A Hot Method for Synthesising Cool Controllers

I. Husien, N. Berthier and S. Schewe

University of Liverpool

July 14, 2017

Outline

1. Motivation
2. Discrete Controller Synthesis
3. Generic Search Techniques
4. Search-based Symbolic Discrete Controller Synthesis
5. Experimental Evaluations
6. Conclusion

Motivation: Solving Symbolic DCS Problems by using General Search Techniques

Symbolic Discrete Controller Synthesis (DCS) Problem

Given:
- System Model S with means of control
- Desired Objective Behaviours ω

Compute:
- Control Strategy *s.t* S controlled accordingly fulfils ω

Motivation: Solving Symbolic DCS Problems by using General Search Techniques

Symbolic Discrete Controller Synthesis (DCS) Problem

Given:
- System Model S with means of control
- Desired Objective Behaviours ω

Compute:
- Control Strategy *s.t* S controlled accordingly fulfils ω

General Search Techniques

- Explore the space of all Candidate Solutions by performing Local Changes to candidates selected by means of a measure of their Fitness

  e.g., Genetic Programming and Simulated Annealing

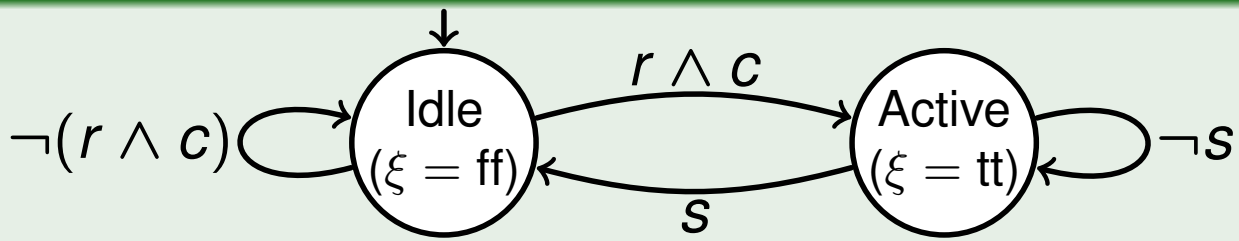
Related, Inspirational Works

- Genetic Programming already investigated for Program Synthesis [Clark and Jacob, 2001, Henderson et al., 2003, Johnson, 2007, Katz and Peled, 2008, 2009, 2011]
- Simulated Annealing performs better than Genetic Programming for Program Synthesis [Husien and Schewe, 2016]

⇝ What about for solving Symbolic DCS Problems?

DCS by Example: System, Objectives, & Strategies

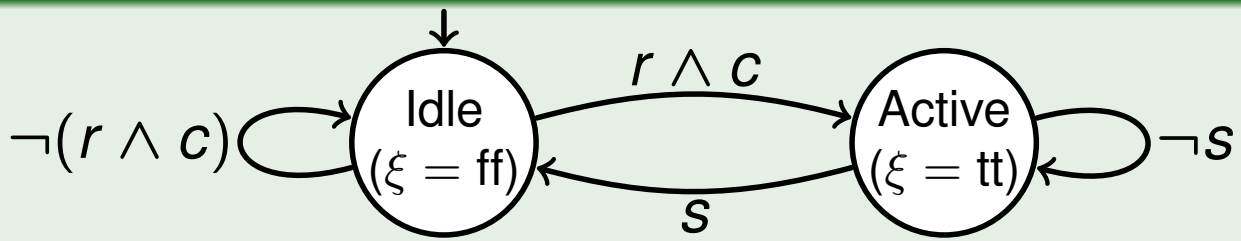
Two-state Task



- Non-controllable input variables: $\{r, s\}$
- Controllable input variables: $\{c\}$
- State variables: $\{\xi\}$

DCS by Example: System, Objectives, & Strategies

Two-state Task



- Non-controllable input variables: $\{r, s\}$
- Controllable input variables: $\{c\}$
- State variables: $\{\xi\}$

Safety Objective as CTL Formula

- Never enter Active while s holds: $AG(s \Rightarrow AX\neg\xi)$

Symbolic Discrete Controller Synthesis Problem

Symbolic DCS Problem

Given:

- Symbolic Finite-state System S with Controllable input variables
- Set of Objective CTL Formulae ω involving the variables of S

Compute:

- A Strategy for assigning values to the Controllable input variables *s.t* S satisfies every objective in ω

Symbolic Discrete Controller Synthesis Problem

Symbolic DCS Problem

Given:

- Symbolic Finite-state System S with Controllable input variables
- Set of Objective CTL Formulae ω involving the variables of S

Compute:

- A Strategy for assigning values to the Controllable input variables *s.t* S satisfies every objective in ω

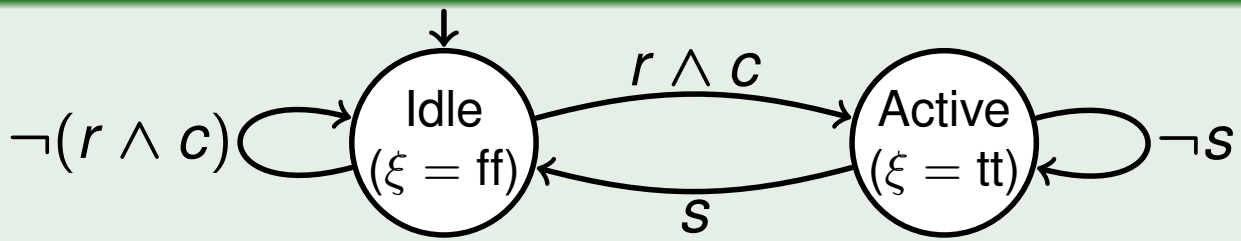
Two Families of Symbolic Strategies

Non-deterministic Propositional Predicate involving variables of S

Deterministic Assignments to every Controllable input variable of S

DCS by Example: System, Objectives, & Strategies

Two-state Task



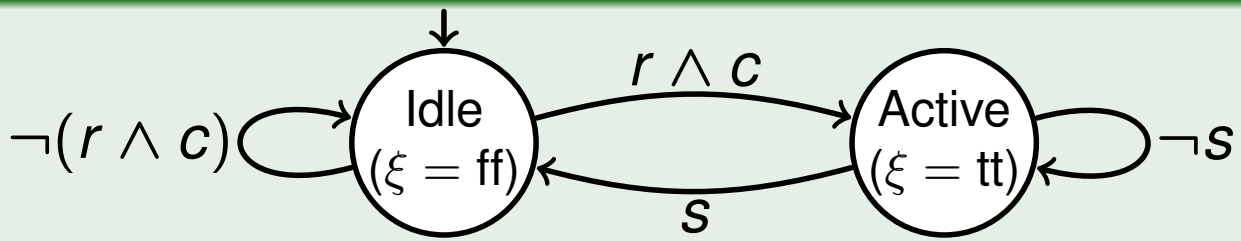
- Non-controllable input variables: $\{r, s\}$
- Controllable input variables: $\{c\}$
- State variables: $\{\xi\}$

Safety Objective as CTL Formula

- Never enter Active while s holds: $AG(s \Rightarrow AX\neg\xi)$

DCS by Example: System, Objectives, & Strategies

Two-state Task



- Non-controllable input variables: $\{r, s\}$
- Controllable input variables: $\{c\}$
- State variables: $\{\xi\}$

Safety Objective as CTL Formula

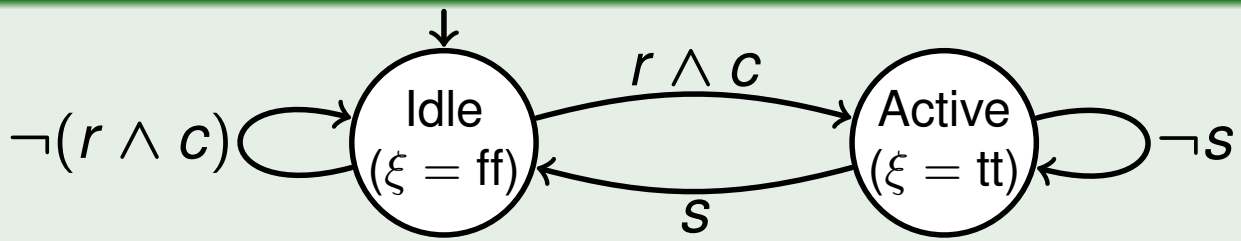
- Never enter Active while s holds: $AG(s \Rightarrow AX\neg\xi)$

Correct Strategies for Choosing Values of c

- Choose c *s.t* $(\neg\xi \wedge s \wedge r \Rightarrow \neg c)$ holds (Non-deterministic)

DCS by Example: System, Objectives, & Strategies

Two-state Task



- Non-controllable input variables: $\{r, s\}$
- Controllable input variables: $\{c\}$
- State variables: $\{\xi\}$

Safety Objective as CTL Formula

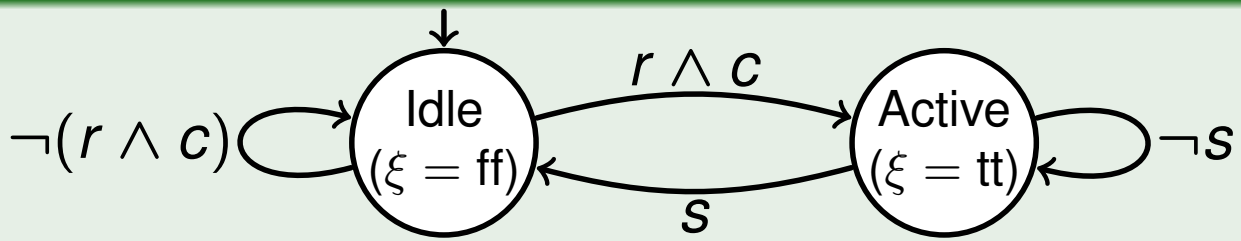
- Never enter Active while s holds: $AG(s \Rightarrow AX\neg\xi)$

Correct Strategies for Choosing Values of c

- Choose c *s.t* $(\neg\xi \wedge s \wedge r \Rightarrow \neg c)$ holds (Non-deterministic)
- Set c to $(\xi \vee \neg r \vee \neg s)$ (Deterministic)

DCS by Example: System, Objectives, & Strategies

Two-state Task



- Non-controllable input variables: $\{r, s\}$
- Controllable input variables: $\{c\}$
- State variables: $\{\xi\}$

Safety Objective as CTL Formula

- Never enter Active while s holds: $AG(s \Rightarrow AX\neg\xi)$

Correct Strategies for Choosing Values of c

- Choose c *s.t* $(\neg\xi \wedge s \wedge r \Rightarrow \neg c)$ holds (Non-deterministic)
- Set c to $(\xi \vee \neg r \vee \neg s)$ (Deterministic)
- Set c to ff (Deterministic)

Generic Search Techniques

Genetic Programming

- Maintain a Population P of λ Candidates
- Until a candidate solution is found in P:
  Select $\mu \ll \lambda$ candidates based on their Fitness;
  Generate new candidates by means of Mutations (and Crossovers)

Generic Search Techniques

Genetic Programming

- Maintain a Population P of λ Candidates
- Until a candidate solution is found in P:
  Select $\mu \ll \lambda$ candidates based on their Fitness;
  Generate new candidates by means of Mutations (and Crossovers)

Simulated Annealing

- Temperature θ decreasing over time ("Cooling schedule")
- Unless the Candidate x is a solution:
  Mutate x to obtain x';
  Repeat with x' instead of x if $\mathrm{Fitness}(x') \geqslant \mathrm{Fitness}(x)$,
  or with a probability increasing with temperature (*i.e.,* decreasing with time) and decreasing with their fitness gap

Generic Search Techniques

Genetic Programming

- Maintain a Population P of λ Candidates
- Until a candidate solution is found in P:
  Select $\mu \ll \lambda$ candidates based on their Fitness;
  Generate new candidates by means of Mutations (and Crossovers)

Simulated Annealing

- Temperature θ decreasing over time ("Cooling schedule")
- Unless the Candidate x is a solution:
  Mutate x to obtain x';
  Repeat with x' instead of x if $\text{Fitness}(x') \geqslant \text{Fitness}(x)$,
  or with a probability increasing with temperature (*i.e.,* decreasing with time) and decreasing with their fitness gap

Search-based Symbolic Discrete Controller Synthesis

Issues

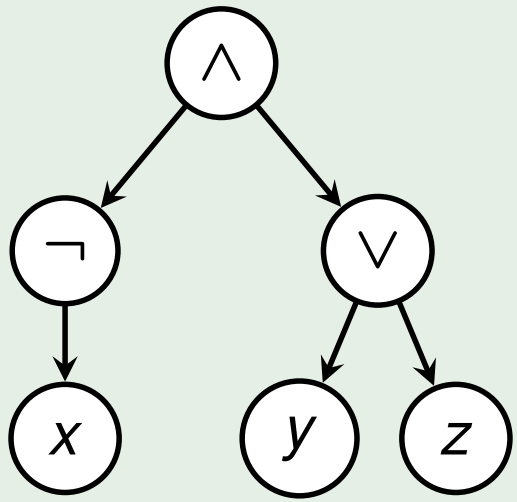
- How to Represent Candidate Strategies?
- How to Mutate Candidates?
- How to Measure the Fitness of a Candidate?
- How to Explore the Search Space Efficiently?

Representing Candidates

Trees built according to the Grammar of Propositional Predicates

- **Leaves** Labelled with either a variable, or a constant in (tt, ff);
- **Nodes** Labelled with $\neg$, $\vee$, or $\wedge$

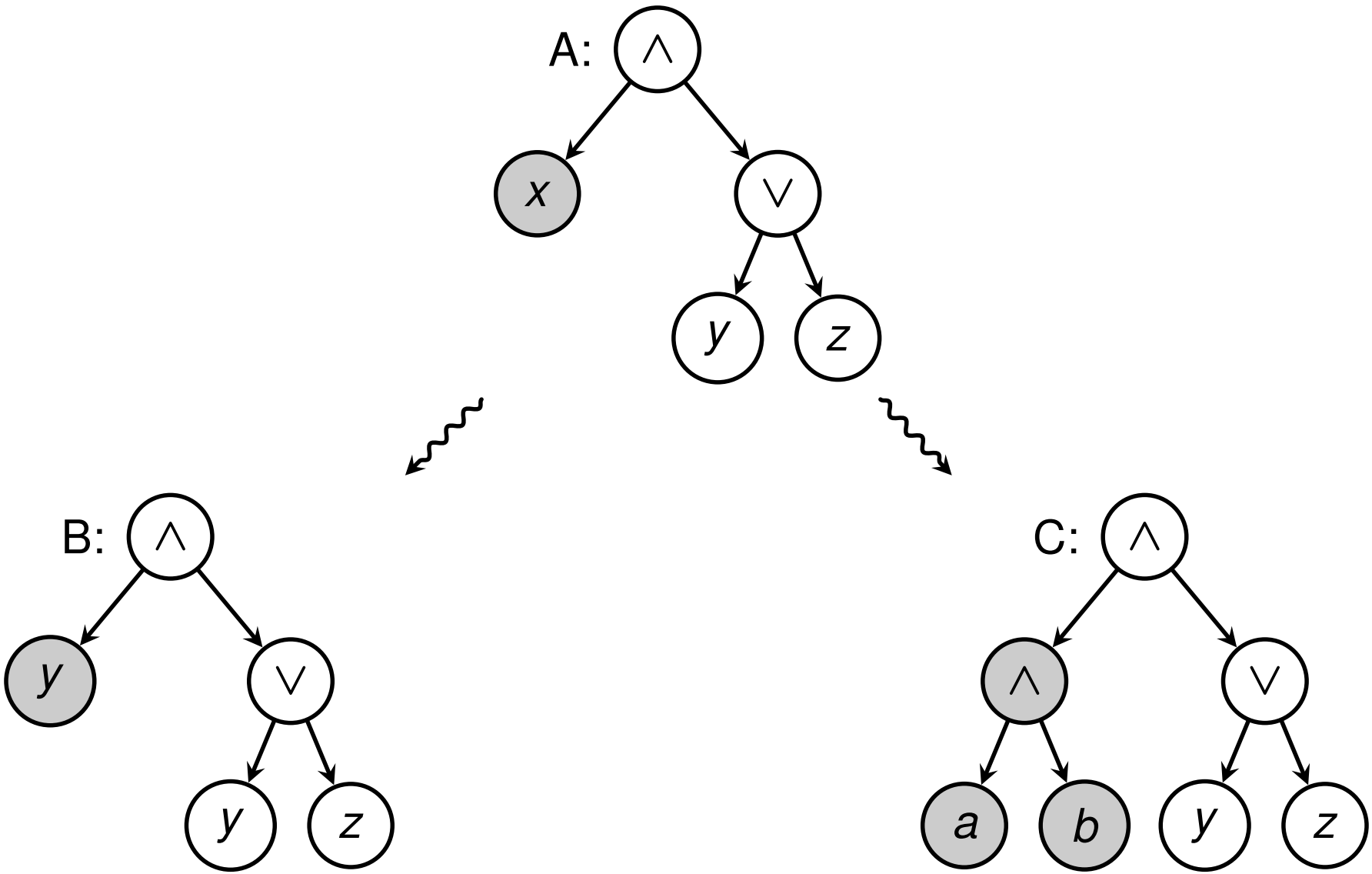
e.g., $\neg x \wedge (y \vee z)$



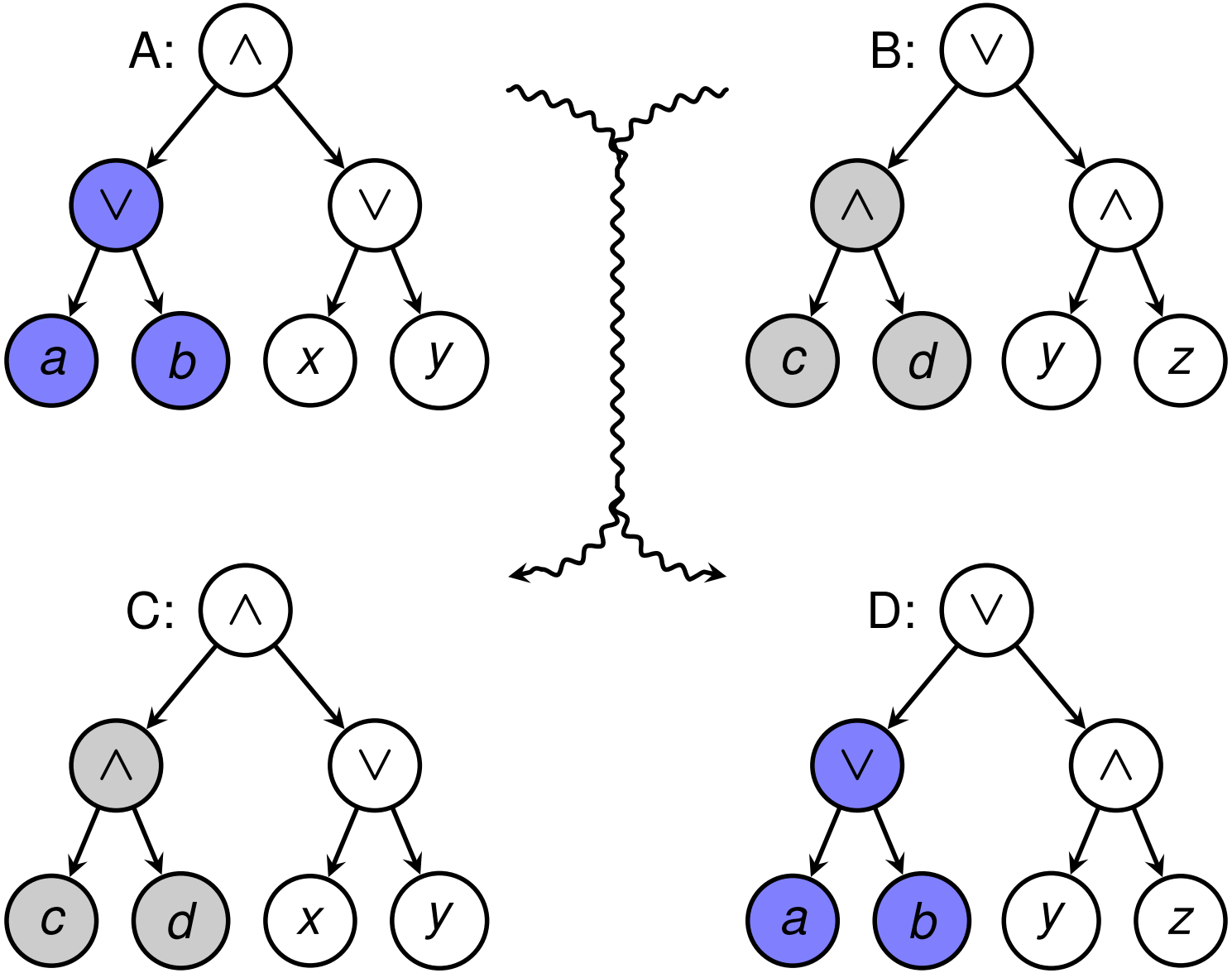
Contrary to Classical Symbolic DCS Algorithms, we directly seek Deterministic Strategies (*i.e.,* Assignments to controllable variables)

$\rightsquigarrow$ One candidate strategy $\equiv$ One tree per controllable variable

How to Make Changes? Example Mutations



How to Make Changes? Example Crossover



Model Checking as a Fitness Function

Deriving Partial Objectives from Target Objectives ω

- ω' Obtained by Substituting one Universal Path Quantifier occurring Positively of Objectives of ω, if any, by an Existential one: $A\psi \rightsquigarrow E\psi$
- ω" Ditto with Objectives of ω'

Model Checking as a Fitness Function

Deriving Partial Objectives from Target Objectives ω

- ω' Obtained by Substituting one Universal Path Quantifier occurring Positively of Objectives of ω, if any, by an Existential one: $A\psi \rightsquigarrow E\psi$
- ω" Ditto with Objectives of ω'

Example Derived Partial Objective Formulae

Given $\omega = \{AG(s \Rightarrow AX\neg\xi)\}$:

$$\omega' = \{EG(s \Rightarrow AX\neg\xi), AG(s \Rightarrow EX\neg\xi)\}$$
$$\omega'' = \{EG(s \Rightarrow EX\neg\xi)\}$$

Model Checking as a Fitness Function (Cont'd)

Measuring the Fitness of a Candidate Strategy Γ

- Build System S' controlled according to Γ
- Model-check S' against objectives of ω, ω', and ω''
- Add m points per satisfied objective of ω
- Add m' ($< m$) points per satisfied objective of ω'
- Add m'' ($< m'$) points per satisfied objective of ω''

Six Search Algorithms for Solving DCS Problems

Simulated Annealing

Rigid SA Considers all objective properties to determine fitness

Safety-first SA Safety properties are considered first

Six Search Algorithms for Solving DCS Problems

Simulated Annealing

Rigid SA Considers all objective properties to determine fitness

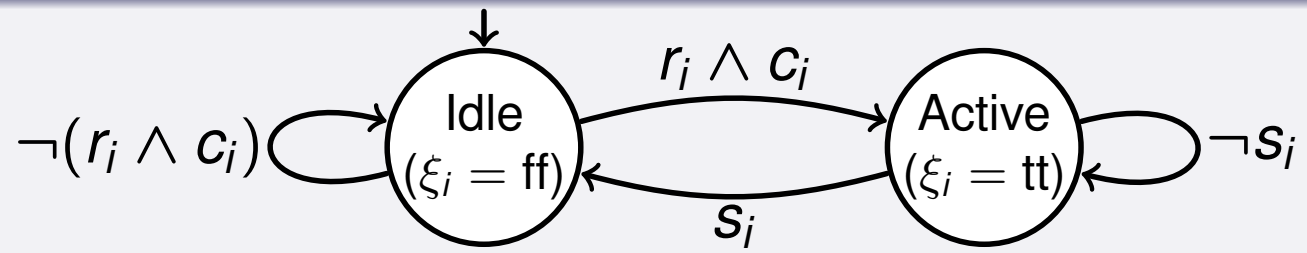
Safety-first SA Safety properties are considered first

Genetic Programming

		Crossovers?	
		no	yes
Safety-first?	no	GP	GP w. CO
	yes	Safety-first GP	Safety-first GPw. CO

Scalable Benchmarks

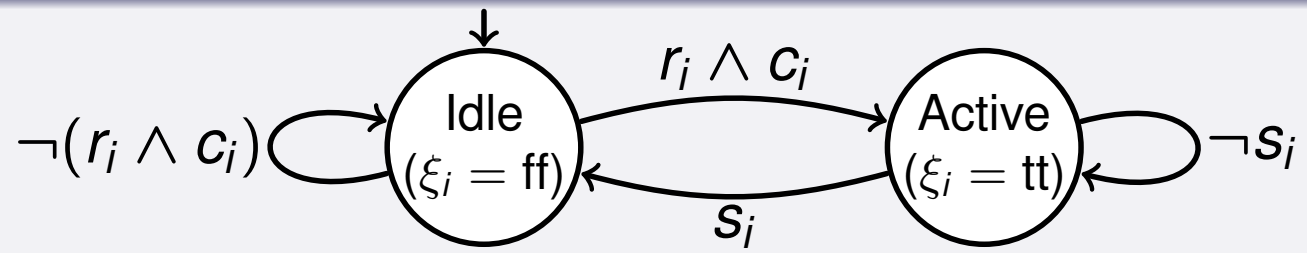
N–Tasks



- Non-controllable input variables: $\{r_1, \ldots, r_N, s_1, \ldots, s_N\}$
- Controllable input variables: $\{c_1, \ldots, c_N\}$

Scalable Benchmarks

N–Tasks



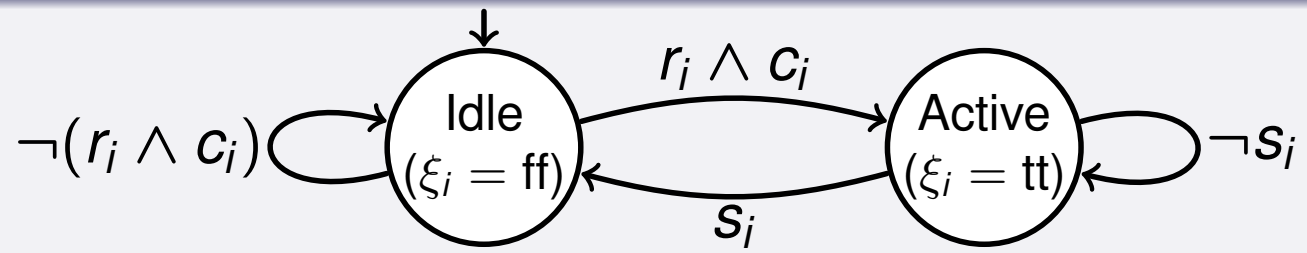
- Non-controllable input variables: $\{r_1, \ldots, r_N, s_1, \ldots, s_N\}$
- Controllable input variables: $\{c_1, \ldots, c_N\}$

Safety Objective: *Mutual Exclusion* between Active tasks

$$\left\{ AG\left(\bigwedge\nolimits_{(i,j)\in\{1,\ldots,N\}^2, i\neq j} \neg\left(\xi_i \wedge \xi_j\right)\right)\right\}$$

Scalable Benchmarks

N–Tasks



- Non-controllable input variables: $\{r_1, \ldots, r_N, s_1, \ldots, s_N\}$
- Controllable input variables: $\{c_1, \ldots, c_N\}$

Safety Objective: *Mutual Exclusion* between Active tasks

$$\left\{ AG\left(\bigwedge\nolimits_{(i,j)\in\{1,\ldots,N\}^2, i\neq j} \neg\left(\xi_i \wedge \xi_j\right)\right)\right\}$$

Liveness Objectives: Every Idle task *Eventually* becomes Active

$$\bigcup\nolimits_{i\in\{1,\ldots,N\}} \{AG((\neg\xi_i \wedge r_i) \Rightarrow AF\xi_i)\}$$

Experimental Setup

Fitness Measure: Points

- $m = 100$ for satisfied Target Objectives (ω)
- $m' = 80$ and $m'' = 10$ for Satisfied Partial Objectives (ω' and ω'')

Parameters for Simulated Annealing

- Initial temperature of 20,000; Decreasing by 0.8 at each iteration

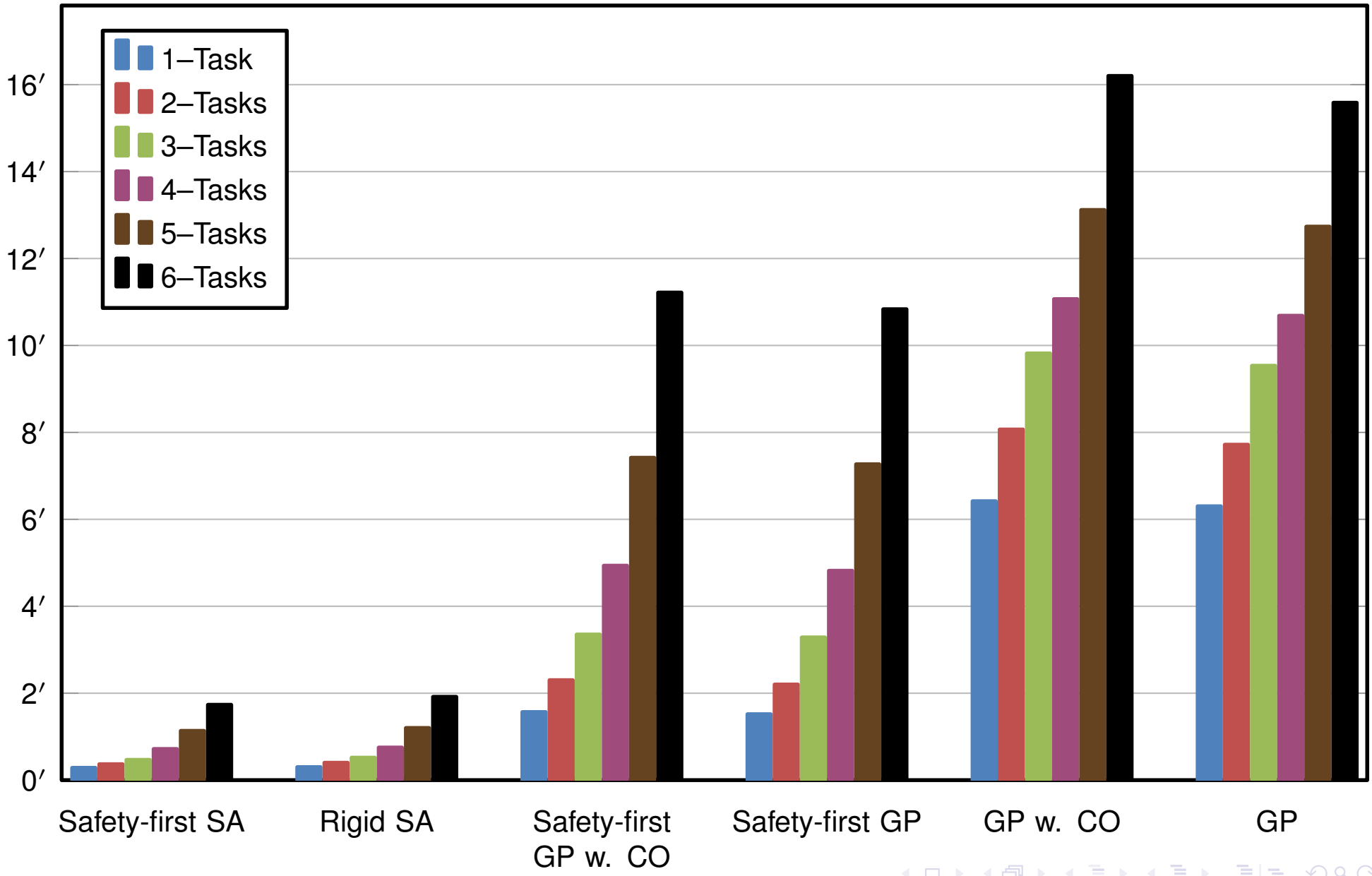
$\rightsquigarrow$ Search ends after 25,000 iterations

Parameters for Genetic Programming

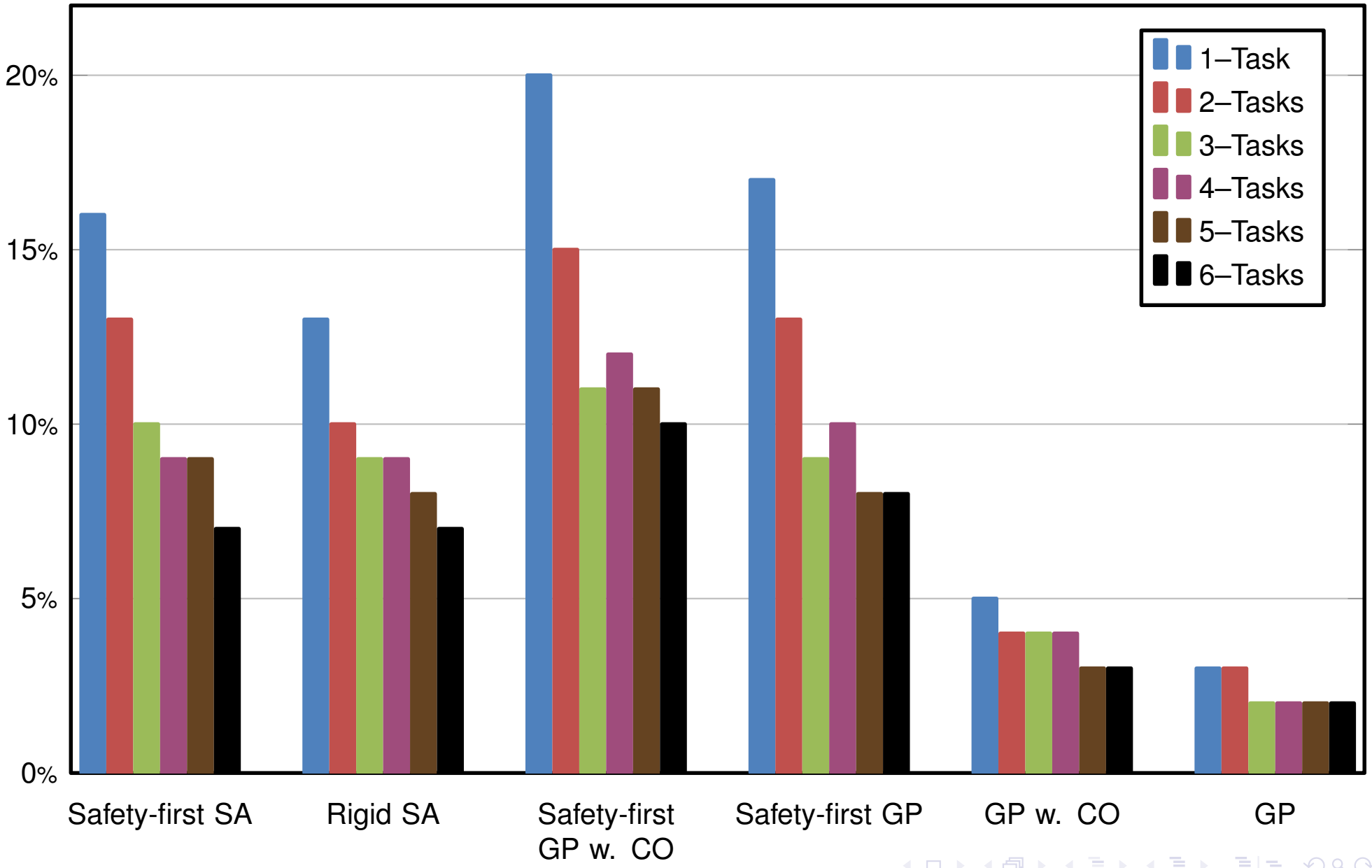
- Population of 150 candidates; Keep 5 at each iteration
- Abort after 2,000 iterations

- Using NuSMV Model-checker
- Running on Intel core i7 3.40 GHz CPU with 16GB RAM

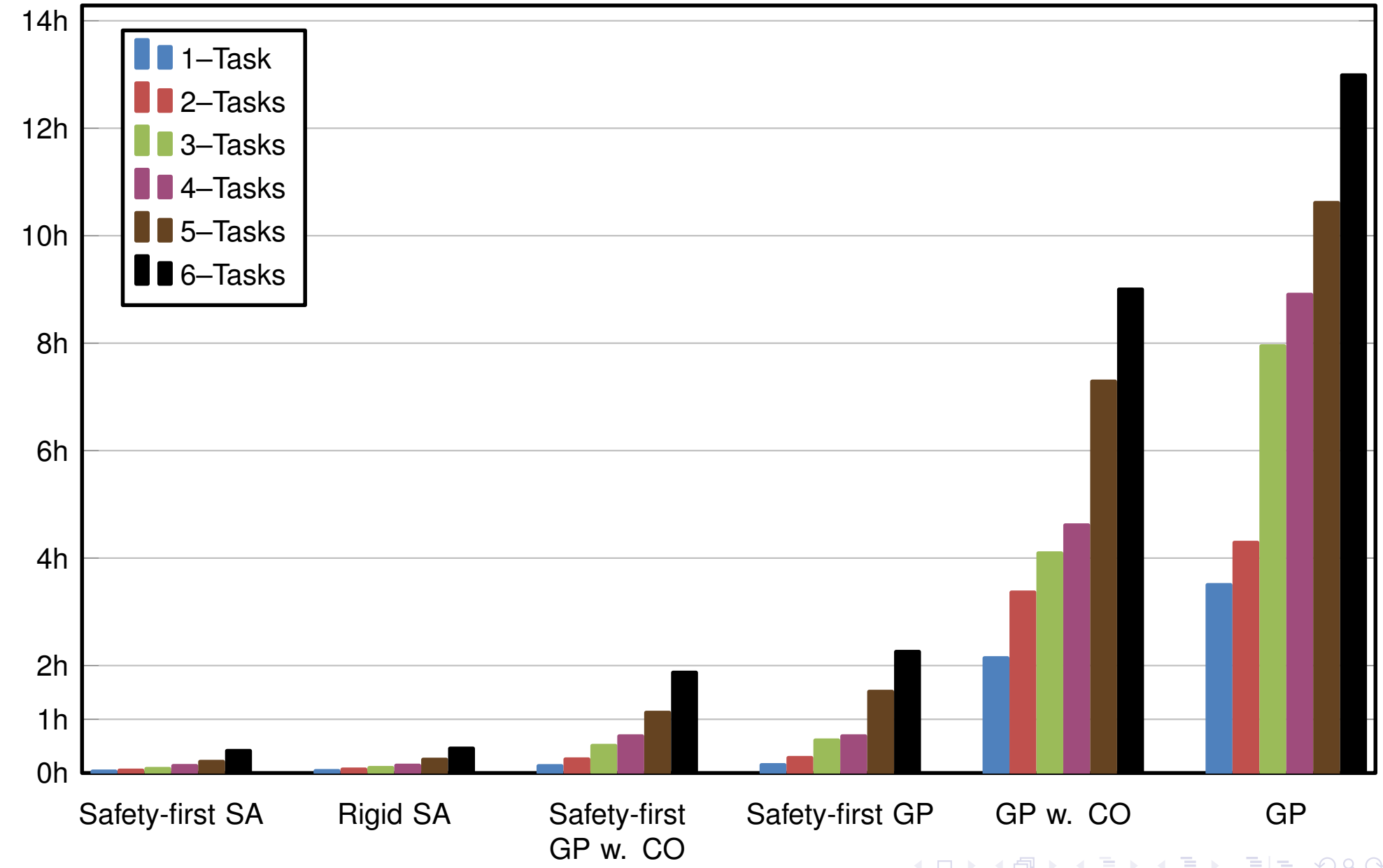
Experimental Results: Individual Execution Time



Experimental Results: Success Rate



Experimental Results: Overall Expected Runtime



Conclusion & Further Works

Summary

- Solving a Class of DCS Problems where Deterministic Strategies are sought with General Search Techniques
- Experimental Evaluation using a Scalable DCS Problem
- Relative Performances of the Algorithms match our previous results for Program Synthesis
- Runtime Performances of Classical Symbolic Techniques are several orders of magnitude better [Marchand et al., 2000, Berthier and Marchand, 2014]

Conclusion & Further Works

Summary

- Solving a Class of DCS Problems where Deterministic Strategies are sought with General Search Techniques
- Experimental Evaluation using a Scalable DCS Problem
- Relative Performances of the Algorithms match our previous results for Program Synthesis
- Runtime Performances of Classical Symbolic Techniques are several orders of magnitude better [Marchand et al., 2000, Berthier and Marchand, 2014]

Trails for Improvements & Further Investigations

- Parameter tuning: Good cooling schedule? Population size?
- Integration in a Model-checker

Thanks!

References I

N. Berthier and H. Marchand. Discrete Controller Synthesis for Infinite State Systems with ReaX. In 12th Int. Workshop on Discrete Event Systems, WODES'14, pages 46–53. IFAC, May 2014. ISBN 978-3-902823-61-8.

J. A. Clark and J. L. Jacob. Protocols are programs too: the meta-heuristic search for security protocols. Information and Software Technology, 43:891–904, 2001.

D. Henderson, S. H. Jacobson, and A. W. Johnson. The theory and practice of simulated annealing. In Handbook of metaheuristics, pages 287–319. Springer, Berlin, Heidelberg, 2003.

I. Husien and S. Schewe. Program generation using simulated annealing and model checking. In International Conference on Software Engineering and Formal Methods, pages 155–171, Berlin, Heidelberg, 2016. Springer.

References II

C. G. Johnson. Genetic programming with fitness based on model checking. In EuroGP, volume 4445 of LNCS, pages 114–124, Berlin, Heidelberg, 2007. Springer.

G. Katz and D. Peled. Model checking-based genetic programming with an application to mutual exclusion. In TACAS, volume 4963 of LNCS, pages 141–156, Berlin, Heidelberg, 2008. Springer.

G. Katz and D. Peled. Model checking driven heuristic search for correct programs. In MoChArt, volume 5348 of LNCS, pages 122–131, Berlin, Heidelberg, 2009. Springer.

G. Katz and D. Peled. Synthesizing solutions to the leader election problem using model checking and genetic programming. In Proceedings of the 5th International Haifa Verification Conference on Hardware and Software: Verification and Testing, HVC'09, pages 117–132, Berlin, Heidelberg, 2011. Springer. ISBN 978-3-642-19236-4.

References III

H. Marchand, P. Bournai, M. Le Borgne, and P. Le Guernic. Synthesis of discrete-event controllers based on the signal environment. Discrete Event Dynamic System: Theory and Applications, 10(4): 325–346, Oct. 2000.