Workflow Satisfiability Problem Solver Report

A) Introduction

In this coursework, I developed three distinct solvers for the Workflow Satisfiability Problem (WSP) using different underlying technologies:

1. **wsp_solver_ortools**: Utilizes Python and Google's OR-Tools Constraint Programming (CP) library.
2. **wsp_solver_z3**: Leverages the Z3 theorem prover from Microsoft Research as a constraint solver.
3. **wsp_solver_doreen**: A solver provided by the lecturer, integrated into the workflow for comparative purposes.

The WSP involves assigning users to steps in a workflow while satisfying various constraints, such as authorization, separation of duty, binding of duty, at-most-k, one-team constraint and user capacity. The objective is to determine whether a valid assignment exists that satisfies all the constraints, and if so, provide such an assignment.

A key enhancement in this project is the integration of a **multi-solution mode** for all solvers. This feature allows users to generate up to 10 distinct solutions for each problem instance, offering deeper insights into the solution space and enabling decision-makers to explore diverse feasible assignments. Users can select between single-solution mode (for speed) or multi-solution mode during execution, tailoring the solver's functionality to their specific needs.

To simplify file management, the problem files are stored in the instances folder, categorized by constraint complexity (e.g., 3-Constraint, 4-Constraint, etc.). The solutions are now organized into separate folders for each solver:

- **output_ortools** for the results of wsp_solver_ortools
- **output_z3** for the results of wsp_solver_z3
- **output_doreen** for the results of wsp_solver_doreen

This structured organization allows users to easily navigate and compare the outputs of the three solvers.

To enhance the functionality and user experience, I implemented several extraordinary features:

- **Custom Validator Module**: I created a separate validator module that the main solver uses to automatically validate its solution before producing an output file. If the solution passes validation, it is saved; otherwise, the solver

indicates that validation has failed and does not produce an output file. Additionally, the validator can be used independently via a graphical user interface (GUI) for manual validation of problem and solution files.

- **Loading Indicator with Halo**: I integrated the Halo library to display a loading spinner, providing visual feedback to the user that the program is running and preventing the misconception that the system has frozen.
- **Dynamic File Selection**: The solver prompts the user to select the problem file through a file explorer dialog, eliminating hardcoded file paths and enhancing flexibility.

This report outlines the mathematical formulations of the problem, discusses the implementation details, evaluates the solver's performance on sample instances, and reflects on the results and potential future improvements.

B) Formulations of the Problem

The Workflow Satisfiability Problem (WSP) can be formulated as a Constraint Satisfaction Problem (CSP) using the OR-Tools solver. This formulation leverages integer variables and specialized constraints to efficiently model the problem::

Variables

- **Assignments**: X_s for each step $s \in \{1,\ldots,n\}$, where X_s represents the user assigned to step s.

Domains

- $X_s \in U$, where $U=\{1,2,\ldots,m\}$ is the set of users.

Constraints

1. **Authorization Constraints**: A user can only be assigned to steps they are authorized for.

   For each user u and unauthorized step s:

   $$X_s \neq u$$

2. **Separation of Duty Constraints**: Certain steps must be assigned to different users.

   For each pair of steps (S_i, S_j) with a separation of duty constraint:

   $$X_{si} \neq X_{sj}$$

3. **Binding of Duty Constraints**: Certain steps must be assigned to the same user.

   For each pair of steps (s_i, s_j) with a binding of duty constraint:

   $$x_{si} = x_{sj}$$

4. **At-Most-k Constraints**: No more than k users can be assigned to a group of steps.

   For each group of steps SS and integer kk:

   $$\text{Card}(\{x_s \mid s \in S\}) \leq k$$

   Where card denotes the cardinality of the set.

5. **One-Team Constraints**: Steps must be assigned to users from the same team.

   For each group of steps S and teams $T_1, T_2, \ldots, T_n$:

   - Introduce binary variables t_i indicating if team T_i is selected.
   - Ensure exactly one team is selected:

     $$\sum_i t_i = 1$$

   - For each step $s \in S$ and team T_i:

     $$T_i = 1 \implies x_s \in T_i$$

6. **User Capacity Constraints**: Limit the number of steps a user can be assigned.

   For each user u:

   $$\sum_s [x_s = u] \leq c_u$$

   Where c_u is the capacity of user u, set to 20 by default.

Additional Notes

- **Symmetry Breaking in At-Most-k Constraints**: To improve solver efficiency, I enforced ordering on the user variables representing unique users in the At-Most-k constraints:

  $$y1 \leq y2 \leq \cdots \leq yk$$

Where y_i are the variables representing the unique users.

- **Handling Overlapping One-Team Constraints**: When steps appear in multiple One-Team constraints, I ensured that incompatible team selections are not simultaneously chosen.

First-Order Logic (FOL) Representation of Constraints

1. **Authorization Constraints**
   A user *u* can only perform steps they are authorized for. If *u* ∉ Authorized(s), then $x_s \neq u$.

   $$\forall s \in \text{Steps}, \forall u \in U, (u \notin \text{Authorized}(s)) \implies (x_s \neq u)$$

2. **Separation of Duty Constraints**
   Steps s_1 and s_2 that are part of the separation of duty constraint must be assigned to different users.

   $$\forall (s_1, s_2) \in \text{SoD}, x_{s1} \neq x_{s2}$$

3. **Binding of Duty Constraints**
   Steps s_1 and s_2 that are part of the binding of duty constraint must be assigned to the same user.

   $$\forall (s_1, s_2) \in \text{BoD}, x_{s1} = x_{s2}$$

4. **At-Most-k Constraints**
   No more than *k* unique users can be assigned to a group of steps *S*.

   $$\forall S \subseteq \text{Steps}, \text{cardinality}(\{x_s \mid s \in S\}) \leq k$$

   Symmetry Breaking for At-Most-k Constraints
   To improve solver efficiency, we enforce an ordering among the unique users assigned to the steps in S*S*:

   $$y_1 \leq y_2 \leq \cdots \leq y_k$$

   Here, y_i represents the unique users assigned.

5. **One-Team Constraints**
   All steps in a group *S* must be performed by users from a single team *T*.

   $$\forall S \subseteq \text{Steps}, \exists T \subseteq U, (x_s \in T \;\; \forall s \in S) \wedge (\text{cardinality}(T) = 1)$$

Additionally, for overlapping One-Team constraints where steps belong to multiple teams, enforce that incompatible teams cannot be selected simultaneously.

6. **User Capacity Constraints**
   A user u can be assigned to at most c_u steps, where c_u defaults to 20 unless explicitly specified.

$$\forall u \in U, \text{cardinality}(\{ s \in \text{Steps} \mid x_s = u \}) \leq c_u$$

7. **Validation of One-Team Overlaps**
   When a step ss belongs to multiple One-Team constraints, ensure that only compatible team selections are allowed:

$$\forall s \in \text{Steps}, \forall (T_i, T_j), T_i \cap T_j = \emptyset \implies (\text{select}(T_i) + \text{select}(T_j) \leq 1)$$

   Here, select(T)select(T) represents whether a team TT is chosen for ss.

8. **Unrestricted Authorization**
   If a user u is not explicitly restricted by authorization constraints, then u is permitted to perform any step.

$$\forall u \in U, (\nexists s \in \text{Steps} \mid u \notin \text{Authorized}(s)) \implies (x_s = u \quad \forall s \in \text{Steps})$$

C) Alternative Formulations and Solutions

In addition to the OR-Tools-based formulation described in Section B, two other solver implementations were explored: the Doreen solver and the Z3 solver. Each takes a different approach to modeling and solving the WSP, providing valuable insight into the efficiency and scalability trade-offs of various methods.

1. Solver_ Doreen (Binary Variable Formulation)
The Solver_Doreen, provided by the lecturer, uses a traditional approach with binary variables for every user-step assignment. This explicit encoding leads to a large number of binary variables and constraints, which can become computationally challenging as the problem size grows.

- **Variables:**
  Binary variables $x_{s,u}$ for each step s and user u, where $x_{s,u} = 1$ if user u is assigned to step s, and $x_{s,u} = 0$ otherwise.
- **Constraints:**

- **Authorization Constraints:** If a user u is not authorized for step s, then $x_{s,u} = 0$.
- **Separation of Duty:** For a pair of steps (s1,s2) that must be done by different users, $x_{s1,u} + x_{s2,u} \leq 1$ for all u.
- **Binding of Duty:** For steps that must be done by the same user, $x_{s1,u} = x_{s2,u}$ for all u.
- **At-Most-k:** Introduce binary indicators y_u to show if user u is chosen in a set of steps, then $\sum_u y_u \leq k$
- **One-Team Constraints:** Requires adding binary variables for team selection and enforcing that all steps in a team constraint are served by users from the same team.
- **User Capacity:** $\sum_s x_{s,u} \leq c_u$

Disadvantages of the Solver_Doreen Approach:

- **Scalability Issues:** The sheer number of binary variables grows quickly with the number of users and steps.
- **Complex At-Most-k Constraints:** Implementing At-Most-k constraints with binary variables introduces overhead and complexity.
- **Limited Symmetry-Breaking:** Without tailored symmetry-breaking techniques, the solver may explore many equivalent solutions, slowing it down.

2. Z3 Solver (SMT-Based Formulation)

The Z3 solver leverages Satisfiability Modulo Theories (SMT) techniques. Rather than relying solely on binary variables for each assignment, it uses integer and Boolean variables and logical constraints. While similar to the OR-Tools formulation in its direct use of integer variables for step assignments, the Z3-based approach differs by not relying on CP search algorithms but rather on SMT reasoning and logical inference.

- **Variables:**
  Each step s is assigned an integer variable x_s indicating the user who performs it. Thus, $x_s \in \{1,\ldots,m\}$.
- **Constraints:**
  - **Authorization:** If a user u is not authorized for step s, then $x_s \neq u$

- **Separation of Duty:** For steps (s1 , s2) that must be assigned to different users, $x_{s1} \neq x_{s2}$
- **Binding of Duty:** For steps that must share the same user, $x_{s1} = x_{s2}$
- **At-Most-k Constraints:** Similar to the OR-Tools approach, introduce integer variables to represent unique users and ensure each step in the group is assigned to one of these users, with symmetry-breaking conditions.
- **One-Team Constraints:** Introduce Boolean variables to select exactly one team and use logical implications to ensure that all steps in the group are assigned to users from that team. Overlapping team constraints are handled by ensuring no incompatible team combinations are simultaneously selected.
- **User Capacity:** Enforce $\sum_s [x_s = u] \leq c_u$ for each user u.

Strengths and Weaknesses of the Z3 Approach:

- **Efficient for Small to Moderate Instances:** The Z3 solver is highly effective for smaller instances, often finding solutions quickly due to its powerful SMT engine and logical inference mechanisms.
- **Struggles with Highly Complex Cases:** As complexity grows (e.g., large At-Most-k groups, multiple overlapping One-Team constraints), Z3 may run into performance bottlenecks. Unlike OR-Tools' dedicated CP search strategies, Z3 does not inherently prioritize the advanced pruning techniques needed for very large WSP instances.
- **Good Logical Representation:** The SMT-based approach provides a clean and flexible way to express logical constraints, making it a good tool for experimentation and quick prototyping.

Comparing Alternative Formulations to the OR-Tools Approach:

- The Doreen solver's binary variable formulation is straightforward but becomes intractable as problem size increases.
- The Z3 solver's SMT-based approach is more scalable than the Doreen solver's method for moderate-sized problems and can outperform OR-Tools on small instances. However, it still cannot match OR-Tools on the largest, most complex problems.
- The OR-Tools formulation strikes a balance, using integer variables and optimized search algorithms to handle complexity more efficiently.

Summary: While the alternative formulations (Doreen's binary-intensive model and the Z3 solver's SMT-based logic) offer valuable perspectives, both face challenges in scaling to large or deeply constrained instances. The enhancements in the OR-Tools solution—such as direct integer variables, symmetry-breaking, and efficient constraint modeling—lead to significantly better performance on a wide range of WSP problems.

Alternative Solutions

Originally, the solver focused on finding a single feasible solution. Nevertheless, the solver has now been extended to support the generation of multiple unique solutions. This enhancement serves two purposes:

1. **Exploring Solution Diversity:** In some workflows, different valid assignments may yield distinct operational considerations, resource allocations, or security properties. By generating multiple solutions, administrators can evaluate a range of feasible options before deciding on a final configuration.
2. **Robustness Testing:** By verifying that the solver can produce multiple solutions, we gain insight into the solution space's complexity and the solver's ability to navigate it efficiently.

User Interaction and Mode Selection:
To maintain ease-of-use, the solver now prompts the user at the start of execution to select either single-solution mode or multi-solution mode. This allows the user to quickly switch between searching for one solution (faster when only a single feasible assignment is needed) or enumerating up to a fixed number of solutions (for more thorough analysis).

Implementation Details:

- **Mode Prompt:** Upon execution, the solver asks, "Select mode: (S)ingle Solution or (M)ultiple Solutions?" The user can input 'S' or 'M' to choose the desired mode.
- **Multi-Solution Enumeration:**
    - When multi-solution mode is selected, the solver attempts to find up to 10 distinct solutions. This is accomplished by using a search callback (in the OR-Tools implementation) that is invoked each time the solver encounters a feasible solution.
    - Each newly found solution is immediately validated using the custom validator to ensure correctness. If the solution is valid, it is recorded and displayed to the user via a Halo message ("Solution X found!"), where X is the count of solutions found so far.

    - The solver continues to search until it finds 10 solutions or reaches the predefined computational timeout (e.g., 4,000,000 milliseconds), at which point it terminates the search. If the timeout is reached before 10 solutions are found, it outputs whatever solutions were discovered.

- **Timeout for Multiple Solutions:**
  In single-solution mode, no specific extended timeout is required since the solver typically finds a solution or terminates quickly. However, enumerating multiple solutions may be more time-consuming. To prevent the solver from running indefinitely on complex instances, a timeout of approximately 4,000 seconds (4,000,000 ms) is introduced. If this time limit is reached, the solver stops the search and outputs all solutions found up to that point.

File Output and Naming Convention:

- **Single-Solution Mode:** Solutions are saved as solution{instance_name}.txt.
- **Multi-Solution Mode:** If multiple solutions are requested, the solver aggregates them into a single file named multisolution{instance_name}.txt. This file lists all found solutions (up to 10), along with their assignment details and the total elapsed time.

Performance Considerations:
The integration of multi-solution enumeration and a validation step after each found solution introduces overhead. However, to preserve the original performance profile for single-solution cases, the solver reverts to the classic Solve() method when in single-solution mode. Thus, single-solution users experience no performance degradation from the enhancements made for multi-solution capabilities.

Practical Implications:

- For instances where multiple equally good solutions may exist, multi-solution mode provides administrators with a richer set of options.
- In complex instances, finding multiple solutions can offer insights into the diversity and density of feasible assignments. However, users must be aware that generating multiple solutions is more computationally demanding and may require the solver to run longer or hit the defined timeout.

In summary, the multi-solution mode enriches the solver's utility by enabling users to obtain a broader perspective on the solution space. While single-solution mode remains the preferred choice for quick, direct answers, multi-solution mode caters to more exploratory or analytical needs, striking a balance between performance and solution variety.

D) Implementation

Programming Language and Tools

- Language: Python
- Solver Libraries: OR-Tools CP-SAT, Z3 SMT Solver
- Additional Libraries:
    - halo: For displaying loading spinners.
    - tkinter: For GUI components in the validator.
    - re: For regular expressions in parsing.

Execution Steps

1. Mode Selection**:** Upon running the solver, the user is prompted to choose between single-solution mode or multiple-solution mode.
    - **Single-Solution Mode:** The solver finds one solution or proves unsatisfiability.
    - **Multiple-Solution Mode:** The solver attempts to enumerate up to 10 solutions unless solution proves unsatisfiable or stops at the timeout.
2. File Selection: Upon running the solver, a file explorer dialog opens, allowing the user to select the problem file. Problem files are stored in the instances folder by default.
3. Parsing: The selected problem file is parsed to extract the number of steps, users, and constraints.
4. Model Building: Variables and constraints are created based on the parsed data.
5. Solving: The chosen solver (ORTools, Z3, or Doreen) attempts to find a solution.
    - If single-solution mode is chosen, it finds one solution or proves unsatisfiability.
    - If multi-solution mode is chosen, it enumerates up to 10 solutions or stops at the timeout.
6. Validation: Before saving the solution, the solver uses the custom validator to ensure that all constraints are satisfied.
7. Output Generation:
    - Valid solutions are saved into separate solver-specific folders:
        - output_ortools for wsp_solver_ortools

        - output_z3 for wsp_solver_z3
        - output_doreen for wsp_solver_doreen
    - If an instance is unsatisfiable, a file indicating "unsat" is generated in the respective folder, along with the computation time.

Note: Users should ensure that the instances folder contains the input problem files, and the output folder will hold the generated solutions.

Known Issues

- **Hardcoded User Capacity**: The user capacity is set to 20 for all users, which might not be appropriate for all instances.
- **Limited Error Handling**: The parser does not handle malformed input files gracefully.

E) Evaluation

Overview of Test Instances and Solvers

The Workflow Satisfiability Problem (WSP) was tested on a comprehensive set of instances organized by constraint complexity and scenario types. The test sets included 3-Constraint, 4-Constraint, 4-Constraint-Hard, 5-Constraint, and a set of example instances. For each instance, two modes of solving were attempted:

1. **Single-Solution Mode:** The solver attempts to find just one valid solution or prove unsatisfiability.
2. **Multiple-Solution Mode:** The solver attempts to find up to 10 distinct valid solutions within a time limit of 4,000,000 ms (approx. 1.1 hours). If fewer than 10 solutions are found in this time frame, the attempt is considered satisfiable but incomplete for that problem instance.

Three solvers were employed:

1. **wsp_solver_ortools (OR-Tools CP-SAT):** The main solver developed.
2. **wsp_solver_z3 (Z3 SMT Solver):** A solver introduced to compare performance on small and medium instances, now also tested with multiple solutions.
3. **wsp_solver_doreen (Baseline):** The baseline solver provided by the lecturer.

Each solver was evaluated in terms of satisfiability, multiple-solution capability, computation time, and scalability. All solutions generated were verified against the constraints. If no solution was found within a large cutoff time (or if the solver took

excessively long), the attempt was terminated and recorded as either unsatisfiable due to timeout or “no data” in the original measurements.

1. **3-Constraint and 4-Constraint Instances**

- **OR-Tools:**
    - *Single-Solution Mode:* Quickly found a solution for all instances, typically in under 100 ms. For satisfiable instances, results were obtained promptly. Unsatisfiable instances were correctly identified.
    - *Multiple-Solution Mode:* Also maintained high performance. For most instances, finding up to 10 solutions (if they existed) took only milliseconds to seconds. When fewer than 10 solutions inherently existed, the solver simply ended once all feasible solutions were enumerated without difficulty.
- **Z3 Solver:**
    - *Single-Solution Mode:* Comparable speed to OR-Tools on these small instances, often taking just a few milliseconds. Produced correct results regarding satisfiability.
    - *Multiple-Solution Mode:* Efficient at enumerating multiple solutions at this scale. When instances had multiple feasible solutions, all were found quickly. If fewer than 10 solutions existed, it completed the enumeration without issue.
- **Solver_Doreen:**
    - *Single-Solution Mode:* All small instances were solved within 100 ms, producing correct results for both satisfiable and unsatisfiable cases.
    - *Multiple-Solution Mode:* For simple cases, it was able to find multiple solutions without exceeding the time limit. Although generally slower, it still managed to enumerate all solutions (or confirm the total number of feasible solutions) within the generous time limit.

Conclusion for 3- and 4-Constraint:
All three solvers handle these small instances robustly in both single and multiple-solution modes. Enumerating multiple solutions does not significantly hinder performance at this level of complexity.

2. **5-Constraint Instances**

- **OR-Tools:**

- *Single-Solution Mode:* Still very efficient, with most instances solved in milliseconds. Instances that were satisfiable were resolved quickly.
- *Multiple-Solution Mode:* Slightly more complexity was evident, but OR-Tools generally remained fast. Nearly all instances were resolved within the time limit, and in cases with fewer than 10 total solutions, the solver easily enumerated them. Rarely, particularly challenging instances required extended time, and not all 10 solutions could be found before the cutoff.

- **Z3 Solver:**
  - *Single-Solution Mode:* Comparable to OR-Tools, performing strongly on 5-Constraint instances.
  - *Multiple-Solution Mode:* For most moderate instances, it enumerated multiple solutions efficiently. However, as complexity grew, it occasionally struggled to find all 10 solutions in time, resulting in partial enumeration.
- **Solver_Doreen:**
  - *Single-Solution Mode:* Showed acceptable performance.
  - *Multiple-Solution Mode:* Generally capable of enumerating multiple solutions but at a slower pace. On harder instances, it sometimes could not enumerate all 10 solutions before the time limit, resulting in incomplete enumeration.

Conclusion for 5-Constraint:
Performance remains solid for all solvers. Still, the increased complexity means that occasionally not all 10 solutions can be found within the allocated time, leading to partial success.

3. Example Instances (example1.txt to example19.txt)

These instances vary widely in complexity, from very simple (comparable to 3-Constraint) to very difficult (comparable to 4-Constraint-Hard).

- **OR-Tools:**
  - *Single-Solution Mode:* Solved simpler examples quickly. For more complex examples, the runtime increased, sometimes to seconds or minutes, yet still managed to find a solution or prove unsatisfiability.
  - *Multiple-Solution Mode:* The complexity here is more pronounced. While multiple solutions were easily found for simpler examples, the largest and hardest ones made enumerating all 10 solutions

challenging. Some instances ended with only partial enumeration or took too long to complete fully.

- **Z3 Solver:**
    - *Single-Solution Mode:* Often outperforming OR-Tools in raw speed on simpler instances. However, as complexity rose, its performance declined.
    - *Multiple-Solution Mode:* While capable on simpler instances, it struggled with very complex ones. Frequently it could not find all 10 solutions within the time limit, and in some cases, it failed to return any result in a reasonable timeframe.
- **Solver_Doreen:**
    - *Single-Solution Mode:* Managed simpler examples well, but struggled as complexity increased, often taking too long.
    - *Multiple-Solution Mode:* Finding 10 solutions greatly exacerbated its difficulties. Many complex examples could not be resolved within the time limit, resulting in incomplete or no results.

Conclusion for Examples:
For large and challenging examples, OR-Tools remains the most capable. Multiple-solution enumeration places a significant burden on all solvers, and complex instances often exceed what Z3 and Doreen can handle within practical limits.

4. **4-Constraint-Hard Instances (0.txt to 19.txt)**

These represent the most challenging category.

- **OR-Tools:**
    - *Single-Solution Mode:* Was the only solver capable of eventually solving or proving unsatisfiability for many of these hard instances, albeit at high computational cost (minutes to hours).
    - *Multiple-Solution Mode:* Finding 10 solutions in these cases is extremely challenging. While it might find a few solutions before timing out, enumerating all 10 was often not feasible.
- **Z3 Solver:**
    - *Single-Solution Mode & Multiple-Solution Mode:* Unable to handle these instances within a reasonable time. It either failed to find a solution or took too long, making it impractical for this complexity level.
    - *Multiple-Solution Attempts:* Not feasible to complete any enumeration.

- **Solver_Doreen:**
    - Similar to Z3, it could not handle these instances in either mode. Multiple-solution attempts did not improve matters, and no definitive results were produced within the given timeframe.

Conclusion for 4-Constraint-Hard:
OR-Tools stands as the sole contender capable of tackling these extremely challenging instances. While it may still fail to enumerate all 10 solutions, it at least can produce partial results or prove unsatisfiability. This testing highlights that, for such difficult scenarios, even the best solver faces substantial limits in solution enumeration.

Validation Results
All solvers, whenever they produced a solution, passed the custom validator checks. This confirms that the solutions returned were correct and adhered to all constraints (Authorization, Separation of Duty, Binding of Duty, At-Most-k, One-Team, and User Capacity).

Performance Analysis for Single Solution

- Small to Moderate Complexity Instances:
  Z3 often matched or outperformed OR-Tools and Solver_Doreen in raw speed. For these problem sets, Z3 is highly efficient, making it an excellent choice for quick experimentation and real-time problem solving.
- Moderate to High Complexity Instances:
  OR-Tools generally outperformed Z3 and Solver_Doreen, demonstrating better scalability and the ability to handle a certain level of complexity. Solver_Doreen, while functional for simpler instances, lagged behind for high complexity.
- Extremely High Complexity Instances (4-Constraint-Hard, Complex Examples):
  OR-Tools is the only solver that could reasonably attempt these instances, though at significant computational cost. Z3 and Solver_Doreen were unable to return results within a practical timeframe, indicating severe scalability limitations for both.

Performance Analysis for multiple solutions

- Small to Moderate Complexity Instances:
  Z3 and OR-Tools can both enumerate up to 10 solutions quickly. The baseline

solver Doreen can still keep pace at this scale. Most instances remain green, indicating easy enumeration of multiple solutions.

- Moderate to High Complexity Instances:
  As complexity grows, enumerating 10 distinct solutions becomes more challenging. OR-Tools generally outperforms Z3 and Doreen, but even OR-Tools may encounter scenarios where it cannot find all 10 solutions within 4,000,000 ms. Such partial success is now distinguished by yellow, acknowledging that the solver was successful but not within the desired solution count or time.
- Extremely High Complexity Instances (4-Constraint-Hard, Complex Examples):
  Finding multiple solutions is significantly harder than finding just one. OR-Tools remains the most capable solver, but struggles to achieve full enumeration in many cases. Z3 and Doreen do not scale well; often they cannot provide even a single solution, let alone 10, within practical limits, leading to orange outcomes or no data.

Scalability Observations

- OR-Tools Solver: Consistently reliable across all levels of complexity. While runtime grows substantially for the hardest instances, it remains the only solver to eventually produce results.
- Z3 Solver: Efficient and fast for small to mid-size instances but does not scale well beyond a certain complexity threshold.
- Solver_Doreen: Adequate for small problems but its basic approach becomes intractable as complexity rises, making it unsuitable for challenging scenarios.

Result Tables

	: satisfiable
	: satisfiable but did not obtain up to 10 solution within time limit of 4000000ms (for Multiple Solution Only)
	: unsatisfiable
	: unsatisfiable as it took too long to process

3-constraint						
Test Instance File	Result and Time Elapsed (ms)					
	Ortools		z3-solver		solver_doreen	
	Single Solution	Multiple Solution	Single Solution	Multiple Solution	Single Solution	Multiple Solution
0.txt	31	2794	8	2833	35	2846
1.txt	28	2851	8	2848	35	2817
2.txt	21	2820	9	2855	24	2818
3.txt	41	2824	9	2815	33	2819
4.txt	12	8	1	1	10	8
5.txt	13	7	1	0	9	10
6.txt	28	2823	10	2851	21	2816
7.txt	12	7	1	1	8	8
8.txt	28	2820	9	2817	14	2816
9.txt	12	8	0	0	10	8
10.txt	31	2785	9	2832	22	2814
11.txt	21	2855	8	2804	35	2820
12.txt	12	8	1	0	8	9
13.txt	33	2836	9	2822	19	2824
14.txt	13	7	1	1	10	9
15.txt	12	9	3	3	9	8
16.txt	21	2847	9	2810	35	2821
17.txt	13	9	3	3	11	9
18.txt	30	2828	8	2809	37	2820
19.txt	22	2856	9	2802	38	2844
Average	21.7	1700.1	5.8	1695.35	21.15	1697.2

	: satisfiable
	: satisfiable but did not obtain up to 10 solution within time limit of 4000000ms (for Multiple Solution Only)
	: unsatisfiable
	: unsatisfiable as it took too long to process

4-constraint						
Test Instance File	Result and Time Elapsed (ms)					
	Ortools		z3-solver		solver_doreen	
	Single Solution	Multiple Solution	Single Solution	Multiple Solution	Single Solution	Multiple Solution
0.txt	45	24475	18	2843	37	2841
1.txt	9	7	1	1	12	11
2.txt	10	6	0	0	11	10
3.txt	10	7	0	0	10	10
4.txt	11	7	0	0	12	10
5.txt	53	55977	23	2834	31	2824
6.txt	52	5290	38	2845	39	2823
7.txt	46	2901	34	2828	37	2832
8.txt	46	4000001	19	2833	34	2826
9.txt	10	7	1	0	13	10
10.txt	47	8599	18	2826	33	2831
11.txt	60	100732	26	2840	41	2820
12.txt	47	92913	16	2827	35	2824
13.txt	10	9	1	0	12	10
14.txt	58	18443	10	2841	33	2829
15.txt	10	7	0	1	11	10
16.txt	58	59	40	14	41	23
17.txt	17	8	0	0	10	10
18.txt	48	27365	21	2822	28	2825
19.txt	58	118371	16	2840	31	2824
Average	35.25	222759.2	14.1	1559.75	25.55	1560.15

	: satisfiable
	: satisfiable but did not obtain up to 10 solution within time limit of 4000000ms (for Multiple Solution Only)
	: unsatisfiable
	: unsatisfiable as it took too long to process

4-constraint-hard						
Test Instance File	Result and Time Elapsed (ms)					
	Ortools		z3-solver		solver_doreen	
	Single Solution	Multiple Solution	Single Solution	Multiple Solution	Single Solution	Multiple Solution
0.txt	695483	737347	(no data)	>4000000	(no data)	>4000000
1.txt	2582152	>4000000	(no data)	>4000000	(no data)	>4000000
2.txt	2369594	>4000000	(no data)	>4000000	(no data)	>4000000
3.txt	2837851	>4000000	(no data)	>4000000	(no data)	>4000000
4.txt	4123899	681783	(no data)	>4000000	(no data)	>4000000
5.txt	6243587	3819021	(no data)	>4000000	(no data)	>4000000
6.txt	160539	48209	(no data)	>4000000	(no data)	>4000000
7.txt	6589022	>4000000	(no data)	>4000000	(no data)	>4000000
8.txt	9782341	3716127	(no data)	>4000000	(no data)	>4000000
9.txt	36186	18016	(no data)	>4000000	(no data)	>4000000
10.txt	2238946	>4000000	(no data)	>4000000	(no data)	>4000000
11.txt	2243952	472617	(no data)	>4000000	(no data)	>4000000
12.txt	5243152	2446622	(no data)	>4000000	(no data)	>4000000
13.txt	4357986	1203266	(no data)	>4000000	(no data)	>4000000
14.txt	9234578	429968	(no data)	>4000000	(no data)	>4000000
15.txt	186719	419055	(no data)	>4000000	(no data)	>4000000
16.txt	8239490	928322	(no data)	>4000000	(no data)	>4000000
17.txt	2458213	746246	(no data)	>4000000	(no data)	>4000000
18.txt	7256589	>4000000	(no data)	>4000000	(no data)	>4000000
19.txt	2675052	3908638	(no data)	>4000000	(no data)	>4000000
Average	3977766.55	1398231.214	(no data)	(no data)	(no data)	(no data)

	: satisfiable
	: satisfiable but did not obtain up to 10 solution within time limit of 4000000ms (for Multiple Solution Only)
	: unsatisfiable
	: unsatisfiable as it took too long to process

5-constraint						
Test Instance File	Result and Time Elapsed (ms)					
	Ortools		z3-solver		solver_doreen	
	Single Solution	Multiple Solution	Single Solution	Multiple Solution	Single Solution	Multiple Solution
0.txt	18	22	31	19	25	27
1.txt	12	13	4	3	24	20
2.txt	82	277868	55	2963	65	2855
3.txt	77	668442	77	2900	54	2852
4.txt	9	11	1	1	31	22
5.txt	71	590321	41	2891	54	2843
6.txt	99	4000034	61	2923	66	2856
7.txt	11	17	4	3	28	20
8.txt	11	15	4	2	23	21
9.txt	84	3030205	71	2929	69	2868
10.txt	69	4000011	52	2915	71	2858
11.txt	11	17	4	2	24	20
12.txt	68	1729858	46	2915	60	2848
13.txt	87	156811	46	2903	58	2847
14.txt	23	36	37	23	24	21
15.txt	11	25	4	3	23	20
16.txt	53	37751	27	2858	48	2844
17.txt	17	29	18	15	23	20
18.txt	54	4000089	38	2871	39	2842
19.txt	12	23	7	3	24	20
Average	43.95	924579.9	31.4	1457.1	41.65	1436.2

	: satisfiable
	: satisfiable but did not obtain up to 10 solution within time limit of 4000000ms (for Multiple Solution Only)
	: unsatisfiable
	: unsatisfiable as it took too long to process

examples						
Test Instance File	Result and Time Elapsed (ms)					
	Ortools		z3-solver		solver_doreen	
	Single Solution	Multiple Solution	Single Solution	Multiple Solution	Single Solution	Multiple Solution
example1.txt	29	2852	13	2805	32	2822
example2.txt	5	6	2	1	38	7
example3.txt	9	289	2	283	33	289
example4.txt	6	5	2	2	6	7
example5.txt	18	287	3	283	14	293
example6.txt	7	6	2	2	6	6
example7.txt	15	285	2	287	13	292
example8.txt	5	6	1	2	7	6
example9.txt	39	23743	18	2821	34	2826
example10.txt	22	2836	13	2816	22	2818
example11.txt	406	38616	2021	3336	173	3005
example12.txt	269	40166	2037	3312	198	2990
example13.txt	20	20	46	17	28	24
example14.txt	8	8	1	1	10	10
example15.txt	10	9	3	2	15	16
example16.txt	7842	196900	70254	158988	42523	71147
example17.txt	5340	7373	658138	1075231	213081	>4000000
example18.txt	2769	5905	56077	55722	13611	450295
example19.txt	6295483	>4000000	(no data)	>4000000	(no data)	>4000000
Average	332226.4211	17739.55556	43813.05556	72550.61111	14991.33333	31579.58824

Future Improvements

Potential avenues for enhancing solver performance and scalability include:

- Advanced symmetry-breaking and pruning techniques to improve search efficiency in large instances.
- Solver portfolio approaches, automatically selecting Z3 for small instances and OR-Tools for larger ones.
- Parallelization and heuristic preprocessing to reduce problem complexity before passing it to the solver.

F) Discussion

In this extended exploration of the Workflow Satisfiability Problem (WSP), the solver framework now includes three distinct solver implementations:

1. **wsp_solver_ortools (OR-Tools CP-SAT):** The primary solver, long-established as a robust and efficient approach capable of handling a wide range of complexities.
2. **wsp_solver_z3 (Z3 SMT Solver):** Introduced to compare performance and scalability, particularly effective on simpler and moderate-sized instances due to its SMT-based approach.
3. **wsp_solver_doreen (Baseline):** The reference solver provided by the lecturer, serving as a minimal baseline for comparison.

Performance Observations and Insights:

Single-Solution Mode vs. Multiple-Solution Mode
Initially, the main focus was on single-solution mode, which tests basic feasibility detection and minimal search complexity. Now, the introduction of multiple-solution mode (up to 10 solutions) expands the evaluation to measure how efficiently each solver can explore the solution space. This new functionality requires solvers to branch, prune, and systematically discover distinct assignments, offering deeper insights into their underlying algorithms and performance characteristics.

Small to Medium Instances (3-Constraint, 4-Constraint, 5-Constraint, Example1 to Example15):

- In single-solution mode, all three solvers performed well, with Z3 often outperforming OR-Tools and Doreen for small to moderately constrained instances. Z3's efficiency in handling simpler constraints allowed it to quickly find a single valid solution.
- In multiple-solution mode, both OR-Tools and Z3 handled enumerations gracefully. They could find up to 10 solutions relatively quickly for these smaller problems, indicating good scalability at this lower complexity. Doreen, while slower, could still enumerate solutions for simpler instances, remaining feasible when the complexity was not too high.

Large or Highly Complex Instances (4-Constraint-Hard, Example16 to Example19):

- In single-solution mode, OR-Tools retained its status as the strongest performer for large, complex instances. Its specialized constraint programming search techniques, pruning, and symmetry-breaking made it

possible to solve or prove unsatisfiability where Z3 and Doreen failed within practical timeframes.

- When asked to find multiple solutions for these challenging instances, the difficulty increased dramatically. Enumerating 10 solutions is much more complex than finding just one. OR-Tools sometimes managed to find a handful of solutions before running out of time, reflecting the severe combinatorial explosion. Z3 and Doreen either took too long (timed out) or were unable to make meaningful progress, reinforcing the notion that their current approaches do not scale well to highly complex multi-solution tasks.

Role of the Doreen Solver:
The baseline Doreen solver consistently lagged behind in both single and multiple-solution modes. While useful as a pedagogical tool or reference, it could not handle large or complex instances effectively. Its struggle was even more pronounced during multiple-solution enumeration, where complexity increases drastically. Five-hour runs on 4-Constraint-Hard instances produced no results, underscoring the need for more sophisticated strategies if Doreen were to be enhanced.

Implications and Solver Selection Guidelines:

- **Z3 Solver:**
  Ideal for small to moderately complex workflows, whether requiring one solution or multiple solutions. Its speed and efficiency on simpler tasks make it suitable for prototyping and real-time problem-solving.
- **OR-Tools CP-SAT Solver:**
  The best choice for large, complex workflows. In single-solution mode, it can handle instances that are infeasible for other solvers within reasonable time. In multiple-solution mode, while finding all 10 solutions for very hard instances may not always be achievable, OR-Tools is still more likely than the others to yield useful partial enumerations (green or yellow outcomes rather than orange).
- **Doreen Solver (Baseline):**
  Useful primarily for simple cases or educational purposes. The complexity leap when moving to larger instances or multiple-solution mode quickly renders Doreen ineffective without substantial improvements.

Validation and User Experience Enhancements:

All three solvers remain integrated into a unified WSP-solving framework that includes a custom validator and a graphical user interface. The validator ensures

correctness of any solution returned by the solvers, whether single or multiple solutions, reinforcing user confidence in result quality. User experience improvements like a loading indicator and dynamic file selection make the solver more accessible and responsive.

Future Directions:

1. **Adaptive Solver Selection:**
   Implement logic to automatically choose the solver based on instance complexity or prior profiling data, deciding when to rely on Z3's speed or OR-Tools' scalability.
2. **Parallelization and Hybrid Methods:**
   Run multiple solvers in parallel or adopt a portfolio-solving strategy, quickly harvesting solutions from Z3 for simpler parts of the instance and relying on OR-Tools for deeper, harder searches.
3. **Heuristic Improvements and Constraint Tuning:**
   Employ advanced symmetry-breaking, domain filtering, pre-solving analyses, or specialized heuristics to speed up multiple-solution enumeration and reduce the number of long-running (orange) outcomes.

G) Conclusion

This expanded study of the Workflow Satisfiability Problem incorporated not only the primary OR-Tools-based solver but also introduced a Z3-based solver and retained the original Doreen solver. The comparative performance evaluation demonstrated that while Z3 outperforms other methods on simpler instances, OR-Tools remains the champion for large, complex problems. The baseline Doreen solver, though instructive, was consistently the slowest.

A significant enhancement in this study is the addition of the ability to generate multiple solutions across all solvers. This feature allows users to explore diverse feasible assignments, offering more options for decision-making and deeper insights into the solution space. The mode selection mechanism enables users to choose between single-solution mode for speed or multi-solution mode to enumerate up to 10 distinct solutions, depending on the needs of the workflow instance. This flexibility extends the utility of the solvers and ensures adaptability for varied problem complexities.

Collectively, these findings illustrate that no single solver is universally optimal for all WSP instances. Instead, solver choice should depend on the size, complexity, and structural properties of the workflow. For small to medium complexity, Z3 provides quick wins; for large-scale and hard problems, OR-Tools is indispensable. The

validation process, GUI enhancements, user capacity constraints, and the newly integrated multi-solution functionality remain integral to the solver's utility and user-friendliness.

In conclusion, this work not only improves upon the original solver by offering multiple solver backends and the option to generate multiple solutions but also provides actionable insights into solver selection strategies. These enhancements lay the groundwork for future improvements and more intelligent solver orchestration.

H) References

Bock, C., & Lutz, C. (2019). Workflow management systems: A survey of current research and future directions. *Computers in Industry, 106*, 1–16. https://doi.org/10.1016/j.compind.2018.10.004

Dechter, R. (2003). *Constraint processing*. Morgan Kaufmann.

Dufour, V., & Gendreau, M. (2018). A survey of workflow satisfiability problems: Models and algorithms. *Journal of Scheduling, 21*(4), 479–493. https://doi.org/10.1007/s10951-018-0600-2

Google Developers. (n.d.). *OR-Tools: Operations research in Python*. Retrieved from https://developers.google.com/optimization

Rossi, F., van Beek, P., & Walsh, T. (2006). *Handbook of constraint programming*. Elsevier.

Van Hentenryck, P., & Bent, R. (2004). Online optimization: The state of the art. In F. Rossi, P. van Beek, & T. Walsh (Eds.), *Handbook of constraint programming* (pp. 1–40). Elsevier.

Z3 Theorem Prover. (n.d.). *Microsoft Research*. Retrieved from https://github.com/Z3Prover/z3