Generalizing Backdoors

Roberto Rossi[1]
Steven D. Prestwich[1]
S. Armagan Tarim[2]
Brahim Hnich[3]

[1]Cork Constraint Computation Centre, University College Cork, Ireland
[2]Department of Management, Hacettepe University, Ankara, Turkey
[3]Faculty of Computer Science, Izmir University of Economics, Izmir, Turkey

5th International Workshop on Local Search Techniques in Constraint Satisfaction

Outline

Abstract

A powerful intuition in the design of search methods is that one wants to proactively select variables that simplify the problem instance as much as possible when these variables are assigned values. The notion of "Backdoor" variables follows this intuition.

Abstract

A powerful intuition in the design of search methods is that one wants to proactively select variables that simplify the problem instance as much as possible when these variables are assigned values. The notion of "Backdoor" variables follows this intuition.

- We generalize Backdoors in such a way to allow more general classes of sub-solvers, both complete and heuristic

Abstract

A powerful intuition in the design of search methods is that one wants to proactively select variables that simplify the problem instance as much as possible when these variables are assigned values. The notion of "Backdoor" variables follows this intuition.

- We generalize Backdoors in such a way to allow more general classes of sub-solvers, both complete and heuristic
    - Pseudo-Backdoors

Abstract

A powerful intuition in the design of search methods is that one wants to proactively select variables that simplify the problem instance as much as possible when these variables are assigned values. The notion of “Backdoor” variables follows this intuition.

- We generalize Backdoors in such a way to allow more general classes of sub-solvers, both complete and heuristic
  - Pseudo-Backdoors
  - Heuristic-Backdoors

Abstract

A powerful intuition in the design of search methods is that one wants to proactively select variables that simplify the problem instance as much as possible when these variables are assigned values. The notion of "Backdoor" variables follows this intuition.

- We generalize Backdoors in such a way to allow more general classes of sub-solvers, both complete and heuristic
  - Pseudo-Backdoors
  - Heuristic-Backdoors
- We applied these techniques to

Abstract

A powerful intuition in the design of search methods is that one wants to proactively select variables that simplify the problem instance as much as possible when these variables are assigned values. The notion of "Backdoor" variables follows this intuition.

- We generalize Backdoors in such a way to allow more general classes of sub-solvers, both complete and heuristic
  - Pseudo-Backdoors
  - Heuristic-Backdoors
- We applied these techniques to
  - A Multiple Knapsack Problem

Abstract

A powerful intuition in the design of search methods is that one wants to proactively select variables that simplify the problem instance as much as possible when these variables are assigned values. The notion of "Backdoor" variables follows this intuition.

- We generalize Backdoors in such a way to allow more general classes of sub-solvers, both complete and heuristic
  - Pseudo-Backdoors
  - Heuristic-Backdoors
- We applied these techniques to
  - A Multiple Knapsack Problem
  - An Inventory Control Problem

Constraint Satisfaction Problem

A slightly formal definition

A Constraint Satisfaction Problem is a triple $\langle V, D, C \rangle$.

Constraint Satisfaction Problem

A slightly formal definition

A Constraint Satisfaction Problem is a triple $\langle V, D, C \rangle$.

- $V = \{v_1, \ldots, v_n\}$ is a set of variables

Constraint Satisfaction Problem

A slightly formal definition

A Constraint Satisfaction Problem is a triple $\langle V, D, C\rangle$.

- $V = \{v_1, \ldots, v_n\}$ is a set of variables
- D is a function mapping each variable v_i to a domain $D(v_i)$ of values

Constraint Satisfaction Problem

A slightly formal definition

A Constraint Satisfaction Problem is a triple $\langle V, D, C \rangle$.

- $V = \{v_1, \ldots, v_n\}$ is a set of variables
- D is a function mapping each variable v_i to a domain $D(v_i)$ of values
- C is a set of constraints

Constraint Satisfaction Problem

A slightly formal definition

A Constraint Satisfaction Problem is a triple $\langle V, D, C \rangle$.

- $V = \{v_1, \ldots, v_n\}$ is a set of variables
- D is a function mapping each variable v_i to a domain $D(v_i)$ of values
- C is a set of constraints

Sample CSP

- $V = \{x, y\}$
- $D(x) = \{1, 3, 4, 5\}$ $D(y) = \{4, 5, 8\}$
- $C = \{x + 3 = y\}$

A possible solution for the CSP is $x = 1$ and $y = 4$.

Hidden Structures: Backdoors

- A powerful intuition in the design of search methods is that one wants to **proactively select variables** that simplify the problem instance as much as possible when these variables are assigned values.

Hidden Structures: Backdoors

- A powerful intuition in the design of search methods is that one wants to **proactively select variables** that simplify the problem instance as much as possible when these variables are assigned values.
- This intuition leads to the common heuristic of branching on the **most constrained variable first**.

Hidden Structures: Backdoors

- A powerful intuition in the design of search methods is that one wants to **proactively select variables** that simplify the problem instance as much as possible when these variables are assigned values.
- This intuition leads to the common heuristic of branching on the **most constrained variable first**.
- In Williams et al. [9] discuss a **formal framework** inspired by these techniques.

Hidden Structures: Backdoors

- A powerful intuition in the design of search methods is that one wants to **proactively select variables** that simplify the problem instance as much as possible when these variables are assigned values.
- This intuition leads to the common heuristic of branching on the **most constrained variable first**.
- In Williams et al. [9] discuss a **formal framework** inspired by these techniques.
- One of the main contributions in this work is the notion of "**Backdoor**" variables.

Hidden Structures: Backdoors

Backdoor

Backdoor Set: a set of variables for which there is a value assignment such that the simplified problem can be solved by a **poly-time algorithm** called the “sub-solver”

Hidden Structures: Backdoors

Backdoor

Backdoor Set: a set of variables for which there is a value assignment such that the simplified problem can be solved by a **poly-time algorithm** called the “sub-solver”

Strong Backdoor

Strong Backdoor Set: a set of variables for which **any assignment** leads to a poly-time solvable subproblem

Hidden Structures: Backdoors

Sub-solver

A sub-solver A given as input a CSP, C:

Hidden Structures: Backdoors

Sub-solver

A sub-solver A given as input a CSP, C:

- either **rejects** the input C, or “determines” C correctly (as unsatisfiable or satisfiable), returning a solution if satisfiable

Hidden Structures: Backdoors

Sub-solver

A sub-solver A given as input a CSP, C:

- either **rejects** the input C, or “determines” C correctly (as unsatisfiable or satisfiable), returning a solution if satisfiable
- runs in **polynomial** time

Hidden Structures: Backdoors

Sub-solver

A sub-solver A given as input a CSP, C:

- either **rejects** the input C, or “determines” C correctly (as unsatisfiable or satisfiable), returning a solution if satisfiable
- runs in **polynomial** time
- can determine if C is trivially true (has no constraints) or trivially false (has a contradictory constraint)

Hidden Structures: Backdoors

Sub-solver

A sub-solver A given as input a CSP, C:

- either **rejects** the input C, or "determines" C correctly (as unsatisfiable or satisfiable), returning a solution if satisfiable
- runs in **polynomial** time
- can determine if C is trivially true (has no constraints) or trivially false (has a contradictory constraint)
- if A determines C, then for any variable x and value v, then A determines the **simplified CSP** where x is assigned to v

Backdoors in practice...

An example

Backdoors can be exploited to **dynamically switch** the propagation logic and achieve a **higher level of consistency** during the search

Backdoors in practice...

An example

Backdoors can be exploited to **dynamically switch** the propagation logic and achieve a **higher level of consistency** during the search

- Let us consider the following CSP=$\langle V, C, D\rangle$:
  $V \equiv \{X_1, X_2, ..., X_m, N\}$,
  $D \equiv \{X_1, X_2, ..., X_m, N \in \{1, \ldots, m\}\}$,
  $C \equiv \{\texttt{NValue}([X_1, X_2, ..., X_m], N), N = m\}$.

Backdoors in practice...

An example

Backdoors can be exploited to **dynamically switch** the propagation logic and achieve a **higher level of consistency** during the search

- Let us consider the following CSP=$\langle V, C, D\rangle$:
  $V \equiv \{X_1, X_2, ..., X_m, N\}$,
  $D \equiv \{X_1, X_2, ..., X_m, N \in \{1, \ldots, m\}\}$,
  $C \equiv \{\mathtt{NValue}([X_1, X_2, ..., X_m], N), N = m\}$.

Propagating the `NValue` constraint is NP-hard (Bessiere et al. [2]) and thus its propagator, which we shall call P, **does not achieve** hyper-arc consistency since this would be computationally too expensive

Backdoors in practice...

An example

Backdoors can be exploited to **dynamically switch** the propagation logic and achieve a **higher level of consistency** during the search

- Let us consider the following CSP=$\langle V, C, D \rangle$:
  $V \equiv \{X_1, X_2, ..., X_m, N\}$,
  $D \equiv \{X_1, X_2, ..., X_m, N \in \{1, \ldots, m\}\}$,
  $C \equiv \{\texttt{NValue}([X_1, X_2, ..., X_m], N), N = m\}$.

Nevertheless it is clear that in the given CSP, once constraint $N = m$ is propagated, constraint $\texttt{NValue}([X_1, X_2, ..., X_m], N)$ becomes equivalent to $\texttt{allDiff}([X_1, X_2, ..., X_m])$

Backdoors in practice...

An example

Backdoors can be exploited to **dynamically switch** the propagation logic and achieve a **higher level of consistency** during the search

- Let us consider the following CSP=$\langle V, C, D \rangle$:
  $V \equiv \{X_1, X_2, ..., X_m, N\}$,
  $D \equiv \{X_1, X_2, ..., X_m, N \in \{1, \ldots, m\}\}$,
  $C \equiv \{\texttt{NValue}([X_1, X_2, ..., X_m], N), N = m\}$.

Let A be the **poly time** algorithm that achieves hyper-arc consistency for `allDiff`, then $N \rightarrow m$ is a Backdoor with respect to A

Backdoors in practice...

An example

Backdoors can be exploited to **dynamically switch** the propagation logic and achieve a **higher level of consistency** during the search

- Let us consider the following CSP=$\langle V, C, D\rangle$:
  $V \equiv \{X_1, X_2, ..., X_m, N\}$,
  $D \equiv \{X_1, X_2, ..., X_m, N \in \{1, \ldots, m\}\}$,
  $C \equiv \{\texttt{NValue}([X_1, X_2, ..., X_m], N), N = m\}$.

In this regard an interesting discussion is carried on in Bessiere et al. [1], where the **parameterized complexity** of global constraints is discussed.

Hidden Structures: Backdoors

A given sub-solver *A* must **run in polynomial time** and must **reject** (in polynomial time) the input if it is not able to either conclude satisfiability or unsatisfiability.

Hidden Structures: Backdoors

A given sub-solver A must **run in polynomial time** and must **reject** (in polynomial time) the input if it is not able to either conclude satisfiability or unsatisfiability.

Backdoor Condition

Given a CSP, C, a *Backdoor Condition* with respect to a sub-solver A is a (global) constraint P on the subset $S \subseteq V$ of the decision variables in C that are currently instantiated, such that if the partial assignment $a_S : S \subseteq V \rightarrow D$ satisfies P, then a_S is a Backdoor in C for A. Determining if a_S satisfies P must be performed in polynomial time.

Generalizing Backdoors

- Having an efficient (polynomial) algorithm for handling a subproblem that arises when some of the decision variables are fixed is indeed **desirable**

Generalizing Backdoors

- Having an efficient (polynomial) algorithm for handling a subproblem that arises when some of the decision variables are fixed is indeed **desirable**
- Nevertheless, often it may be the case that, after some decision variables have been fixed, the remaining subproblem is still NP-hard, but it has some **additional structure** that the original problem does not have

Generalizing Backdoors

- Having an efficient (polynomial) algorithm for handling a subproblem that arises when some of the decision variables are fixed is indeed **desirable**
- Nevertheless, often it may be the case that, after some decision variables have been fixed, the remaining subproblem is still NP-hard, but it has some **additional structure** that the original problem does not have
- If this is the case, it is possible that specialized algorithms, such as dedicated propagators or heuristic procedures, may be able to **exploit this additional structure** in order to either achieve a stronger filtering or quickly produce promising or optimal assignments for all or some of the remaining decision variables

Pseuso-Backdoors

Informally...

- We **relax** the assumption stating that a sub-solver *A* must run in polynomial time

Pseuso-Backdoors

Informally...

- We **relax** the assumption stating that a sub-solver *A* must run in polynomial time
- Therefore we may accept sub-solvers having an **exponential worst-case run time** required to “determine” a solution for the CSP

Pseuso-Backdoors

Informally...

- We **relax** the assumption stating that a sub-solver *A* must run in polynomial time
- Therefore we may accept sub-solvers having an **exponential worst-case run time** required to “determine” a solution for the CSP
- Nevertheless the sub-solver should still be able to **reject the input in polynomial time** if satisfiability or unsatisfiability cannot be inferred

Pseuso-Backdoors

Informally...

- We **relax** the assumption stating that a sub-solver *A* must run in polynomial time
- Therefore we may accept sub-solvers having an **exponential worst-case run time** required to “determine” a solution for the CSP
- Nevertheless the sub-solver should still be able to **reject the input in polynomial time** if satisfiability or unsatisfiability cannot be inferred
- The **key idea** then is that, although a given sub-solver is not guaranteed to produce a solution in polynomial time, it should be able to **produce competitive run times in practice**.

Pseuso-Backdoors

Formally...

Pseuso-Backdoors

Formally...

We consider a sub-solver $\widehat{A}$ that is able to **reject an input in polynomial time**, but that **may require exponential time** to “determine” a solution for the CSP or to conclude unsatisfiability.

Pseuso-Backdoors

Formally...
We consider a sub-solver $\widehat{A}$ that is able to **reject an input in polynomial time**, but that **may require exponential time** to "determine" a solution for the CSP or to conclude unsatisfiability.

Pseudo-Backdoor

A nonempty subset S of the variables is a Pseudo-Backdoor in C for $\widehat{A}$ if for some $a_s : S \rightarrow D$, $\widehat{A}$ returns a satisfying assignment of $C[a_S]$ or concludes unsatisfiability of $C[a_S]$.

Pseuso-Backdoors

Formally...
We consider a sub-solver $\widehat{A}$ that is able to **reject an input in polynomial time**, but that **may require exponential time** to "determine" a solution for the CSP or to conclude unsatisfiability.

Strong Pseudo-Backdoor

A nonempty subset S of the variables is a Strong Pseudo-Backdoors in C for $\widehat{A}$ if for all $a_s : S \rightarrow D$, $\widehat{A}$ returns a satisfying assignment or concludes unsatisfiability of $C[a_S]$.

Pseuso-Backdoors

Formally...
We consider a sub-solver $\widehat{A}$ that is able to **reject an input in polynomial time**, but that **may require exponential time** to "determine" a solution for the CSP or to conclude unsatisfiability.

Pseudo-Backdoor Condition

A Pseudo-Backdoor Condition with respect to a sub-solver $\widehat{A}$ is a (global) constraint P on the subset $S \subseteq V$ of the decision variables in C that are currently instantiated, such that if the partial assignment $a_S : S \subseteq V \to D$ satisfies P, then a_S is a Pseudo-Backdoor in C for $\widehat{A}$. Determining if a_S satisfies P must be performed in polynomial time.

Pseuso-Backdoors

An example: Multiple Knapsack

We consider a **multiple knapsack problem** with two bins into which objects can be fitted. A set of objects is given, for each object a **profit** and a **weight** are also given. Each bin is assigned a certain **capacity**. We want to fit as many objects as possible in the bins in such a way to **maximize profit** and to not exceed the capacity available for each bin.

Pseuso-Backdoors

An example: Multiple Knapsack

A simple observation directly leads to an effective **Pseudo-Backdoor Condition**. As soon as *the objects fitted in one of the two containers occupy enough capacity so that none of the remaining objects can be fitted in it*, the remaining problem is then to fit the unassigned objects to a "virtual bin" having a capacity equal to the residual capacity of the other bin.

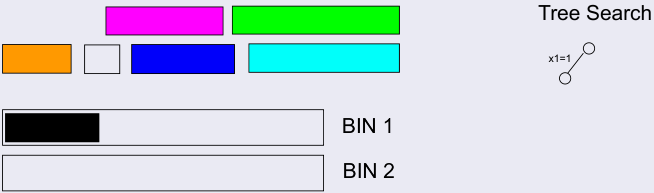
Pseuso-Backdoors

An example: Multiple Knapsack

Once a given partial assignment a_S satisfies the Pseudo-Backdoor Condition described, the remaining problem is obviously a **simple 0-1 Knapsack**.

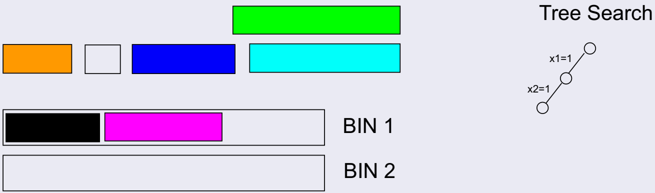
Pseuso-Backdoors

An example: Multiple Knapsack



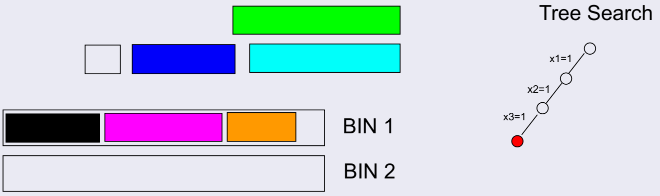
Pseuso-Backdoors

An example: Multiple Knapsack



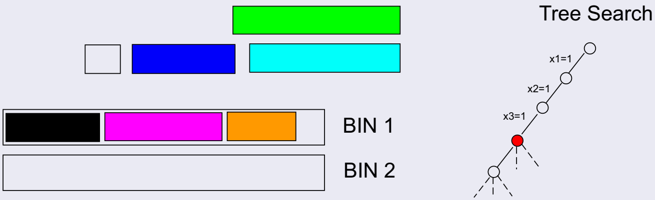
Pseuso-Backdoors

An example: Multiple Knapsack



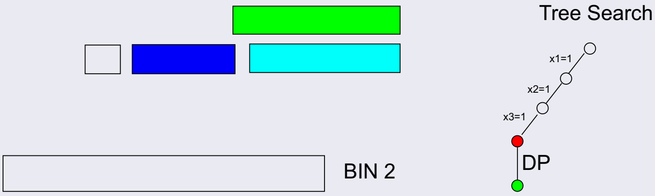
Pseuso-Backdoors

An example: Multiple Knapsack



Pseuso-Backdoors

An example: Multiple Knapsack



Pseuso-Backdoors

An example: Multiple Knapsack

Items	KP-DFS	KP-DFS-DP
10	0.02	0.03
15	0.45	0.04
20	14	0.100
25	210	0.270

Table: Multiple Knapsack Problem. Comparison between the run times (in seconds) of a pure depth-first search strategy (KP-DFS) and of the hybrid depth-first/dynamic programming search strategy based on the Pseudo-Backdoor discussed (KP-DFS-DP).

Heuristic-Backdoors

Another requirement we could relax for a given sub-solver A is **completeness**. This means that the sub-solver may adopt a **heuristic strat-
egy**.

Heuristic-Backdoors

Another requirement we could relax for a given sub-solver A is **completeness**. This means that the sub-solver may adopt a **heuristic strategy**.

In **CSPs** the former observation leads to the following approach:

- A solution method in which the sub-solver is used for **heuristically produce a feasible assignment** for some or all the remaining decision variables.

Heuristic-Backdoors

Another requirement we could relax for a given sub-solver *A* is **completeness**. This means that the sub-solver may adopt a **heuristic strategy**.

In **COPs** the former observation can lead to two different approaches:

- A complete solution method in which the heuristic sub-solver is used to **generate a near-optimal solution** that provides a **good bound** during the search. This approach is typically used in branch and bound algorithms (Lawler and Wood [7]).
- A heuristic solution method in which the heuristic sub-solver is used for **assigning "promising" values** to some or all the remaining decision variables.

Heuristic-Backdoors

More formally a **heuristic sub-solver** $\tilde{A}$ given as input a CSP, C, either

Heuristic-Backdoors

More formally a **heuristic sub-solver** $\tilde{A}$ given as input a CSP, C, either

- **rejects** the input C in polynomial time, or "**may induce**" a (partial) assignment on it

Heuristic-Backdoors

More formally a **heuristic sub-solver** $\tilde{A}$ given as input a CSP, C, either

- **rejects** the input C in polynomial time, or "**may induce**" a (partial) assignment on it
- if $\tilde{A}$ "may induce" a (partial) assignment on C, then for any variable x and value v, then $\tilde{A}$ "may induce" a (partial) assignment on the **simplified CSP** where x is assigned to v

Heuristic-Backdoors

More formally a **heuristic sub-solver** $\tilde{A}$ given as input a CSP, C, either

- **rejects** the input C in polynomial time, or "**may induce**" a (partial) assignment on it
- if $\tilde{A}$ "may induce" a (partial) assignment on C, then for any variable x and value v, then $\tilde{A}$ "may induce" a (partial) assignment on the **simplified CSP** where x is assigned to v

In order to clarify, "may induce" means that the sub-solver will actually **induce an assignment** if the heuristic strategy employed is able to produce such an assignment **within the given time/runs limit**, otherwise the sub-solver will simply **reject the input**.

Heuristic-Backdoors

Heuristic-Backdoor

A nonempty subset S of the variables is a Heuristic-Backdoor in C for $\tilde{A}$ if for some $a_s : S \rightarrow D$, $\tilde{A}$ may return a feasible assignment for $C[a_S]$.

Heuristic-Backdoors

Strong Heuristic-Backdoor

A nonempty subset S of the variables is a Strong Heuristic-Backdoor in C for $\tilde{A}$ if for all $a_s : S \rightarrow D$, A may return a feasible assignment for $C[a_S]$.

Heuristic-Backdoors

Heuristic-Backdoor Condition

Given a CSP, C, a *Heuristic-Backdoor Condition* with respect to a heuristic sub-solver $\tilde{A}$ is a (global) constraint P on the subset $S \subseteq V$ of the decision variables in C that are currently instantiated, such that if the partial assignment $a_S : S \subseteq V \to D$ satisfies P, then a_S is a Heuristic-Backdoor in C for $\tilde{A}$. Determining if a_S satisfies P must be performed in polynomial time.

Heuristic-Backdoors

Discussion

- (Strong) Heuristic-Backdoors are particularly suitable for developing **structured ways of heuristically solving complex problems**.

Heuristic-Backdoors

Discussion

- (Strong) Heuristic-Backdoors are particularly suitable for developing **structured ways of heuristically solving complex problems**.
- In what follows we will show that using this novel concept it is possible to develop **effective heuristic approaches** to complex combinatorial optimization problems by employing very simple heuristic strategies, such as Hill Climbing procedures.

Heuristic-Backdoors

Discussion

- (Strong) Heuristic-Backdoors are particularly suitable for developing **structured ways of heuristically solving complex problems**.
- In what follows we will show that using this novel concept it is possible to develop **effective heuristic approaches** to complex combinatorial optimization problems by employing very simple heuristic strategies, such as Hill Climbing procedures.
- The main reason for this is that, by using tree search, the original problem is **split into much smaller problems**. On these smaller problems simple heuristic rules such as iterative improvement often produce high quality assignments in almost no time.

Heuristic-Backdoors

An example: Multiple Knapsack

Let $\tilde{A}$ be a simple **Greedy Algorithm** for solving 0-1 Knapsack problems. In this algorithm objects are **ordered by decreasing profit over weight**. Once ordered, objects are scanned sequentially and put into the knapsack if the residual capacity allows the insertion. This can be seen as a simple **Hill Climbing strategy** in which at each step we perform an “improving” move (insertion of an object in the bin) until a local maximum is achieved (no more objects can be fit in the bin).

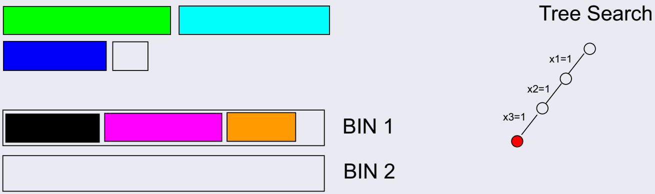
Heuristic-Backdoors

An example: Multiple Knapsack

In the former example the Pseudo-Backdoor Condition described incidentally is also a Heuristic-Backdoor Codition with respect to this Greedy algorithm $\tilde{A}$. Thus as soon as this condition is met by a given partial assignment a_S the remaining subproblem can be solved in a heuristic way by using $\tilde{A}$.

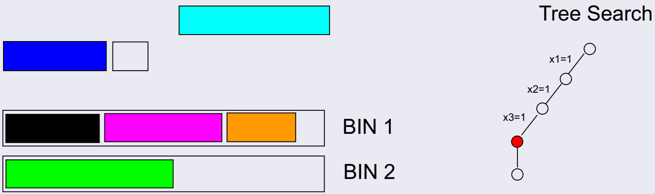
Heuristic-Backdoors

An example: Multiple Knapsack



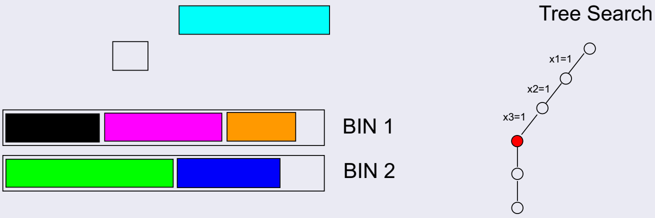
Heuristic-Backdoors

An example: Multiple Knapsack



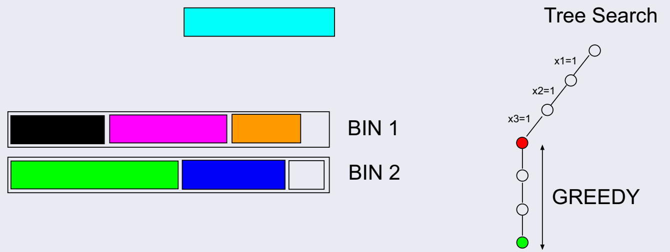
Heuristic-Backdoors

An example: Multiple Knapsack



Heuristic-Backdoors

An example: Multiple Knapsack



Heuristic-Backdoors

An example: Multiple Knapsack

Items	KP-DFS	KP-DFS-DP	KP-DFS-LS	% of real optimum
10	0.02	0.03	$<$0.001	100
15	0.45	0.04	$<$0.001	97.9
20	14	0.100	0.01	100
25	210	0.270	0.02	99.2

Table: Multiple Knapsack Problem. Comparison between the run times (in seconds) of a pure depth-first search strategy (KP-DFS), of the hybrid depth-first/dynamic programming search strategy based on the Pseudo-Backdoor discussed (KP-DFS-DP), and of the hybrid depth-first/local search strategy based on the Heuristic-Backdoor discussed (KP-DFS-LS). % of real optimum denotes the fraction (in percentage) of the optimum profit achieved by the heuristic approach.

Inventory Control

An example

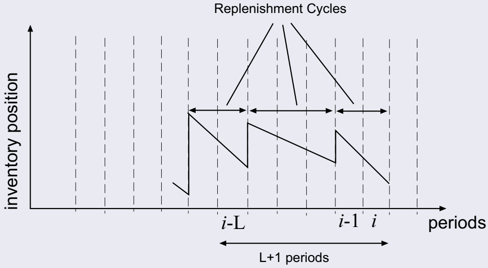


Figure: Replenishment Cycles corresponding to the following partial assignment for replenishment decisions: $\delta_{i-L-1} = 1$, $\delta_{i-L} = 0$, $\delta_{i-L+1} = 1$, $\delta_{i-L+2} = 0$, $\delta_{i-L+3} = 0$, $\delta_{i-1} = 1$, $\delta_i = 0$. Since at least L periods before period i are covered by this set of consecutive cycles it is possible to determine the service level at period i.

Related Works

- The concept of Backdoors has been **originally introduced** in Williams et al. [9].

Related Works

- The concept of Backdoors has been **originally introduced** in Williams et al. [9].
- Since then, much of the work on Backdoors has been focused on **SAT problems**, see for instance Lynce and Silva[8].

Related Works

- The concept of Backdoors has been **originally introduced** in Williams et al. [9].
- Since then, much of the work on Backdoors has been focused on **SAT problems**, see for instance Lynce and Silva[8].
- In Cambazard et al. [3] the authors propose an explanation-based approach exploiting Backdoors for dynamically **identifying and exploiting** structures in CSPs.

Related Works

- The concept of Backdoors has been **originally introduced** in Williams et al. [9].
- Since then, much of the work on Backdoors has been focused on **SAT problems**, see for instance Lynce and Silva[8].
- In Cambazard et al. [3] the authors propose an explanation-based approach exploiting Backdoors for dynamically **identifying and exploiting** structures in CSPs.
- Nevertheless, to the best of our knowledge, in the literature Bakdoors **have not been used** so far for **switching the search strategy** either to a complete or incomplete different strategy **not necessarily polynomial** (such as Dynamic Programming).

Related Works

- The **integration of Operations Research and Constraint Programming** techniques for combinatorial optimization is a very active research field (Focacci et al. [6])

Related Works

- The **integration of Operations Research and Constraint Programming** techniques for combinatorial optimization is a very active research field (Focacci et al. [6])
- Nevertheless, operations research techniques are typically employed for **generating valid relaxations** used for performing domain filtering and, with the exception of Bender's Decomposition in Cambazard et al. [3], they are **not employed as alternative search strategies** that can take over the control of the search process when a given condition is met.

Related Works

The **integration between Constraint Programming and Local Search** has been discussed in a variety of works (a review by Focacci et al. [5]):

Related Works

The **integration between Constraint Programming and Local Search** has been discussed in a variety of works (a review by Focacci et al. [5]):

- Local search engine is used to “**guide**” the search, while Constraint Programming is used for exploring promising neighborhood.

Related Works

The **integration between Constraint Programming and Local Search** has been discussed in a variety of works (a review by Focacci et al. [5]):

- Local search engine is used to “**guide**” the search, while Constraint Programming is used for exploring promising neighborhood.
- Alternatively, local search techniques can be introduced within a **constructive** global search algorithm (Cesta et al. [4]).

Related Works

The **integration between Constraint Programming and Local Search** has been discussed in a variety of works (a review by Focacci et al. [5]):

- Local search engine is used to “**guide**” the search, while Constraint Programming is used for exploring promising neighborhood.
- Alternatively, local search techniques can be introduced within a **constructive** global search algorithm (Cesta et al. [4]).

The technique we propose is of this **second** kind, but the notion of Heuristic-Backdoor makes our approach novel and more general compared to other specialized approaches presented in the literature.

Conclusions

Conclusions

- We **generalized Backdoors** in such a way to allow sub-solvers that **do not run** in polynomial time.

Conclusions

- We **generalized Backdoors** in such a way to allow sub-solvers that **do not run** in polynomial time.
- This led to **Pseudo-Backdoors** and to **Heuristic-Backdoors**, that let us switch the search logic (or the propagation logic of a given global constraint) as soon as a **known structure** in the remaining subproblem that has to be solved **is revealed** by a given partial assignment.

Conclusions

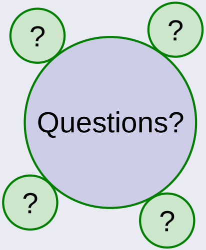
- We **generalized Backdoors** in such a way to allow sub-solvers that **do not run** in polynomial time.
- This led to **Pseudo-Backdoors** and to **Heuristic-Backdoors**, that let us switch the search logic (or the propagation logic of a given global constraint) as soon as a **known structure** in the remaining subproblem that has to be solved **is revealed** by a given partial assignment.
- We applied both Pseudo-Backdoors and Heuristic-Backdoors to a simple **Multiple Knapsack Problem** taken as running example.

Conclusions

- We **generalized Backdoors** in such a way to allow sub-solvers that **do not run** in polynomial time.
- This led to **Pseudo-Backdoors** and to **Heuristic-Backdoors**, that let us switch the search logic (or the propagation logic of a given global constraint) as soon as a **known structure** in the remaining subproblem that has to be solved **is revealed** by a given partial assignment.
- We applied both Pseudo-Backdoors and Heuristic-Backdoors to a simple **Multiple Knapsack Problem** taken as running example.
- We have also discussed the effectiveness of Heuristic-Backdoors on a complex combinatorial optimization problem.

The End

The End



Acknowledgments: this work was supported by Science Foundation Ireland under Grant No. 03/CE3/I405 as part of the Centre for Telecommunications Value-Chain Research (CTVR) and Grant No. 05/IN/I886. S. Armagan Tarim and Brahim Hnich are supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant No. SOBAG-108K027

C. Bessiere, E. Hebrard, B. Hnich, Z. Kiziltan, C.-G. Quimper, and T. Walsh.
The parameterized complexity of global constraints.
In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, 2008.

C. Bessiere, E. Hebrard, B. Hnich, Z. Kiziltan, and T. Walsh.
Filtering algorithms for the nvalue constraint.
Constraints, 11(4):271–293, 2006.

H. Cambazard and N. Jussien.
Identifying and exploiting problem structures using explanation-based constraint programming.
Constraints, 11(4):295–313, 2006.

A. Cesta, G. Cortellessa, A. Oddi, N. Policella, and A. Susi.
A constraint-based architecture for flexible support to activity scheduling.

Lecture Notes in Computer Science, 2175:369+, 2001.

F. Focacci, F. Laburthe, and A. Lodi.
Local Search and Constraint Programming.
In F. Glover and G. Kochenberger, editors, Handbook of Metaheuristics, volume 57 of International Series in Operations Research and Management Science. Kluwer Academic Publishers, Norwell, MA, 2002.

F. Focacci and M. Milano.
Connections and integrations of dynamic programming and constraint programming.
In *Proceedings of the International Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimization Problems CP-AI-OR 2001*, 2001.

E. L. Lawler and D. E. Wood.
Branch-and-bound methods: A survey.

Operations Research, 14(4):699–719, 1966.

I. Lynce and J. Marques-Silva.
Hidden structure in unsatisfiable random 3-sat: An empirical study.
In *ICTAI '04: Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*, pages 246–251, Washington, DC, USA, 2004. IEEE Computer Society.

R. Williams, C. P. Gomes, and B. Selman.
Backdoors to typical case complexity.
In Georg Gottlob and Toby Walsh, editors, *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 1173–1178. Morgan Kaufmann, 2003.