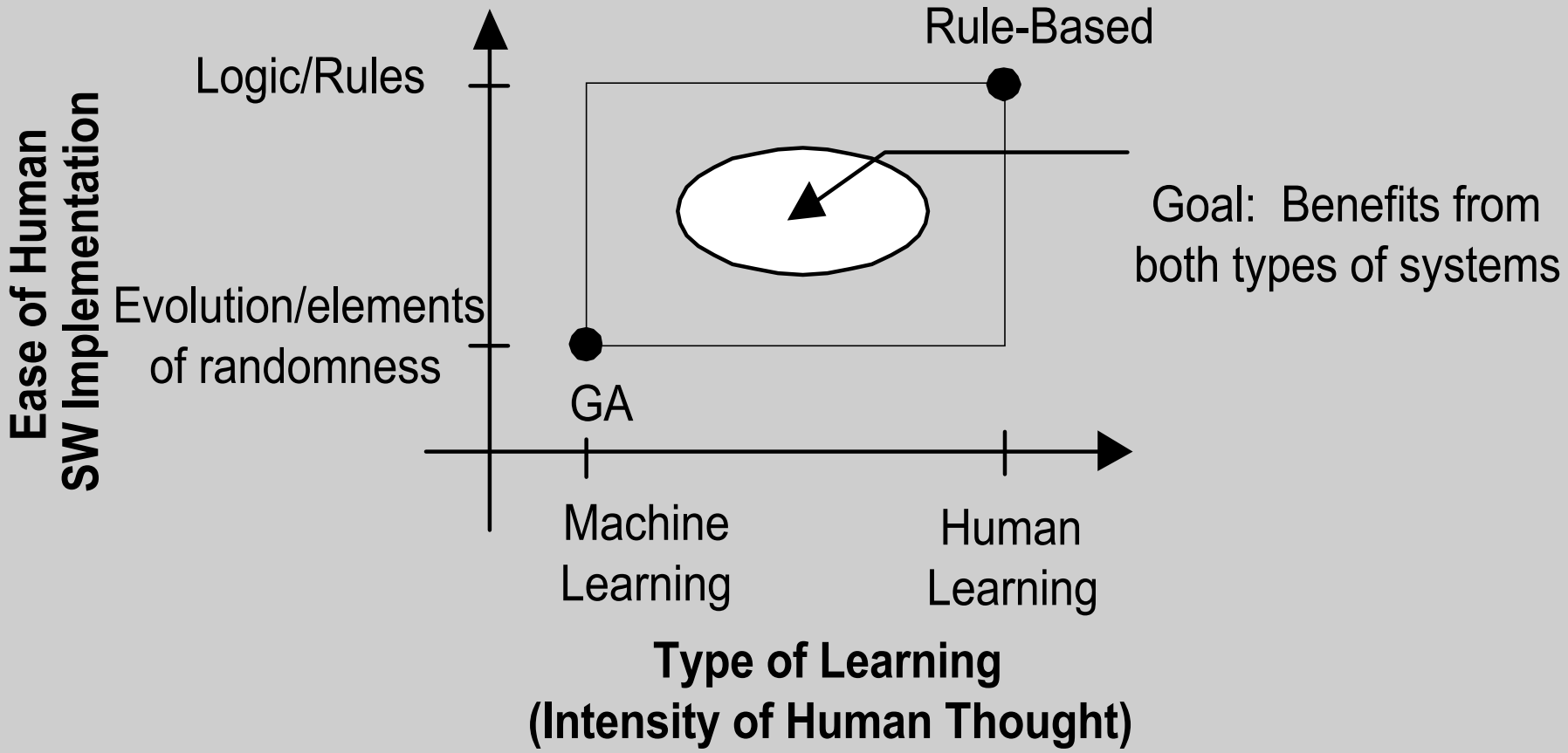
Our Experience with Evolutionary Methods…

An Intuitive Portrayal

Difference between Minimal and Algorithmic Solutions



Cybernetic Problem Solver (CPS)



OVERVIEW – Cybernetic Problem Solver (CPS)

Goal:

- **Automatically designs application specific algorithms, for classes of problems**
  - **Binary/MVL Synthesis and Minimization Problems**
  - **Robotics**

Why?

- **Human-designed Expert Systems**
  - **Work Well; Long Development Time**
  - **Limited in application**
- **Functional Decomposition**
  - **Good Solutions**
  - **Not easily tunable to all technologies**

Difficulties with Solution Approach Mechanisms

- Evolutionary Algorithms (GA/GP)
  - Time/Size Limitations
  - Quasi-Optimal Solutions (experience)
  - No Guarantees
  - No Explanation/Description
    - > Design Methodology, or
    - > Transferable Rules of Generalization
  - No Optimized Approach for Problem Class (too general)
- Traditional Exhaustive Search Mechanisms
  - Breadth First, Depth First, etc.
  - Guarantee an Optimal Solution
  - Prohibitively Time/Resource Consuming

Difficulties with Solution Approach Mechanisms

Unsatisfactory Search Strategies for a General Problem Solving Technique:

- **Complete (entire state-space) Search**
- **Incomplete (evolutionary and rule-based) Search**

⇒CPS Algorithm incorporates:

- **Pure and Heuristic Search Strategies**
- **Problem Solving/Learning Paradigms**
- **Builds on strengths and thus, reduces weakness of individual methods**

OVERVIEW – Cybernetic Problem Solver (CPS)

- **How?**

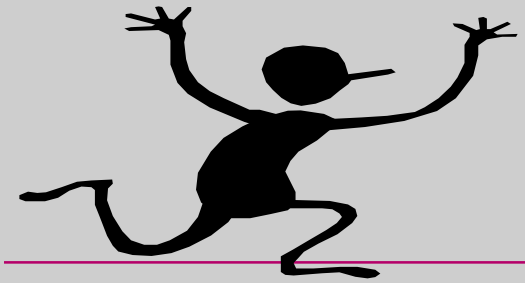
  Human Expertise + Search Mechanisms ⇒
  Efficient Problem-Solving Methods

  - (Don't waste time applying search mechanisms to re-invent known methods/solutions)

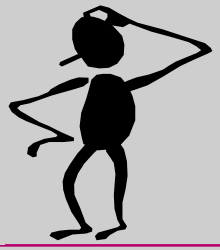
- **What?** **Cybernetic Problem Solver**

- Multi-strategic
- Intelligent
- Superset "Toolbox"
- Gaming Concepts: compete & cooperate

→ Various Learning Methodologies Combined ←

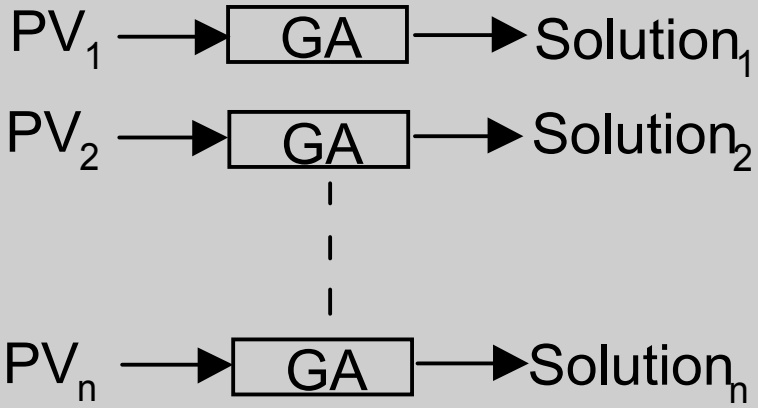


I have an idea…

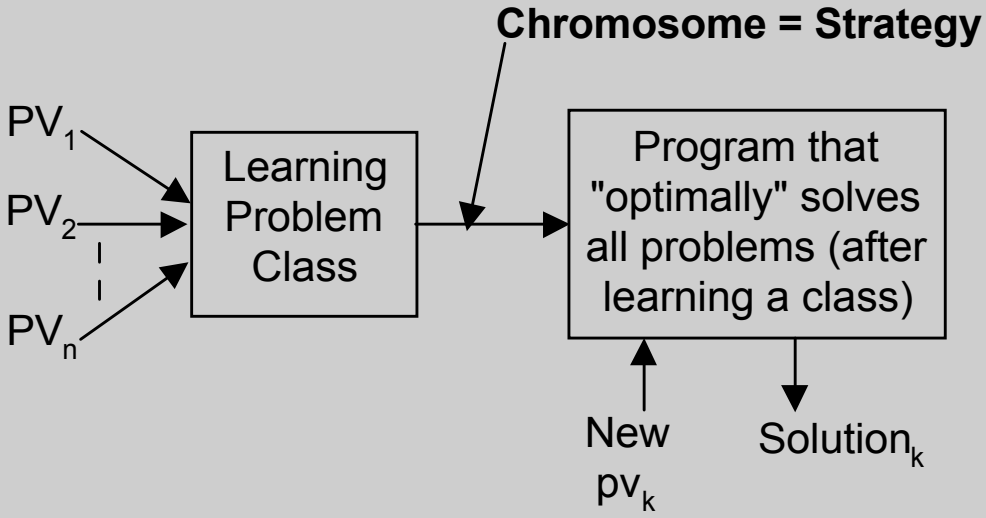


Learning Differences: GA vs. CPS

GA



CPS



Learning Differences:

- GA learns specific problems
- CPS learns classes of problems

Set Covering Problem:
CPS Example Application & Preliminary Results:

Set-Covering Problem: Column minimization problem, often seen in logic design (e.g., PLA minimization, test minimization, multi-level design, robotic CIM minimization, security, image processing, layout, etc.)

Given: A Covering Table of a number of rows and columns, each associated with a cost.

-Set R = $\{r_1, r_2, \ldots, r_k\}$, where each r_i is a row in the table

-Set C = $\{c_1, c_2, \ldots, c_n\}$, where each c_j is a column in the table

-Row r_i covers column c_j, with the relation COV(r_i,c_j)

Goal: Find a set of rows that cover all the columns at a minimal cost.

Set-Covering Problem:

CPS Example Application & Preliminary Results

	c_1	c_2	c_3	c_4	c_5	c_6
r_1		x	x		x	
r_2	x		x			
r_3		x	x			
r_4				x	x	x
r_5		x		x		

Problem: Find the smallest subset of rows that will cover all columns.

Solution:

***The solution must have:**

r_2 to cover c_1
and
r_4 to cover c_6.

Decision Function:

For covering

c_1 use	$\mathbf{r_2}$
c_2	$r_1 + r_3 + r_5$
c_3	$r_1 + r_2 + r_3$
c_4	$r_4 + r_5$
c_5	$r_1 + r_4$
c_6	$\mathbf{r_4}$

$DF = \mathbf{r_2} * (r_1 + r_3 + r_5) * (r_1 + r_2 + r_3) * (r_4 + r_5) * (r_1 + r_4) * \mathbf{r_4} = 1$

Smallest set to satisfy condition: Exact Minimal Solutions

Sol = $\{r_2, r_4, r_1\}, \{r_2, r_4, r_3\}, \{r_2, r_4, r_5\}$

Set-Covering Problem:
CPS Example Application & Preliminary Results

CPS Preliminary Results (over set of tables):

- Equivalence/Dominance/Indispensable conditions <u>each</u> reduces search space ~2-3x

- Experimental Results compared with the initial random vector

> Reduces generated space ~50-200 times

> Increases computation time in the learning phase by 20-30%

> Vector of coefficients (c's)* obtained decreased generated solution space size 15-20% from initial vector

- *(c's in search methods' chromosome, $c_1w_1 + c_2w_2 + \ldots$)

Productivity Problem:
CPS Example Application & Preliminary Results

· **Productivity Problem**, CAD Applications, Encoding

Given: n machines and n workers, the productivity of the worker i on machine j is denoted w_{ij}.

Goal: The assignment matrix $W_{nxn} = [W_{ij}]$ is given from which n elements must be selected in such a way that when any two of them are taken from different rows and columns, that the sum of the elements is maximum.

Example of Productivity Problem

Toy Widget Workshop

Output/hr.	Job A	Job B	Job C
Worker	**Woozles**	**Wizzles**	**Weazles**
Susan	4	2	**12**
Paul	**7**	6	4
Mark	3	**10**	5

Best Assignments for Maximum Productivity
of all toy product lines:

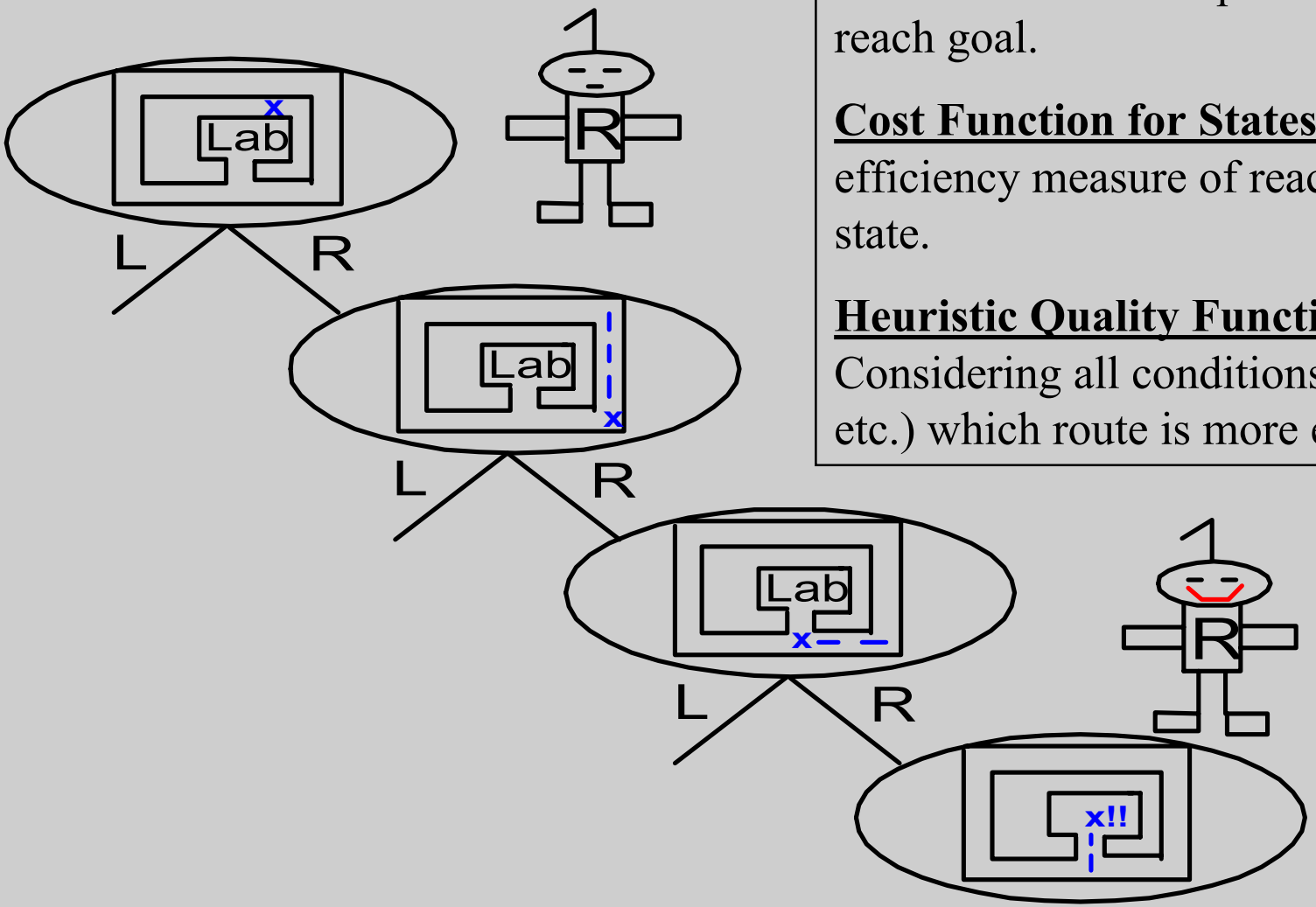
Woozles (Job A): Paul

Wizzles (Job B): Mark

Weazels (Job C): Susan

CPS Process: Function Specifications…

Example: Runaway Robot



Quality Function for Operators: Measure of cost of operations needed to reach goal.

Cost Function for States: Total efficiency measure of reaching particular state.

Heuristic Quality Function for Nodes: Considering all conditions (traffic, speed, etc.) which route is more efficient.

Productivity Problem: CPS Learning Methods

Process:

- Numerical Specification of:
  - > Quality Function for Operators
  - > Cost Function for States
  - > Heuristic Quality Function for Nodes

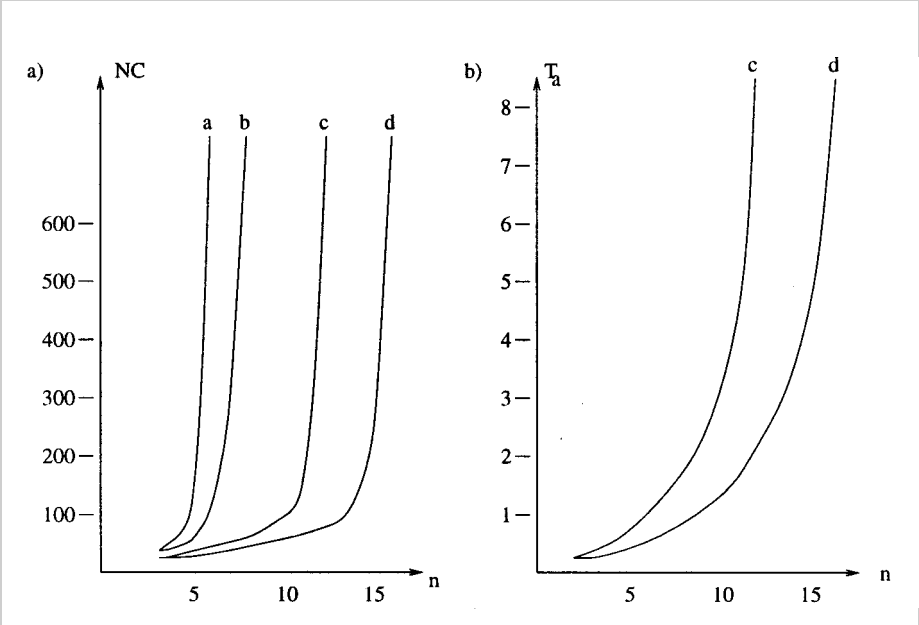
Results:

- Ordered-Search Strategy most efficient
  - > Cutting branches at small depth of tree
- Stopping Moment learning technique effective
  - > (more info later…)
  - > problem's improvement curve has sufficiently large number of steps
- Entire space generated by the system included:
  - > avg. 600 nodes
  - > 530 expansion nodes sufficient for method
  - > 420 nodes for optimal solution

Productivity Problem: CPS Learning Methods

Results:

- Ordered-Search Strategy most efficient



(a) The dependence of number of nodes to find solution, and
(b) The processing time on the problem size for strategies
Where n = dimension of matrix, (workers * jobs)

Strategies:
(a) Breadth First, (b) Depth First, (c) Branch-and-Bound, (d) Ordered Search

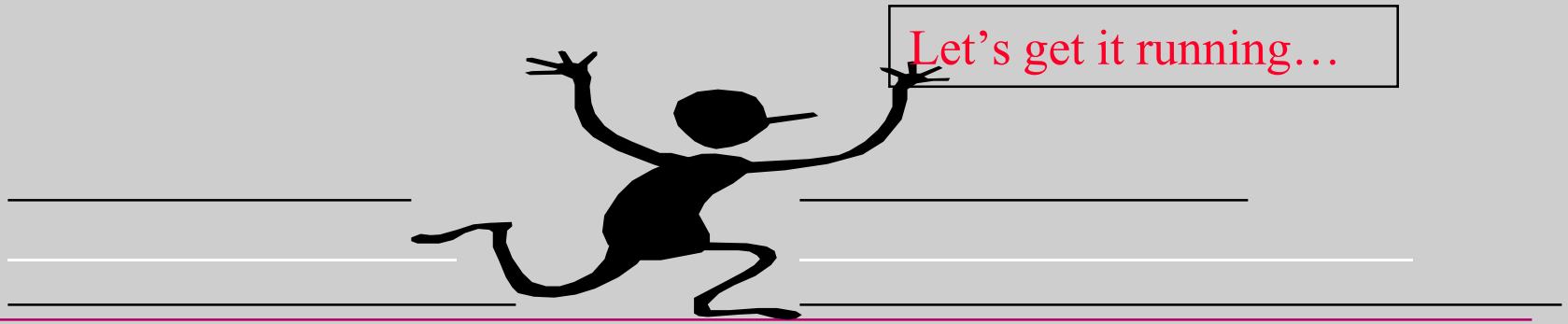
CPS Theory/Methodology

- **Automatically designs application specific solutions for classes of problems: Binary/MVL**
- **State-Space Approach: Problem Class Solutions**

  -Assumption: ANY combinatorial logic problem can be solved by searching some space of known states (I.e. netlists being optimized)
- **CPS Solutions: Intelligent derived Strategy**

  - Rule-Based Systems - GA/GP - Human Logic Experience

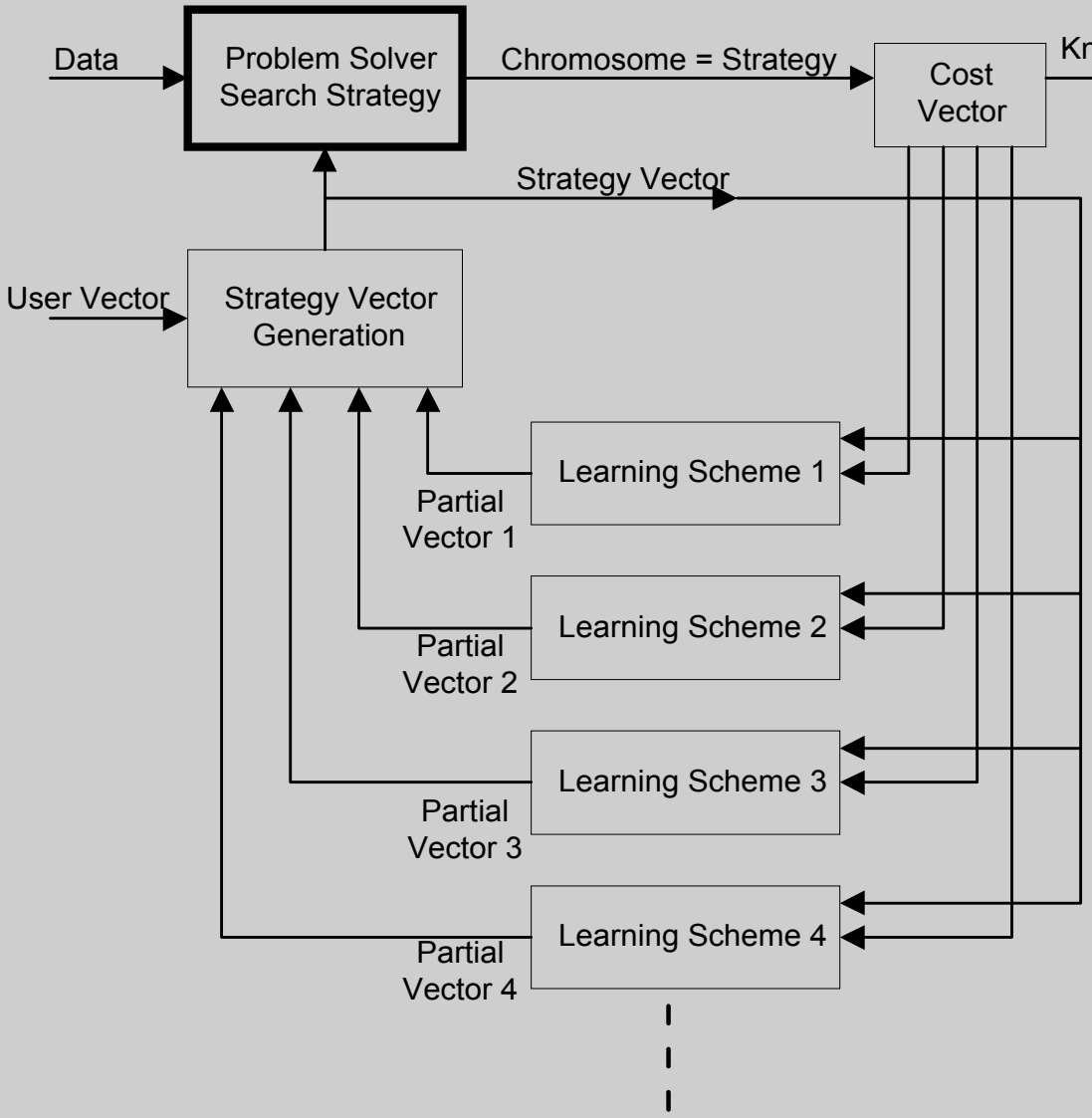
  Human Design Heuristics + State-Space Search



CPS Algorithm Internal Organization

- ***Standard Input/Output***: **Netlist**
- ***Internal Data Representation***: **Trees, Expressions, Netlists**
- ***Two Types of Trees***:
  - **Search Strategy Tree**
    - » **Managed with Tree Properties and Transformations**
    - » **Described by Conditions, Relations, Sorting Functions, Strategy Parameters**
    - » **In-Program Learning: Dynamic Modification of Flags/Coefficients**
  - **Solutions/Data Tree**
    - » **Customized to Application**
    - » **Subset of "Best" Results, (mapped out by tree)**
- ***Data Visualization:*** **Tree, Decision Diagram, K-map, Algebraic Form**

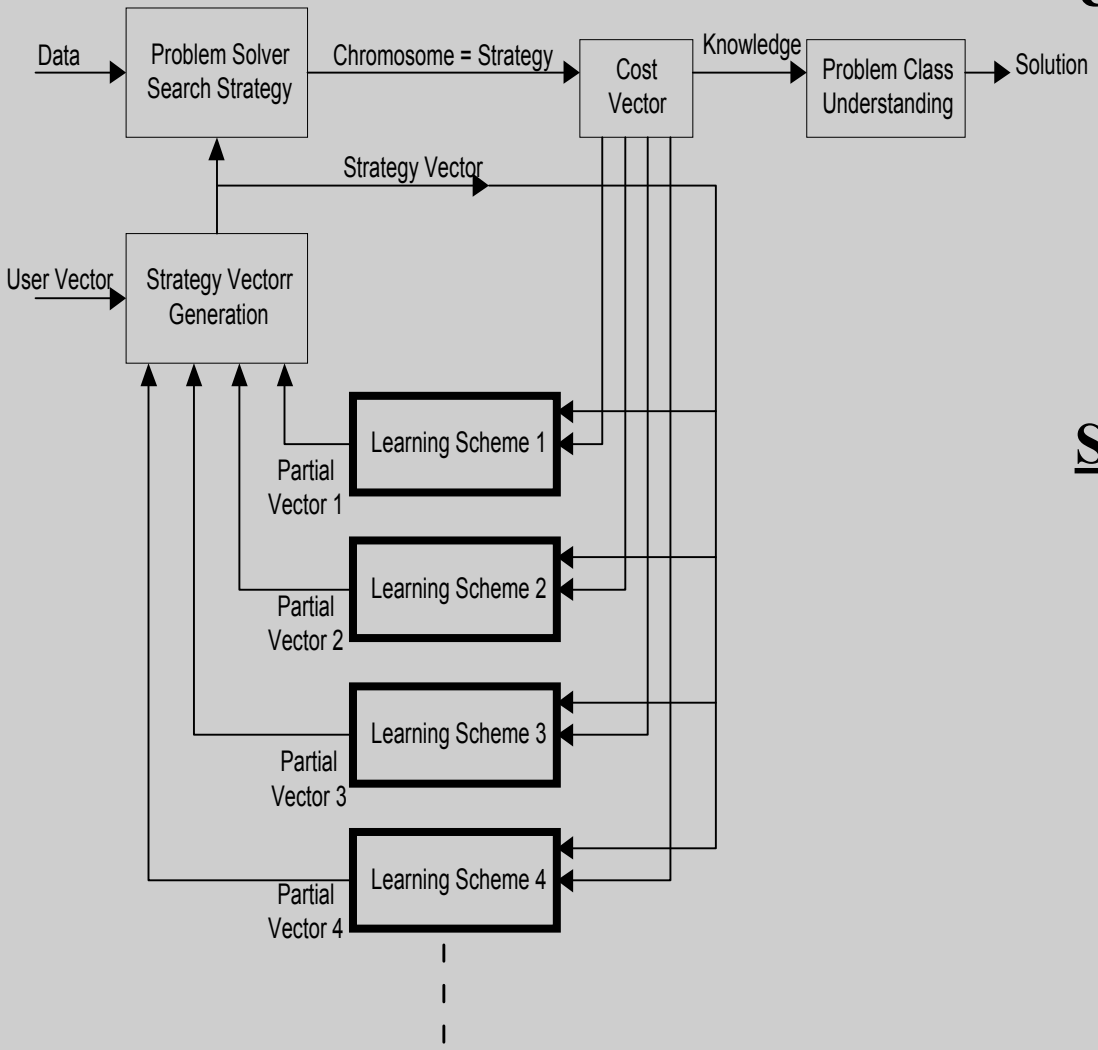
CPS Architecture: Problem Solver Search Strategy



Problem Solver Search Strategy:

- Tree searching technique, with all search methods;
- Search strategy specified by a vector of parameters, treated as a chromosome.

CPS Architecture: Learning Schemes



Learning Schemes:

Use various learning strategies:

- **GA**
- **Perceptron (NN)**
- **Functional Decomposition**
- **Rule-based**
- **etc.**

Strategy Sub-Vectors:

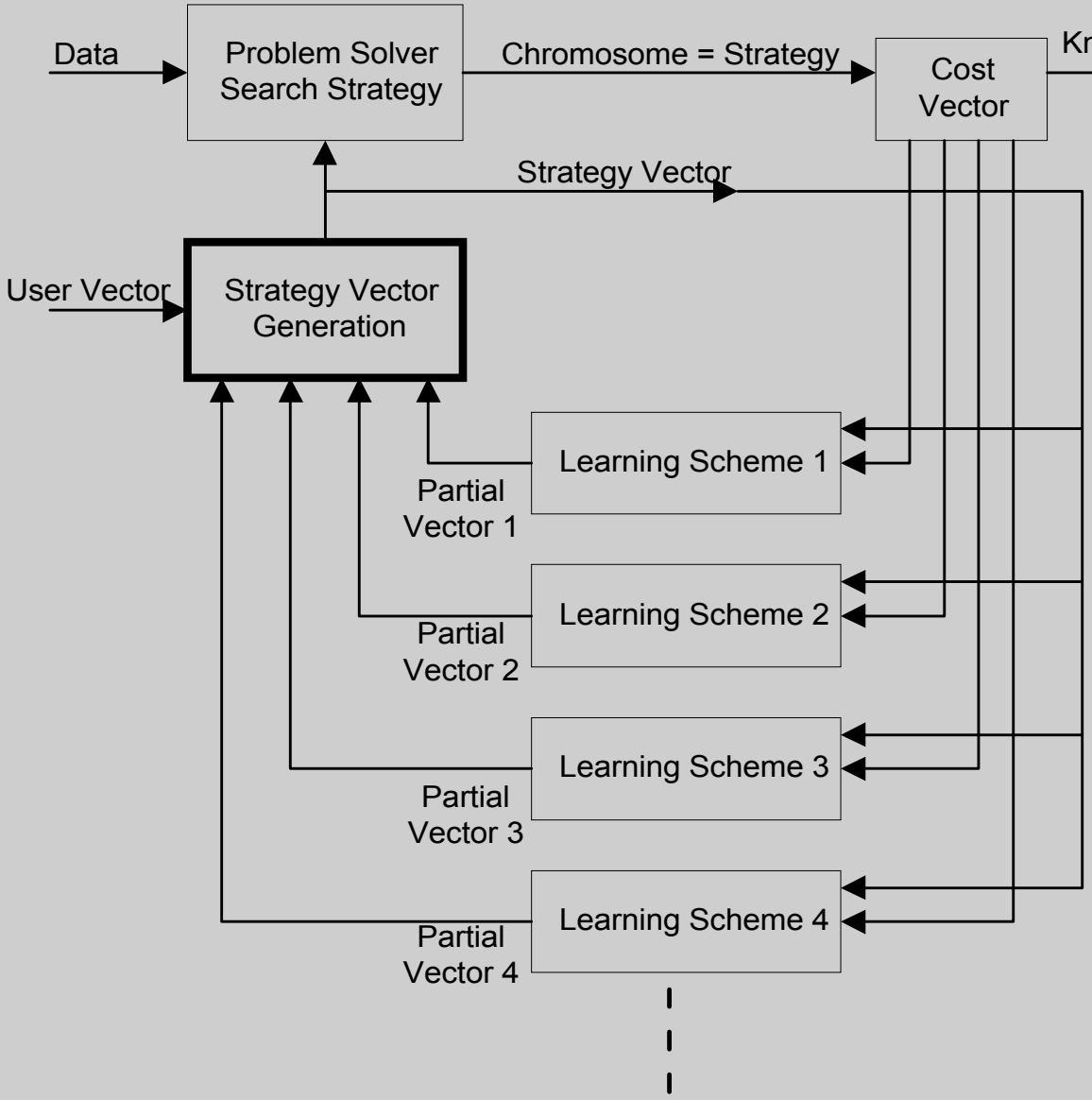
(1) Evaluating functions for operators

(2) Evaluating functions for nodes of the solution tree

(3) Evaluating the Stopping Moment

(4) Probabilities of calling various subroutines/using parameters

CPS Architecture: Strategy Vector Generator



Strategy Vector Generator:

- Solves conflict or competition between learning/search schemes with game concepts, i.e. best cost, voting, etc., decides a "winner"
- Search Strategy encoded in chromosome

Game Concepts: Selection of Search Techniques (Tree)

Two-Stage

- ***Compete for Efficiency - (new tree branch)***

· **Search methods may change during "game", as strategy parameters are altered to improve future behavior**

· **Program remembers "dead ends" search methods**

· **Performance (between state-space search strategies) is compared over partial trees, reiteratively until tree completion**

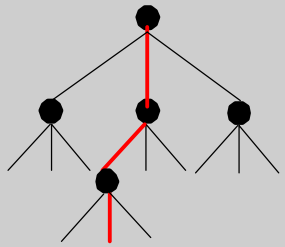
[Idea – from Samuel's ML for Checkers]

2. ***Cooperate for Good Solution – (build whole tree)***

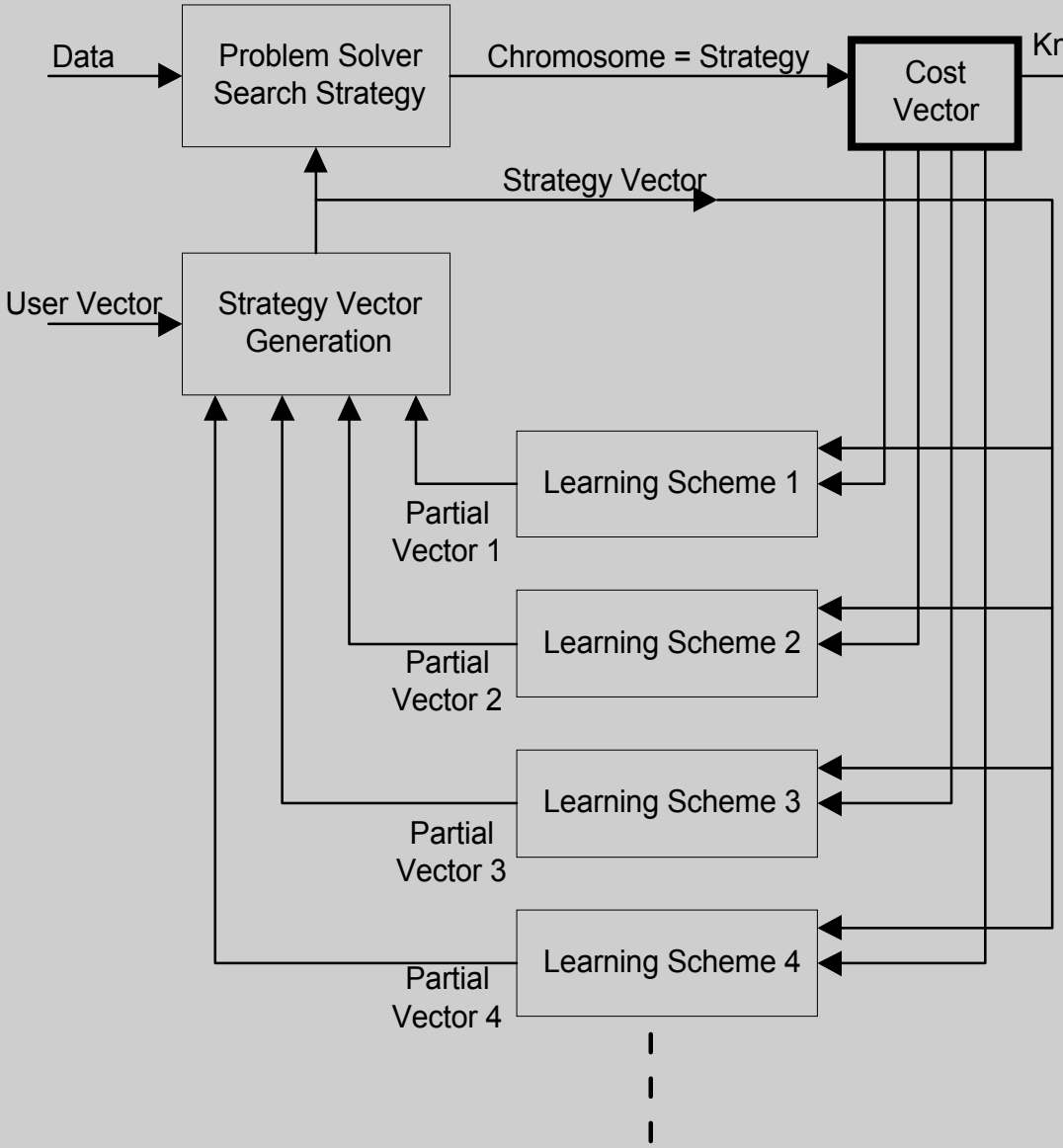
· **Switch Strategies form entire Tree by combining new with previous tree branches**

· **Tree construction for general algorithm for a class of problems, occurs after many learning examples**

Creating the Search Tree



CPS Learning Architecture: Cost Vector/Prob. Class



Cost Vector:

- **Vector of Cost Functions**
- **Applies to the solution for each type of learning scheme**
- **Quantization for a Multi-Criteria Optimization**

Problem Class Under-standing:

- **Through evaluation of a number of search strategies, a good method for solving a class of problems is found.**

CPS Learning Methods for ANY State-Space Search

CPS Learning Methods:

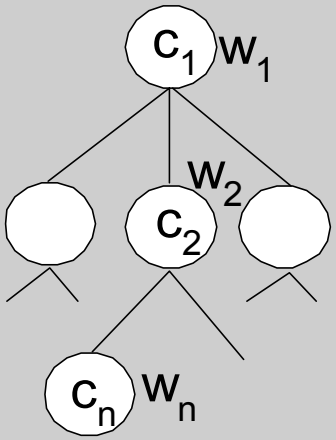
- **Weighted Learning of the Evaluation Function**
- **Learning the Stopping Moment**
- **Learning through Functional Decomposition**

Examples:
Method 1 – Set-Covering Problem
Method 2 – Productivity (Assignment) Problem

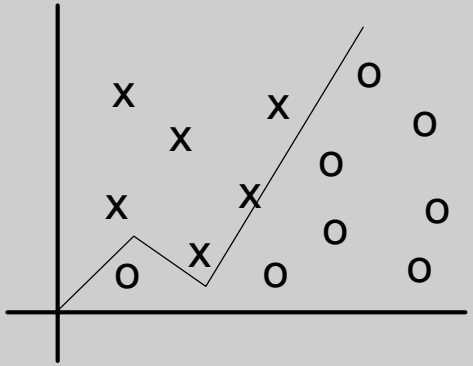
Theory of Learning Methods…

CPS Learning Method 1:
Weighted Learning of the Evaluation Function

- **<u>Evaluator Constructor</u>- System learns the criteria of selecting good operators by using a weighted and decomposed (multi-level) evaluation function and its weight coefficients.**



$$Cost = \sum_{i=1}^{n} w_i c_i$$

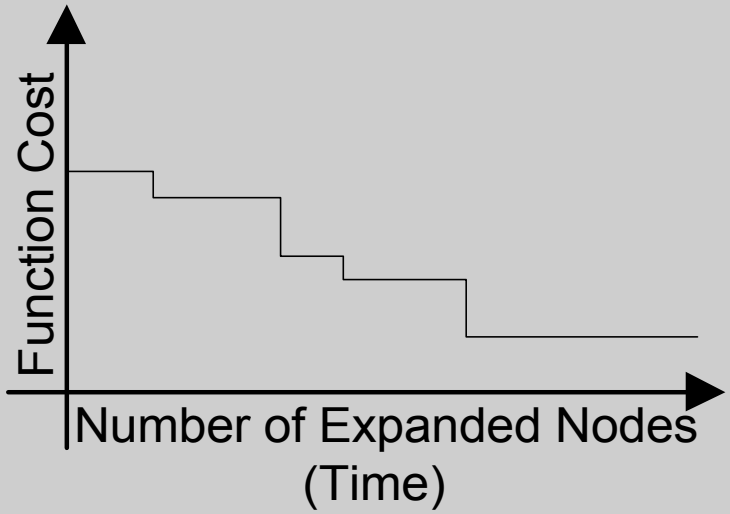


Successively Divides Search Space into good/bad:

- Decomposition
- Linear (Perceptron Neural Net)
- Coefficients
- etc.

CPS Learning Method 2:
Learning the Stopping Moment

2. **Determine Stopping Moment – It is the point when search backtracking can be terminated, because a better solution is very unlikely to be obtained in the future.**
   - **>Normalized shape diagram/signature analysis, or**
   - **>Calculated actual vs. ideal conditions,**
   - **e.g., est. prob. of not loosing optimal solution**



•Assumed: Different combinatorial problems, of the same problem class, regardless of size, exhibit the same learning shape.

CPS Learning Method 2:
Learning the Stopping Moment

***Method Effectiveness based on*:**

- Construction of a Learning System that predicts the effects of further solution space expansion

Assumptions:

- Combinatorial Problem Data Sets have:

> Similar improvement curve shapes

> Statistical dependency between the number of nodes expanded to find better solutions

- Given:

> Sufficiently large number of examples tested

> Normalization for comparison between improvement curves

GRM Minimization Software: GA-GRM

CPS originated due to Problems with Pure Heuristic Search, GA/GP

Our Research:

· GA for Minimization of GRM Forms

What is a GRM?

GRM: Generalized Reed-Muller Form, for 2-level AND-XOR logic --canonical expression, with complete polarity freedom allowed for all variables, in every term

Example: $1 \oplus x_1 \oplus x_2' \oplus x_1'x_2$ is a GRM

Note: There exists only one term for each subset of variables.

Number of Literals in GRM: $n2^{(n-1)}$, for n variable function

Total Number of GRMs: $2^{n*2\wedge(n-1)}$

Traditional GRM Minimization

Complementary Expansions with Substitutions:

x = x' $\oplus$ 1 and x' = x $\oplus$ 1

For example:

$f = 1 \oplus x_3 \oplus x_3x_4 \oplus x_2x_3 \oplus x_1' \oplus x_1'x_3' \oplus x_1x_2x_3 \oplus x_1x_2'x_3'x_4'$

f_1 f_2

Let $f = f_1 \oplus f_2$

Where,

$f_1 = 1 \oplus x_3 \oplus x_3x_4 \oplus x_2x_3 \oplus x_1' \oplus x_1'x_3' \oplus x_1x_2x_3$

$f_2 = x_1x_2'x_3'x_4'$

Minimizing Approach:

Changing the polarity of the last term from $x_1x_2'x_3'x_4'$ to $x_1x_2'x_3'x_4$

Traditional GRM Minimization

Substituting,

$f_2 = x_1x_2'x_3'x_4'$

$= x_1x_2'x_3' \oplus x_1x_2'x_3'x_4$

$=(x_1 \oplus x_1x_2 \oplus x_1x_3 \oplus x_1x_2x_3) \oplus x_1x_2'x_3'x_4$

$=((x_1' \oplus 1) \oplus x_1x_2 \oplus (1 \oplus x_1' \oplus x_3' \oplus x_1'x_3') \oplus x_1x_2x_3) \oplus x_1x_2'x_3'x_4$

$=((x_1' \oplus 1) \oplus x_1x_2(1 \oplus x_1' \oplus (x_3 \oplus 1) \oplus x_1'x_3') \oplus x_1x_2x_3) \oplus$
$x_1x_2'x_3'x_4$

$= 1 \oplus x_3 \oplus x_1'x_3' \oplus x_1x_2 \oplus x_1x_2x_3 \oplus x_1x_2'x_3'x_4$

Solving,

$f = f_1 \oplus f_2$

$f = (1 \oplus x_3 \oplus x_3x_4 \oplus x_2x_3 \oplus x_1' \oplus x_1'x_3' \oplus x_1x_2x_3) \oplus$
$(1 \oplus x_3 \oplus x_1'x_3' \oplus x_1x_2 \oplus x_1x_2x_3 \oplus x_1x_2'x_3'x_4)$

$= x_3x_4 \oplus x_2x_3 \oplus x_1' \oplus x_1x_2 \oplus x_1x_2'x_3'x_4$

{Reduced from 8 to 5 terms}

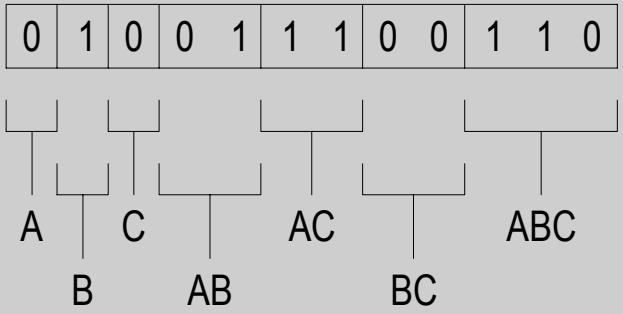
GA-GRM:
GA Encoding for GRM Logic Minimization

- **Example GRM**
  - **Each GRM term must be unique in name and polarity.**
  - **A GRM for f(a,b,c) has the following possible terms.**
  - **By definition, only one term selection is allowed per row.**

Terms	Polarity Vector
a', a	0/1
b', b	0/1
c', c	0/1
a'b', a'b, ab', ab	00-11
a'c', a'c, ac', ac	00-11
b'c', b'c, bc', bc	00-11
a'b'c', a'b'c, a'bc', a'bc, ab'c', ab'c, abc', abc	000-111

GA-GRM Logic Minimization

"Bit String" Polarity Vector Representation, f(a,b,c)



Describes a Valid GRM

$f(a,b,c) = a' \oplus b \oplus c' \oplus a'b \oplus ac \oplus b'c' \oplus abc'$

GA-GRM Logic Minimization

GA-GRM Results:

-Competitive Minimizations for small benchmarks

- Non-Competitive for larger functions ⇒ chromosomal encoding string too long, cumbersome

-GA-GRM interesting design approach because:

> Results are obtained only with genetic operators

> No application specific knowledge/rules

GA Shortcomings (for GA-GRM and ALL other applications):

- Pure GA doesn't incorporate logic rules/methods; must automatically "re-invent" them for each problem

- Considers ALL possible, rather than just reasonable solutions

GRM Minimization Software: GRMIN2

- **Debnath/Sasao's GRMIN2 Outline:**
- **The input is a PSDRM or a SOP.**
- **Simplifies multiple-output functions.**
- **Uses eight rules iteratively to reduce the number of products.**
- **Modifies the cubes repeatedly by replacing a pair of cubes with another one, while keeping the array to represent a GRM.**
- **When reduction of the number of products becomes impossible in the iterative improvement mode, it temporarily increases the number of products and simplifies again.**
- **Obtains lower bounds on the number of products in GRMs and often proves the minimality of the solution.**

GRM Minimization Software: GRMIN2

GRMIN2 Simplification Rules

Rule 1: X-MERGE $X^A \oplus X^B \Rightarrow X^{(A \oplus B)}$

Rule 2: RESHAPE $X^A Y^B \oplus X^C Y^D \Rightarrow X^A Y^{(B \cap D')} \oplus X^{(A \cup C)} Y^D$,
if $(A \cap C = \varnothing, B \supset D)$

Rule 3: DUAL-COMPLEMENT

$X^A Y^B \oplus X^C Y^D \Rightarrow X^{(A' \cap C)} Y^B \oplus X^C Y^{(B \cap D')}$,
if $(A \subset C, B \supset D)$

Rule 4: X-EXPAND-1 $X^A Y^B \oplus X^C Y^D \Rightarrow X^A Y^{(B \cup D)} \oplus X^{(A \cup C)} Y^D$

$\Rightarrow X^{(A \cup C)} Y^B \oplus X^C Y^{(B \cup D)}$,

if $(A \cap C = \varnothing, B \cap D = \varnothing)$

GRM Minimization Software: GRMIN2

GRMIN2 Simplification Rules

Rule 5: X-EXPAND-2 $X^{A}Y^{B} \oplus X^{C}Y^{D} \Rightarrow X^{(A \cup C)}Y^{B} \oplus X^{C}Y^{(B \cap D')}$,

if $(A \cap C = \varnothing, B \supset D)$

Rule 6: X-REDUCE-1 $X^{A}Y^{B} \oplus X^{C}Y^{D} \Rightarrow X^{(A \cap C')}Y^{B} \oplus X^{C}Y^{(D \cap B')}$,

if $(A \supset C, B \subset D)$

Rule 7: X-REDUCE-2 $X^{A}Y^{B} \oplus X^{C}Y^{D} \Rightarrow X^{(A \cap C')}Y^{B} \oplus X^{C}Y^{(B \cap D')}$,

if $(A \supset C, B \supset D)$

Rule 8: X-COMPLEMENT

$$X^{A} \Rightarrow X^{A'} \oplus X^{P}$$

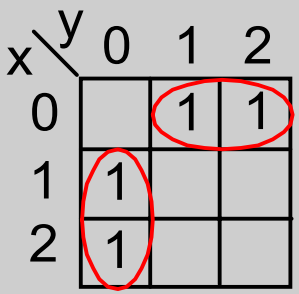
Example of GRMIN2 - Rule 4

X-EXPAND-1: $X^{A}Y^{B} \oplus X^{C}Y^{D} \Rightarrow X^{A}Y^{(B\cup D)} \oplus X^{(A\cup C)}Y^{D}$ (1),

$\Rightarrow X^{(A\cup C)}Y^{B} \oplus X^{C}Y^{(B\cup D)}$ (2),

Condition: If ($A \cap C = \varnothing$, $B \cap D = \varnothing$)

Given:



$f(x,y) = X^{1,2}Y^{0} \oplus X^{0}Y^{1,2}$

Form: $X^{A}Y^{B} \oplus X^{C}Y^{D}$,

Let A = 1,2, B = 0, C = 0, D = 1,2

(Rule 4 conditions hold)

Applying Rule 4:

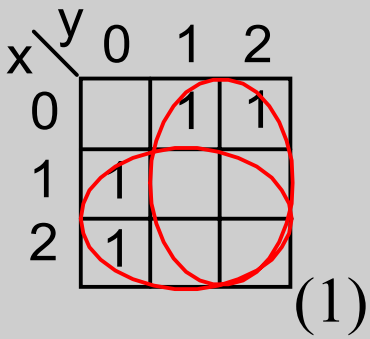
$X^{1,2}Y^{0} \oplus X^{0}Y^{1,2}$

$\Rightarrow X^{1,2}Y^{(0\cup 1,2)} \oplus X^{(1,2\cup 0)}Y^{1,2}$ (1)

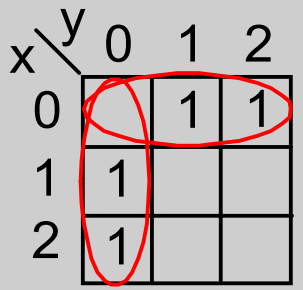
$\Rightarrow X^{1,2} \oplus Y^{1,2}$

$\Rightarrow X^{(1,2\cup 0)}Y^{0} \oplus X^{0}Y^{(0\cup 1,2)}$ (2)

$\Rightarrow Y^{0} \oplus X^{0}$



(1)



(2)

GRMIN2 Influence on CPS

- GRMIN2 Rules are a subset of ESOP rules for the GRM
- Rules are human formulated
- But, CPS may be an improvement over the GRMIN2 by:
  - Combining these human rules with automated search processes
  - Automatically selecting other logic rule subsets and/or strategies
  - Determining an optimal application/combination of these rules
- Further testing of CPS is necessary

Research Experience:
Genetic Operators vs. Human Rules

CPS combines Automatic Learning with Human Experience

<u>Genetic Operators</u>:

- **A simple and practically blind mechanism of Nature**
- **Elegance of Design; Form/Function; Beauty**
- **Universal Applicability**

∴ Basis is that Nature is the pure theory of Darwin; Incorporation of Baldwinian and/or Lamarckian Learning possible

<u>Human Rules (Thought)</u>:

- **Higher level, with multiple goals and much complexity**
- **Logic algorithms are optimal/mathematically sophisticated**

∴ A high quality of learning results that includes: knowledge generalization, discovery, no over-fitting, small learning errors

Conclusion

- **CPS is a new approach for general problem solving and learning, for classes of binary and MVL logic optimization problems**

  - Combines Search Strategies, through competition/cooperation

  - Intelligent, superset "toolbox" of methodologies

  - "Game" concepts build on strengths while reducing weaknesses of different learning methodologies

Conclusion

- New Learning Methods are utilized in CPS
  - Weighted Learning of the Evaluation Function
  - Determination of Stopping Moment
- Initial Representative Applications on Combinatorial Logic
  - Set Covering Problem
  - Productivity (Assignment) Problem
  - Results: Substantial improvement shown with CPS, as compared with traditional search methodologies

References

[1] Debnath, D. and T. Sasao, "GRMIN2: A Heuristic Simplification Algorithm for Generalised Reed-Muller Expressions", IEE Proceedings – Comput. Digit. Tech., Vol. 143, No. 6, November 1996.

[2] Debnath D. and T. Sasao, "GRMIN: A Heuristic Simplification Algorithm for Generalized Reed-Muller Expressions", Proc. of Reed-Muller '95, IFIP WG 10.5, Workshop on Applications of the Reed-Muller Expansion in Circuit Design, Aug. 27-29, 1995, Makuhari, Chiba, Japan.

[3] Dill, K. and M. Perkowski, "Evolutionary Minimization of Generalized Reed-Muller Forms", Proc. of the Int. Conf. on Computational Intelligence and Multimedia 1998, (ICCIMA'98), Monash University, Churchill, VIC, Australia, Feb. 9-11, 1998.

References

[4] Dill, K. and M. Perkowski, "Minimization of Generalized Reed-Muller Forms with Genetic Operators", Proc. of the Genetic Programming Conf., July 1997, Stanford Univ., (San Francisco, California: Morgan Kaufmann Publishers, 1997).

[5] Dill, K., J. Herzog, and M. Perkowski, "Genetic Programming and its Application to the Synthesis of Digital Logic", Proc. of the PACRIM '97 Conference, Victoria, Canada, Aug. 20-22, 1997, (Piscataway, New Jersey: IEEE, 1997).

[6] Dill, K., forthcoming Ph.D. Dissertation, Dept. of Electrical and Computer Engineering, Portland State University, Portland, Oregon.

References

[7] Dill, K., Growing Digital Circuits: Logic Synthesis and Minimization with Genetic Operators, Master of Science Thesis, Dept. of Electrical and Computer Engineering, Oregon State University, June 1997.

[8] Files, C., R. Drechsler, and M. Perkowski, "Functional Decomposition of MVL Functions using Multi-Valued Decision Diagrams", Presentation, Int. Symp. on Multi-Valued Logic 1997, (ISMVL'97).

[9] Goldberg, D. E., Genetic Algorithms in Search, Optimization, & Machine Learning, (New York: Addison-Wesley Publishing Company, Inc., 1989).

[10] Koza, J. R., Genetic Programming: On the Programming of Computers by Means of Natural Selection, (Cambridge, Massachusetts: The MIT Press, 1992).

References

[11] Perkowski, M., J. Liu, and J. Brown, "Chapter 11 – Rapid Software Prototyping: CAD Design of Digital CAD Algorithms", in Progress in Computer-Aided VLSI Design, Vol. 1: Tools, ed. George W. Zobrist, Univ. of Missouri-Rolla, (Norwood, New Jersey: Ablex Publishing Corporation, 1989), pp. 353-401.

[12] Samuel, A. L., "Some Studies in Machine Learning Using the Game of Checkers", IBM Journal of Research and Development, Vol. 3, No. 2, pp. 210-229, July 1959.

[13] Samuel, A. L., "Some Studies in Machine Learning Using the Game of Checkers. II – Recent Progress", IBM Journal of Research and Development, Vol. 11, No. 6, pp. 601-617, November 1967.

References

[14] Zeng, X., M. Perkowski, K. Dill, and A. Sarabi, "Approximate Minimization of Generalized Reed-Muller Forms", Proc. of Reed-Muller'95, IFIP WG 10.5 Workshop on Applications of the Reed-Muller Expansion in Circuit Design, Aug. 27-29, 1995, Makuhari, Chiba, Japan.