15-381 Spring 2007
Assignment 2: Constraint Satisfaction Problems SOLUTIONS

For questions contact Arthur (atu@andrew.cmu.edu)
and Gil (egjones+@cs.cmu.edu)

Spring 2007
Out: Feb. 6
Due: Feb. 20, 1:30pm Tuesday

The written portion of this assignment must be turned in at the beginning of class at 1:30pm on February 20th. Type or write neatly; illegible submissions will not receive credit. Write your name and andrew id clearly at the top of your assignment. If you do not write your andrew id on your assignment, you will lose 5 points.

The code portion of this assignment must be submitted electronically by 1:30pm on February 20th. To submit your code, please copy all of the necessary files to the following directory:

/afs/andrew.cmu.edu/course/15/381/hw2_submit_directory/yourandrewid

replacing yourandrewid with your Andrew ID. The TAs grading this assignment would prefer that you use C/C++, Java, or Matlab. If you would like to use a different programming language, please give instructions on how to execute your code and comment it very thoroughly or you risk decreased partial credit. Feel free to use standard data structures and algorithms in your chosen language (like the C++ STL). If you use non-standard data structures or algorithms (like someone's internet C queue code) make sure you turn that in as well and indicate in the comments if you did not write a particular piece of code. All code will be tested on a Linux system, we will not accept Windows binaries. No matter what language you use, you must ensure that the code compiles and runs in the afs submission directory. Clearly document your program.

Late Policy. Both your written work and code are due at 1:30pm on 2/20. Submitting your work late will affect its score as follows:

- If you submit it after 1:30pm on 2/20 but before 1:30pm on 2/21, it will receive 90% of its score.
- If you submit it after 1:30pm on 2/21 but before 1:30pm on 2/22, it will receive 50% of its score.
- If you submit it after 1:30pm on 2/22, it will receive no score.

Collaboration Policy.

You are to complete this assignment individually. However, you are encouraged to discuss the general algorithms and ideas in the class in order to help each other answer homework questions. You are also welcome to give each other examples that are not on the assignment in order to demonstrate how to solve problems. But we require you to:

- not explicitly tell each other the answers

- not to copy answers
- not to allow your answers to be copied

In those cases where you work with one or more other people on the general discussion of the assignment and surrounding topics, we ask that you specifically record on the assigment the names of the people you were in discussion with (or "none" if you did not talk with anyone else). This is worth five points: for each problem, you solution should either contained the names of people you talked to about it, or "none." If you do not give references for each problem, you will lose five points. This will help resolve the situtaion where a mistake in general discussion led to a replicated weird error among multiple solutions. This policy has been established in order to be fair to everyone in the class. We have a grading policy of watching for cheating and we will follow up if it is detected.

Problem 1 (20 total points)

Alby-Bach University (ABU) wants to start a new degree program: B.S in Judgment Day Prevention (JDP).

Suppose the degree program is associated with the following courses:

15-211	Fundamental Data Structures and Algorithms
15-212	Principles of Programming
15-381	Artificial Intelligence: Representation and Problem-Solving
15-681	Machine Learning
80-310	Logic and Computation
21-484	Graph Theory
70-122	Accounting
70-311	Organizational Behavior
19-601	Information Warfare

In order to graduate from the degree program, one must complete the following four requirements:

Algorithms Requirement: (**15-211** AND **15-212**) OR (**15-211** AND **15-381**) OR (**15-681** AND **21-484**)

Machine Learning Requirement: 15-381 OR **15-681** OR **80-310**

Communications Requirement: 21-484 OR **70-311** OR **70-122**

Information Warfare Requirement: 15-381 OR **19-601**

In addition, the department imposes the following restrictions:

Information Aggressiveness Restriction: So that they can't make their programs TOO smart, students can take only one class from the set **15-381**, **15-681**, and **19-601**.

Basic Arithmetic Restriction: Students can't take both **15-211** and **70-122**.

Organization Restriction: Students can't take both **21-484** and **70-311**.

Finally, courses cannot be used to count towards multiple graduation requirements - so if you use 15-381 to fulfill part of the Algorithms requirement it can't count towards either the Machine Learning Requirement or the Information Warfare Requirement.

Question 1.1 (5 points)

John Conner just started his junior year at ABU, and needs to graduate as soon as possible. Suppose all he has left to take are JDP required classes. Model the problem of his trying to find a set of classes to satisfy all requirements as a CSP (Hint: the requirements should be your variables). What are the initial domains for each of your variables?

Solution 1.1

5 variables - AR_1,AR_2,MLR,CR,IWR

4 constraints -

1. IAR says ≤ 1 of 15-381, 15-681, and 19-601 can be assigned to the 5 variables.

2. BAR says ≤ 1 of 15-211 and 70-122 can be assigned to the 5 variables
3. OR says ≤ 1 of 21-484 and 70-311 can be assigned to the 5 variables
4. No double counting says if a variable is assigned to one variable it can't be assigned to another variable

Initial domains:

AR_1	15-211, 15-212, 15-381, 15-681, 21-484
AR_2	15-211, 15-212, 15-381, 15-681, 21-484
MLR	15-381, 15-681, 80-310
CR	21-484, 70-122. 70-311
IWR	15-381, 19-601

Question 1.2 (8 points)

Show a DFS with backtracking tree for finding a set of classes that fulfill all requirements using a variable order of the requirements in the order they are listed above, and using a value order that selects the lowest department/course number remaining in a variable's domain. Indicate which constraints were violated whenever the DFS needs to backtrack. (Note: to get full credit you must show the full DFS tree and not just the classes that are used to fulfill each requirement).

Solution 1.2

See Figure 1.

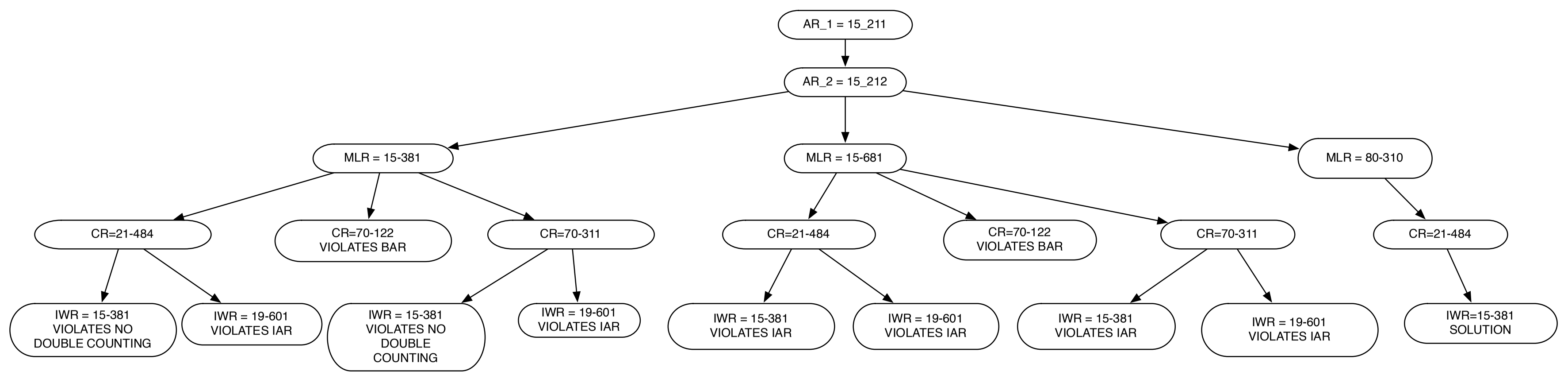
AR_1 = 15_211
AR_2 = 15_212
MLR = 15-381
MLR = 15-681
MLR = 80-310
CR=21-484
CR=70-122 VIOLATES BAR
CR=70-311
CR=21-484
CR=70-122 VIOLATES BAR
CR=70-311
CR=21-484
IWR = 15-381 VIOLATES NO DOUBLE COUNTING
IWR = 19-601 VIOLATES IAR
IWR = 15-381 VIOLATES NO DOUBLE COUNTING
IWR = 19-601 VIOLATES IAR
IWR = 15-381 VIOLATES IAR
IWR = 19-601 VIOLATES IAR
IWR = 15-381 VIOLATES IAR
IWR = 19-601 VIOLATES IAR
IWR=15-381 SOLUTION

Figure 1: DFS tree with backtracking for problem 1.2

Question 1.3 (7 points)

Suppose John has already taken **19-601** towards his Information Warfare Requirement and **15-211** towards his Algorithms Requirement. Use constraint propagation to determine other classes he must take to graduate - indicate which requirements the classes fulfill. Can you create a schedule that satisfies all constraints without using search?

Solution 1.3

AR_1 has been set to 15-211 and IWR has been set to 19-601. Forward checking can then be used as follows:

1. AR_2 domain goes from 15-212,15-381,15-681,21-484 to 15-212 due to IAR.
2. MLR domain goes to 80-310 due to IAR.
3. CR goes to 21-484,70-311 due to BAR.

Constraint propogation can then

1. MLR domain has a single member, so we select 80-310.
2. AR_2 also has a single member, so we can select 15-212.

So we have a schedule which satisfies AR_1, AR_2, MLR, and CR. But to satisfy the CR we have to select between 21-484 and 70-311 (we can't pick both). So technically we would need to do some search here to pick between them.

Problem 2 (20 total points)

Arthur is looking for a group of friends for his start-up, which develops and provides some web-based p2p downloading solutions to college students (this is before the lawsuits). Arthur has determined that he needs 2 C# Programmers, 2 Flash Designers, 1 Photoshop Guru, 1 Database Admin, and 1 Systems Engineer.

Assume that if a person knows two languages/softwares, he or she can take on two roles in the company.

So Arthurs narrowed down his selections to the following people:

Name	Abilities
Peter	C# and Flash
John	Photoshop and Flash
Jim	Flash and Systems
Jane	C# and Database
Mary	Photoshop and Flash
Bruce	Systems and C#
Chuck	Photoshop and Flash

Question 2.1 (5 points)

Suppose Arthur knows C#, and only has funds to hire three more people.

Model this scenario as a CSP - (using variables, value domains, and constraints).

Solution 2.1

There are a number of ways to model the domain. Variables can either be the people, the jobs, or the skills.

For the job-based variables, there are three variables, J_1, J_2 and J_3 (with J_4 already filled) representing the three openings. The domains of all three variables are initially all the people. We can then pose constraints in terms of the skills of people who fill those roles:

1. Number(J,C#) ≥ 2
2. Number(J,Flash) ≥ 2
3. Number(J,Photoshop) ≥ 1
4. Number(J,Database) ≥ 1
5. Number(J,Systems) ≥ 1

There's also a constraint that assigning someone to one job means that the person can't be assigned to another job opening.

We can also think of the skill spots as variables:

C#_1, C#_2, Flash_1, Flash_2, Photoshop, Database, and Systems.

For this, we would initial domains of

C#_1	Arthur
C#_2	Peter, Jane, Bruce
Flash_1	Peter, John, Mary, Chuck
Flash_2	Peter, John, Mary, Chuck
Photoshop	John, Jim, Mary, Chuck
Database	Jane
Systems	Jim, Bruce

We would additionally need constraints that say that one person couldn't fill both assignments of C#, and that we could only hire 3 additional people.

Anything that makes an argument for which things are variables and that properly represent the constraints (must hire at least to fill the requirements and can't hire more than three people) should get credit.

Question 2.2 (6 points)

Suppose Arthur decides to make Jim a co-founder. Arthur and Jim discover that all the developers absolutely refuse to abandon their favorite platforms, and that they can only afford two single-booted workstations.

Name	Abilities	OS
Arthur	C#	Windows
Peter	C# and Flash	Windows
John	Photoshop and Flash	Windows
Jim	Flash and Systems	FreeBSD
Jane	C# and Database	FreeBSD
Mary	Photoshop and Flash	Linux
Bruce	Systems and C#	Linux
Chuck	Photoshop and Flash	Windows

What are the domains for the two remaining positions after constraint propogation?

Solution 2.2

We'll model this using the skills formulation. We've already hired Jim and Arthur, so that means that we can't hire either Mary or Bruce, which leaves:

C#_1	Arthur
C#_2	Peter, Jane
Flash_1	Jim
Flash_2	Peter, John, Chuck
Photoshop	John, Chuck
Database	Jane
Systems	Jim

We have to hire Jane to satisfy the Database requirement, and then there is only one position left to fill, so we can't hire Peter. So we must hire either John or Chuck to satisfy both Flash and Photoshop.

This leaves

C#_1	Arthur
C#_2	Jane
Flash_1	Jim
Flash_2	John, Chuck
Photoshop	John, Chuck
Database	Jane
Systems	Jim

Question 2.3 (9 points)

Assume Arthur and Jim hired Bruce and Mary and they have awarded a contract for their first project, a rush job due Friday at 5 PM. It's Monday at 9 AM and they have to put in 50 total hours of work on it by the due date. There are a number of constraints associated with their work schedules:

- They only have access to two machines of the requisite platform, and only have access to those two machines between 9 AM and 5 PM every day.
- Each person can work a maximum of 20 hours over the course of the week.
- A work session by a single person on a particular machine can last no fewer than two hours.
- Arthur cannot work from 12-4 PM Tu/Th.
- Jim can't work MWF 9-12.

- Bruce can only work between noon and 2 PM every day.
- Mary can only work Thursday and Friday.

Model this scheduling problem as a CSP. Indicate how the indicated constraints impact the domains for all variables.

Solution 2.3

I will make the variables the hours of the day (as there are no constraints on half hour periods), where each day has values 1-8 for each of the two computers. This gives a variable domain T of

$$\begin{aligned} T = & M_1^1 - M_8^1 \\ & M_1^2 - M_8^2 \\ & TU_1^1 - TU_8^1 \\ & TU_1^2 - TU_8^2 \\ & W_1^1 - W_8^1 \\ & W_1^2 - W_8^2 \\ & TH_1^1 - TH_8^1 \\ & TH_1^2 - TH_8^2 \\ & F_1^1 - F_8^1 \\ & F_1^2 - F_8^2 \end{aligned}$$

We can assign these hours one of 5 variables: A (Arthur), Jim(J), Bruce(B), Mary(M) or N(No one).

First we have the total hours for the project requirement, where total function counts over all variables in T: Total(T,N) $\geq$ 50.

Then we have constraints associated with the hour limits for each person:

- Total(T,A) $\leq$ 20
- Total(T,J) $\leq$ 20
- Total(T,B) $\leq$ 20
- Total(T,M) $\leq$ 20

Now we have the sessions are at least two hours on an individual constraint:

$$\text{Assigning a given hour variable } (X_i^k = y) \text{ implies } (X_{i+1}^k = y) \text{ or } (X_{i-1}^k)$$

Then we have individual people's constraints:

Arthur's constraint: $(TU_4^k - TU_6^k) \wedge (TH_4^k - TH_4^k) \neq A$ for $k = 1, 2$.

Jim's constraint: $(M_1^k - M_3^k) \wedge (W_1^k - W_3^k) \wedge (F_1^k - F_3^k) \neq J$ for $k = 1, 2$.

Bruce's constraint:$(X_1^k - X_3^k) \wedge (X_6^k - X_8^k) \neq B$ for $X = M, TU, W, TH, F$ and $k = 1, 2$.

Mary's constraint:$(M_x^k) \wedge (TU_x^k) \wedge (W_x^k) \neq M$ for $X = 1, 2, \ldots, 8$ and $k = 1, 2$.

Problem 3 (50 total points)

A robot is tasked with delivering packages to various locations by certain deadlines associated with package delivery. The robot begins at a given position, and receives a list of package locations and their associated deadlines. The robot must determine an order to visit the locations to deliver each package before its associated deadline.

Assume that time begins at 0, that the robot can move in straight lines between delivery locations (thus a Euclidean distance metric is appropriate), that the robot can deliver packages instantaneously upon arriving at the delivery location, and that the robot can move 1 distance unit per cycle.

For this problem you'll be using depth-first search with backtracking to determine robot schedules that satisfy the package deadline constraints.

There are five scenario files associated with this exercise (scenario1.txt, scenario2.txt, etc..).

The format of each scenario file is

```
<robot initial x location> <robot initial y location>
<number of delivery locations>
<delivery x location> <delivery y location> <package deadline>
additional entries for each package
```

All values are floats except the number of delivery locations, which is an int.

Question 3.1 (30 points)

- (5 points) Did you submit your code?
- (10 points) Did your executables find the right path on scenario7.txt?
- (5 points) Did you give the 10 partial path expansions correctly?
- (5 points) Did you count #expansions correctly?
- (5 points) Did you give the right paths?

For this question you will implement uninformed depth first search with backtracking. Your search tree should be rooted at the initial robot location, and each level of the search tree should correspond to adding an additional delivery location to the path the robot takes from the start location. You must keep track of when the robot arrives at each delivery location and compare it to the location's deadline - your DFS should backtrack if the path violates a deadline constraint.

The successor function you should use for this question should use the order that the delivery locations are listed in the scenario file. In other words, if you label the delivery locations in order starting at 1, your successor function should select the lowest numbered city not already included in the tour. For this exercise make sure your successor function is aware that it would be counterproductive to visit a location in the path more than once.

In addition to submitting your code, please submit an executable called 'dfsb' that takes a single argument specifying the scenario file, and outputs the full ordered path if one is found, the arrival times at each city in the path, and the number of evaluations performed in generating the path. Additionally, in your writeup include the following:

1. The first 10 partial paths generated in your search (i.e. "My search first considered the path $S \rightarrow 1$, then $S \rightarrow 1 \rightarrow 2$, then $S \rightarrow 1 \rightarrow 2 \rightarrow 3$ violated the deadline of location 3, so it then considered $S \rightarrow 1 \rightarrow 2 \rightarrow 4$". You don't need the text, just the sequences) for `scenario1.txt`. Label the delivery locations starting with 1.
2. For each scenario, report whether or not a satisfying solution was found. For `scenario1.txt`, give the ordered path and the arrival time at each location in the path. For the other scenarios if a solution was found just give the ordered path.
3. For each scenario report the number of evaluations performed in the search. An evaluation is performed any time the constraints associated with a partial path are checked for satisfaction.
4. If any scenario is taking more than 500,000 evaluations you can stop your program and indicate that 500,000 evaluations were exceeded in lieu of an answer.

Solution 3.1.1 (First 10 partial evaluations)

```
S->1
S->1->2
S->1->2->3
S->1->2->3->4
S->1->2->3->5
S->1->2->3->6
S->1->2->4
S->1->2->5
S->1->2->5->3
S->1->2->5->4
```

Solution 3.1.2 (Ordered path with arrival times for `scenario1.txt`)

```
Starts at (24.0451,1.63808)
Arrives at city 4 loc (28.571,1.91245) at time 4.53421 for deadline 41.5744
Arrives at city 6 loc (27.8529,4.70066) at time 7.41341 for deadline 66.5376
Arrives at city 5 loc (2.27709,15.517) at time 35.1824 for deadline 77.6843
Arrives at city 2 loc (6.36816,19.3715) at time 40.8032 for deadline 43.7957
Arrives at city 3 loc (20.9536,25.5436) at time 56.6408 for deadline 60.1556
Arrives at city 1 loc (28.6092,20.4382) at time 65.8426 for deadline 77.3159
```

Solution 3.1.3 (Paths, number of evaluations)

Scenario	Path?	Path	Evaluations (± 1)
1	Yes	S 4 6 5 2 3 1	305
2	Yes	S 7 8 9 10 4 2 5 1 3 6	157216
3	Yes	S 4 8 7 6 5 1 2 3	2995
4	No		325656
5	Yes	S 1 2 3 4 6 7 10 16 13 15 5 9 8 11 12 14	608395

Question 3.2 (5 points)

Propose a variable ordering strategy for the successor function for DFS with backtracking. Explain why you think your ordering strategy is reasonable for this domain.

Solution 3.2

Two good ones are earliest deadline and minimum distance from last point in the sequence. Visting cities with early deadlines early should help us get to all cities by their deadlines, and looking at the near cities should find short overall tours, which should help arrive at cities before their deadlines.

Question 3.3 (15 points)

- (2 points) Did you submit your code?
- (4 points) Did your executables act correctly on scenario7.txt?
- (2 points) Did you submit the right partial paths
- (2 points) Did you get the #expansions correct
- (2 points) Did you get the right solutions?
- (3 points) Did you interpret your expansion results correctly?

Implement your chosen variable ordering method in your DFS with backtracking code from question 3.1. This should only require changing your successor function. Report the same values requested in section 3.1 (first 10 partial paths evaluated and satisfying solutions with arrival times for `scenario1.tex`, ordered paths for scenarios 2-5, number of evaluations for each scenario).

In addition to submitting your code, please submit an executable `dfsbvo` that takes a single argument specifying the scenario file, and outputs the full ordered path if one is found, the arrival times at each city in the path, and the number of evaluations performed in generating the path.

Write a paragraph comparing the performance with and without variable ordering. Will your variable ordering method always perform better than uninformed search in terms of evaluations?

Solution 3.3.1 - Earliest Deadline

(First 10 partial evaluations)

```
S->4
S->4->2
S->4->2->3
S->4->2->3->6
S->4->2->3->1
S->4->2->3->1->6
S->4->2->3->1->5
S->4->2->3->5
S->4->2->3->5->6
S->4->2->3->5->1
```

Solution 3.3.2 - Earliest Deadline (Ordered path with arrival times for `scenario1.txt`)

```
Starts at (24.0451,1.63808)
Arrives at city 4 loc (28.571,1.91245) at time 4.53421 for deadline 41.5744
Arrives at city 6 loc (27.8529,4.70066) at time 7.41341 for deadline 66.5376
Arrives at city 5 loc (2.27709,15.517) at time 35.1824 for deadline 77.6843
Arrives at city 2 loc (6.36816,19.3715) at time 40.8032 for deadline 43.7957
Arrives at city 3 loc (20.9536,25.5436) at time 56.6408 for deadline 60.1556
Arrives at city 1 loc (28.6092,20.4382) at time 65.8426 for deadline 77.3159
```

Solution 3.3.3 - Earliest Deadline (Paths, number of evaluations)

Scenario	Path?	Path	Evaluations (±1)
1	Yes	S 4 6 5 2 3 1	88
2	Yes	S 7 10 9 4 2 5 1 6 3 8	60123
3	Yes	S 4 8 7 6 5 1 2 3	5707
4	No		325656
5	Yes	S 4 3 10 14 15 12 13 6 2 9 1 8 7 5 16 11	127

Solution 3.3.1 - Closest Point

```
4
4 6
4 6 1
4 6 1 3
4 6 1 3 2
4 6 1 3 5
4 6 1 3 5 2
4 6 1 2
4 6 1 5
4 6 1 5 2
```

Solution 3.3.2 - Closest Point (Ordered path with arrival times for `scenario1.txt`)

```
Starts at (24.0451,1.63808)
Arrives at city 4 loc (28.571,1.91245) at time 4.53421 for deadline 41.5744
Arrives at city 6 loc (27.8529,4.70066) at time 7.41341 for deadline 66.5376
Arrives at city 5 loc (2.27709,15.517) at time 35.1824 for deadline 77.6843
Arrives at city 2 loc (6.36816,19.3715) at time 40.8032 for deadline 43.7957
Arrives at city 3 loc (20.9536,25.5436) at time 56.6408 for deadline 60.1556
Arrives at city 1 loc (28.6092,20.4382) at time 65.8426 for deadline 77.3159
```

Solution 3.3.3 - Closest Point (Paths, number of evaluations)

Scenario	Path?	Path	Evaluations (±1)
1	Yes	S 4 6 5 2 3 1	38
2	Yes	S 8 7 9 10 4 3 2 6 5 1	38
3	Yes	S 4 8 7 6 5 2 1 3	119
4	No		325656
5	Yes	S 1 2 9 5 10 7 16 14 4 3 12 13 15 11 6 8	17

Solution 3.3.4

Ordering the variables will not always improve the performance of DFS with backtracking in this problem. For instance, if we number locations by their deadlines, and the odd-numbered delivery locations are on one side of town and the even-numbered delivery locations are on the other side of town, we will go back and forth a long distance over and over again instead of visiting locations that are close to each other and making the longer journey only once. (Note: visiting locations close together will not always improve the situation either.) In this case, we got the same solution for scenario 1 with fewer evaluations. In scenario 2, variable ordering found a different solution with fewer evaluations, but a longer total delivery time. In scenario 3, we get the same solution, but this time it takes more evaluations. Scenario 4 is unchanged because we must search the entire space to determine that there is no solution. The variable ordering has made scenario 5 more tractable in this case.