Generalizing Backdoors

Roberto Rossi
Steven D. Prestwich
S. Armagan Tarim
Brahim Hnich

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
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.
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
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
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
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
A slightly formal definition

A Constraint Satisfaction Problem is a triple $\langle V, D, C \rangle$.
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
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
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
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 $y = 4$. 
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.
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**.
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.
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”
<table>
<thead>
<tr>
<th><strong>Backdoor</strong></th>
<th><strong>Backdoor Set</strong>: 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”</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Strong Backdoor</strong></td>
<td><strong>Strong Backdoor Set</strong>: a set of variables for which any assignment leads to a poly-time solvable subproblem</td>
</tr>
</tbody>
</table>
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
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)
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 $v$
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=$\langle V, C, D \rangle$:

$V \equiv \{X_1, X_2, \ldots, X_m, N\}$,

$D \equiv \{X_1, X_2, \ldots, X_m, N \in \{1, \ldots, m\}\}$,

$C \equiv \{NValue([X_1, X_2, \ldots, X_m], N), N = m\}$. 
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=$\langle V, C, D \rangle$:

\[
V \equiv \{ X_1, X_2, \ldots, X_m, N \},
\]
\[
D \equiv \{ X_1, X_2, \ldots, X_m, N \in \{1, \ldots, m\} \},
\]
\[
C \equiv \{ NValue([X_1, X_2, \ldots, X_m], N), N = m \}.
\]

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=\( \langle V, C, D \rangle \):

\[
V \equiv \{ X_1, X_2, \ldots, X_m, N \},
\]

\[
D \equiv \{ X_1, X_2, \ldots, X_m, N \in \{1, \ldots, m\} \},
\]

\[
C \equiv \{ N\text{Value}([X_1, X_2, \ldots, X_m], N), N = m \}.
\]

Nevertheless it is clear that in the given CSP, once constraint \( N = m \) is propagated, constraint \( N\text{Value}([X_1, X_2, \ldots, X_m], N) \) becomes equivalent to \( \text{allDiff}([X_1, X_2, \ldots, X_m]) \).
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 = \( \langle V, C, D \rangle \):

- \( V \equiv \{ X_1, X_2, \ldots, X_m, N \} \),
- \( D \equiv \{ X_1, X_2, \ldots, X_m, N \in \{ 1, \ldots, m \} \} \),
- \( C \equiv \{ N\text{Value}([X_1, X_2, \ldots, X_m], N), N = m \} \).

Let \( A \) be the **poly time** algorithm that achieves hyper-arc consistency for allDiff, then \( N \to m \) 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:\(\langle V, C, D \rangle:\)

\[
V \equiv \{X_1, X_2, \ldots, X_m, N\},
\]

\[
D \equiv \{X_1, X_2, \ldots, X_m, N \in \{1, \ldots, m\}\},
\]

\[
C \equiv \{N\text{Value}([X_1, X_2, \ldots, X_m], N), N = m\}.
\]

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.
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
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.
Informally...

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

Formally...
Formally...
We consider a sub-solver \( \hat{A} \) 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.
Formally...
We consider a sub-solver $\hat{A}$ 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 $\hat{A}$ if for some $a_S : S \rightarrow D$, $\hat{A}$ returns a satisfying assignment of $C[a_S]$ or concludes unsatisfiability of $C[a_S]$. 
Pseudo-Backdoors

Formally...
We consider a sub-solver $\hat{A}$ 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 $\hat{A}$ if for all $a_S : S \rightarrow D$, $\hat{A}$ returns a satisfying assignment or concludes unsatisfiability of $C[a_S]$. 
Pseudo-Backdoors

Formally...
We consider a sub-solver $\hat{A}$ 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 $\hat{A}$ 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 Pseudo-Backdoor in $C$ for $\hat{A}$. Determining if $a_S$ satisfies $P$ must be performed in polynomial time.
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.
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.
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**.
Pseudo-Backdoors

An example: Multiple Knapsack

[Diagram showing a tree search with two bins labeled BIN 1 and BIN 2]
Pseudo-Backdoors

An example: Multiple Knapsack

Tree Search

x1=1
x2=1
Pseudo-Backdoors

An example: Multiple Knapsack

Tree Search

BIN 1

BIN 2
An example: Multiple Knapsack
Pseudo-Backdoors

An example: Multiple Knapsack

Tree Search

BIN 2
Pseudo-Backdoors

**An example: Multiple Knapsack**

<table>
<thead>
<tr>
<th>Items</th>
<th>KP-DFS</th>
<th>KP-DFS-DP</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.02</td>
<td>0.03</td>
</tr>
<tr>
<td>15</td>
<td>0.45</td>
<td>0.04</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>0.100</td>
</tr>
<tr>
<td>25</td>
<td>210</td>
<td>0.270</td>
</tr>
</tbody>
</table>

**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).
Another requirement we could relax for a given sub-solver $A$ is \textbf{completeness}. This means that the sub-solver may adopt a \textbf{heuristic strategy}. 
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
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
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 $v$
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.
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 $C[a_S]$. 
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-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 $a_S : S \subseteq V \rightarrow D$ 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.
(Strong) Heuristic-Backdoors are particularly suitable for developing **structured ways of heuristically solving complex problems.**
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.
(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.
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).
In the former example the Pseudo-Backdoor Condition described incidentally is also a Heuristic-Backdoor Condition 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 $\tilde{A}$. 
An example: Multiple Knapsack

BIN 1

BIN 2
Heuristic-Backdoors

An example: Multiple Knapsack

BIN 1

BIN 2

Tree Search

x₁ = 1

x₂ = 1

x₃ = 1
Heuristic-Backdoors

An example: Multiple Knapsack

BIN 1

BIN 2

Tree Search

x1=1

x2=1

x3=1
Heuristic-Backdoors

An example: Multiple Knapsack

Tree Search

Greedy
Heuristic-Backdoors

An example: Multiple Knapsack

<table>
<thead>
<tr>
<th>Items</th>
<th>KP-DFS</th>
<th>KP-DFS-DP</th>
<th>KP-DFS-LS</th>
<th>% of real optimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.02</td>
<td>0.03</td>
<td>&lt;0.001</td>
<td>100</td>
</tr>
<tr>
<td>15</td>
<td>0.45</td>
<td>0.04</td>
<td>&lt;0.001</td>
<td>97.9</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>0.100</td>
<td>0.01</td>
<td>100</td>
</tr>
<tr>
<td>25</td>
<td>210</td>
<td>0.270</td>
<td>0.02</td>
<td>99.2</td>
</tr>
</tbody>
</table>

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.
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 \( i \).
The concept of Backdoors has been originally introduced in Williams et al. [9].
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].
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.
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 Backdoors 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).
The integration of Operations Research and Constraint Programming techniques for combinatorial optimization is a very active research field (Focacci et al. [6])
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.
The integration between Constraint Programming and Local Search has been discussed in a variety of works (a review by Focacci et al. [5]):
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.
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

- We generalized Backdoors in such a way to allow sub-solvers that do not run in polynomial time.
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 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 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.
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/1886. S. Armagan Tarim and Brahim Hnich are supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant No. SOBAG-108K027
The parameterized complexity of global constraints.  

Filtering algorithms for the nvalue constraint.  

H. Cambazard and N. Jussien.  

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

F. Focacci, F. Laburthe, and A. Lodi.
Local Search and Constraint Programming.

F. Focacci and M. Milano.
Connections and integrations of dynamic programming and constraint programming.

E. L. Lawler and D. E. Wood.

I. Lynce and J. Marques-Silva.
Hidden structure in unsatisfiable random 3-sat: An empirical study.

R. Williams, C. P. Gomes, and B. Selman.
Backdoors to typical case complexity.