Extending the Lifetime of Coarse-grained Runtime Reconfigurable FPGAs by Balancing Processing Element Usage

Bo Hu, Mustafa Shihab, Yiorgos Makris, Benjamin Carrion Schaefer, Carl Sechen
bo.hu, mustafa.shihab, yiorgos.makris, schaferb, carl.sechen@utdallas.edu
Department of Electrical & Computer Engineering, The University of Texas at Dallas,Richardson, TX, U.S.A, 75080-3021

***Abstract*—Contemporary coarse-grained runtime reconfigurable architectures (CGRRAs) are increasingly sensitive to aging effects such as Negative Bias Temperature Instability (NBTI). To address this, we propose a reliability-aware floorplanner for CGRRAs based on a mixed Linear Programming (LP) and Integer Linear Programming (ILP) method that extends the Mean Time to Failure (MTTF) of CGRRAs by balancing processing element (PE) usage. We use this as the basis of a design space explorer that generates a variety of configurations, trading off PE displacement vs. MTTF. On average, a 2.4× improvement in MTTF was obtained for an average critical path delay increase of under 2 percent (although most benchmarks had no delay increase) compared to the default lifetime unaware floorplan.**

***Index Terms*—NBTI; MILP; lifetime; floorplan; CGRRA;**

I. Introduction

Heterogeneous Systems-on-Chips (SoCs) typically include processors, memories, peripherals and a diverse number of dedicated hardware accelerators. These accelerators execute dedicated functions one to two orders of magnitude faster than general purpose processors. In order to reduce the development time and cost of these SoCs, companies started relying on embedded FPGAs (eFPGAs) onto which to map these accelerators, called Reconfigurable SoCs (RSoCs). Coarse-grained FPGAs are often runtime reconfigurable and thus, are often called Coarse-Grained Runtime Reconfigurable Architectures (CGRRAs). Fig. 1 shows a block diagram of an RSoC which includes a CGRRA. A state of the art CGRRAs, the Stream Transpose Processor (STP) [1], is used in this paper. The runtime reconfiguration is achieved by dividing an application into multiple contexts, where each context is mapped onto the CGRRA in a separate clock cycle at runtime. The CGRRA is reconfigured every clock cycle for each context and this significantly increases the stress on the fabric and hence accelerates aging mechanisms such as NBTI (Negative Biased Temperature Instability) which dominates the reliability degradation factors and are usually measured by the increase in the magnitude of the threshold voltage [2], [3].

Lifetime-aware CAD flows have been proposed in the past [2] [4] [5]. These flows have also been extended to FPGAs, making use of the flexibility of the FPGAs to reconfigure themselves [**?**], [6]–[8]. For CGRRAs, [3] proposes periodic remapping of the design to less-used regions by using two different configurations and switching between them. The authors in [9] proposed a method to lower the temperature of the CGRRA by generating different configurations. Similarly, Gu, *et al.* [10] provides a rotation based mapping strategy to balance the stress on multi-context CGRRAs. However, none of the above approaches take into consideration performance variation introduced by the stress time balance strategies.

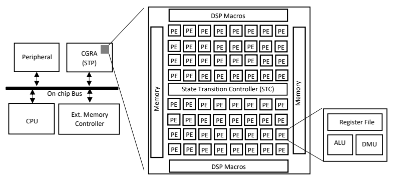

Fig. 1: Reconfigurable SoC with CGRRA eFPGA.

II. Stream Transpose Processor CGRRA

The Stream Transpose Processor (STP) [1] is used in SoCs as a reconfigurable IP. Its main component is the *tile*, each of which consists of an array of 8×8 PEs surrounded by embedded memory and multipliers. The STP can hold up to 64 contexts in its State Transition Controller (STC) located in the middle of the PE array. The STP is composed of tiles which is in turn built out of PEs. Each PE contains an 8-bit arithmetic logic unit (ALU), an 8-bit data manipulation unit (DMU) for 1-bit logic operations and 8-bit shifting and masking, and an 8-bit flip-flop unit (FFU).

Fig. 2(a) shows a diagram of an STP configuration flow. The STP takes ANSI-C as its input and performs HLS to parallelize it. The result is then mapped onto the STP's architecture, placed and routed. On the other hand, the code for the STC is generated with the configuration file for each of the contexts. The typical architecture after HLS is a FSM and a data path, where the FSM generates the control signals for the data path. In the case of the STP, each FSM state is a context, and a new context is loaded every clock cycle onto the device. The latency of the circuit therefore determines the number of contexts and vice versa. This is highlighted in the left column of Fig. 2(b), as the original floorplan. In this case 4 contexts are mapped to the STP, one per clock cycle. The gray area denotes the PEs being used in that particular context. The maximum fabric size is therefore determined by the context which makes use of the maximum number of PEs.

With respect to Fig. 2(b), suppose that the stress time of each used PE is 1 unit, as indicated by the shaded PEs. Summed over all contexts, some PEs have an accumulated stress time of 4 units after the lifetime unaware floorplanning. As shown in the right column of Fig. 2(b), a new lifetime aware floorplan results in a decreased accumulated stress time of 2 and PE usage is distributed across the entire fabric.

III. NBTI modeling and MTTF Computation

The threshold voltage V_{th} model of NBTI mechanisms used in this work follows the standard formulation used in most

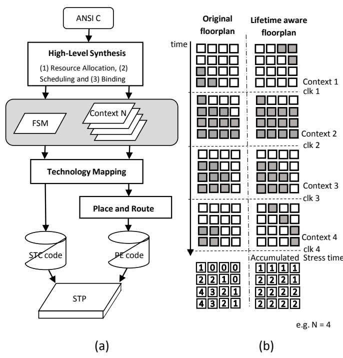

Fig. 2: (a) STP configuration flow diagram and (b) mapping and re-mapping of contexts, one context per clock cycle.

previous work [2], [3], [7], [10]. The threshold voltage V_{th} will increase as indicated in equation 1, where A_{NBTI} is a technology dependent factor, n is a fabrication dependent constant, k is Boltzmann's constant, Ea is the activation energy, T is the temperature, V_{th0} is the starting V_{th}, t_0 is the corresponding starting time when $V_{th} = V_{th0}$ and $ST_{t_0 \rightarrow t}$ is the stress time from t_0 till time t which equals $SR \times (\mathrm{t}_0 \rightarrow t)$. SR is the stress rate and equals the duty cycle, which is the ratio of time that a transistor is on.

$$V_{th} = A_{NBTI} \times (ST_{t_0 \rightarrow t})^n \times e^{(-Ea/kT)} \times V_{th0} \qquad (1)$$

The MTTF model used in this work assumes failure due to a threshold voltage (V_{th}) increase by V_{th_shift}. V_{th_shift} is defined as V_{th_MTTF} - V_{th0}, in which V_{th_MTTF} is the MTTF threshold voltage related to the MTTF as shown in Fig. 3. Thus, computing V_{th} allows us to compute the MTTF of the CGRRA for the default PE floorplan. The blue curve represents the original case without any lifetime aware remapping, while the orange curve represents the case with the lifetime aware remapped floorplan. We assume that the fabric will fail when the V_{th_shift} reaches a certain boundary (*e.g.* 10% [3]) and the corresponding operation time will be the MTTF. The orange curve has lower slope compared to the blue curve, which means higher MTTF. The critical factors which determine V_{th_shift} are the highest stress time and temperature (T) among all PEs. For a given PE, the accumulated stress time is the total stress time of the PE after execution of all contexts.

In order to obtain the original MTTF (blue curve in Fig. 3), we first determine the stress time for each PE in the provided floorplan. The stress time of each PE in every clock cycle (every context) may be different because different functional units may be mapped to a PE in different contexts. For example, the characterization of the ALU and DMU within one PE shows their delays as 0.87 ns and 3.14 ns, which divided by the clock period, are their stress rates, respectively. Next, the commonly used thermal simulator HotSpot [11] is used to compute the temperature of each PE. The thermal simulator takes the stress time maps and floorplans generated in the lifetime unaware mapping generation phase as input information to generate a thermal map of each context. The PE with the maximum accumulated temperature (T) across all contexts is then identified and the corresponding stress

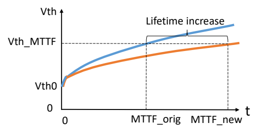

Fig. 3: Threshold voltage shift (V_{th}) vs. MTTF of both the original and re-mapped floorplans [12].

times can be obtained from the floorplan. Thus, V_{th_shift} is computed using equation 1 and a plot as in Fig. 3 is used to obtain the failing point MTTF. Using the lifetime-aware floorplan, a new larger value of MTTF as in the orange curve in Fig. 3 is obtained as the maximum accumulated stress times are reduced.

IV. Proposed CAD design flow for CGRRAs

The input to our proposed CAD flow for CGRRAs is a behavioral description for HLS and the output a lifetime-aware floorplan consisting of N contexts, which are used to configure the CGRRA fabric every clock cycle.

Phase 1: Lifetime-unaware PE Mapping MTTF Computation: This first phase of the proposed flow takes as input the behavioral description to be mapped onto the CGRRA targeted in this work and executes its complete VLSI design flow from HLS to bitstream generation. This includes the technology mapping onto the PEs and the placement and routing. The commercial CAD flow provided with the CGRRA is used for this purpose [1]. The result is a lifetime-unaware configuration that minimizes the bounding box area of the used PEs, while meeting the specified timing constraint.

This phase continues by taking as input the PEs used in each context and the operations mapped on each PE and calculates the *stress time* of each PE and their temperature. Based on the NBTI modeling and MTTF computation equation 1, the MTTF is computed. The result from this phase is the baseline PE mapping in each of the contexts.

Phase 2 : Lifetime-aware re-mapping Design Space Exploration: This phase re-binds the operations in each context to new PEs, *e.g.*, operation j in context i to PE k, in order to increase the MTTF. In this case PE_k is the k^{th} PE in the array of PEs. In this work we formulate this re-binding problem as a relaxed mixed LP and ILP (MILP) formulation as in equation 2. The number of contexts (i) is c, M_i is the number of operations (j) in context i and the number of PEs (k) is N. OP_{ijk} is the probability that operation j in context i is mapped to PE_k. $OP_{ijk} = \{0,1\}$ implies the k^{th} PE in context i is either used or not for operation j, $ST(OP_{ij})$ is the stress time of the j^{th} operation in context i obtained from the stress time maps in the lifetime unaware mapping generation phase. ST_{max_valid} is the accumulated stress time constraint for each PE. k is the location of the k^{th} PE and is the current PE that operation j in context i is mapped to, while k_{orig} is the original PE that operation j in context i was mapped to during the lifetime unaware floorplan. $|k - k_{orig}|$ is the Manhattan displacement distance between PE_k and $PE_{k_{orig}}$.

The objective is to minimize the *displacement*, which serves as an approximation for the delay change, of the utilized PEs relative to the original lifetime unaware floorplan, subject to

ALGORITHM 1: Lifetime-aware re-mapping DSE

```
BList: List of new PE bindings that consider MTTF;
B_l: Unique binding of operations onto PEs: $\{j \rightarrow PE_k\}$;
B_LP: Partial binding of operations onto PEs resulting from LP formulation;
1  /* Step 1: Feasible stress time Budget Minimization */
2  execute equation 3 to get $ST_{max\_opt}$;
3  a set of accumulated stress time bounds: $ST_{max\_valid} = \{ST_{max\_opt}, ST_{max\_l}, \dots, ST_{max\_i}, \dots, ST_{max\_orig}\}$;
4  /* Step 2: Find unique Binding of operations onto PEs */
5  for each $ST_{max\_l}$ in $ST_{max\_valid}$ do
6      $B_{LP}$ = execute equation 2 as LP formulation to get partial floorplan;
7      $B_l$ = execute equation 2 as mixed LP/ILP formulation with $B_{LP}$;
8  end
9  /* Step 3: MTTF vs. delay estimation */
10 $BList$ = Pareto-optimal $\{B_l(MTTF, delay)\}$;
   output : $BList$
```

two constraints: one constraint is that the accumulated stress time for each PE must be less than or equal to a given stress time budget ST_{max_valid}. Another constraint is that in each context i, each operator j is only mapped to one PE. By setting different accumulated stress time constraints (ST_{max_valid}), the proposed formulation automatically finds different floorplans, each with a unique PE displacement vs. MTTF trade off.

$$\begin{aligned} \text{minimize} \quad & displacement = \sum_{i=1}^{c}\sum_{j=1}^{M_i}\sum_{k=1}^{N} OP_{ijk} \times |k - k_{orig}| \\ \text{subject to} \quad & \sum_{i=1}^{c}\sum_{j=1}^{M_i} OP_{ijk} \times ST(OP_{ij}) \leqslant ST_{max\_opt} \\ & \sum_{k=1}^{N} OP_{ijk} = 1 \qquad i \in [1,\dots,c] \\ & \text{if LP then each } OP_{ijk} \in [0,1] \qquad j \in [1,\dots,M_i] \\ & \text{if mixed LP/ILP, linear } OP_{ijk} \in [0,1] \quad k \in [1,\dots,N] \\ & \text{and binary } OP_{ijk} \in \{0,1\} \end{aligned} \tag{2}$$

The main problem with a pure ILP approach is that it does not scale well and we observed that for larger benchmarks, the ILP solver could not find a solution within a reasonable amount of time (5 days). Thus, we propose to relax the problem formulation to a two-step MILP formulation to provide the lifetime aware floorplan efficiently. The first step is to solve equation 2 as an LP formation. In this case we do not force OP_{ijk} to be an integer, but rather a real number between 0 and 1, that is $OP_{ijk} = [0.0, 1.0] \in \mathbb{R}$. We then set those OP_{ijk} values which are very close to 1 to be exactly 1. The LP solver returns a solution within seconds/minutes. To get the entire operator mapping solution, we pre-map the operations to PEs resulting from the previous LP formulation solution and then run equation 2 as an ILP formulation to map the rest of the operations to available PEs, where OP_{ijk} is set back to integer type ($OP_{ijk} = [0, 1]$). The scale of this ILP formulation is then greatly reduced compared to a pure ILP.

Phase 3 : Lifetime-aware re-mapping Design Space Exploration: This phase is divided into three steps highlighted in algorithm 1. *Step 1* minimizes the highest accumulated stress time (ST_{max_opt}) among all PEs while providing a valid floorplan. ST_{max_opt} is the lowest possible stress time and the stress time of the original lifetime unaware design (ST_{max_orig}) is taken as the maximum stress time. For each of a set of stress time limits (ST_{max_valid}) between these extremes (ST_{max_orig} and ST_{max_opt}), in *Step 2* we generate a design with a particular delay vs. MTTF trade-off. Finally, *Step 3* identifies the Pareto optimal delay vs. MTTF trade-offs.

Step 1: Feasible stress time Budget Minimization: The objective of this step is to find the minimum accumulated stress time (ST_{max_opt}) among all PEs on the CGRRA fabric. The solution of the ILP formulation below will lead to a valid PE mapping as follows:

$$\begin{aligned} ST_{max\_opt} \quad & \min_{k} ST_k = \sum_{i=1}^{c}\sum_{j=1}^{M_i} OP_{ijk} \times ST(OP_{ij}) \\ \text{subject to} \quad & \sum_{k=1}^{N} OP_{ijk} = 1 \quad i \in [1,\dots,c], j \in [1,\dots,M_i], k \in [1,\dots,N] \end{aligned} \tag{3}$$

where ST_k is the accumulated stress time of PE_k. In this step, no displacement constraint is set in the formulation.

Step 2: Find unique bindings of operations onto PEs: This step takes as input the ST_{max_valid} values obtained in the previous step (step 1) and uses them as the constraints in the overall two-step MILP formulation as described in equation 2. This process is repeated for each ST_{max_valid}. Thus, the result of this phase is a list of PE bindings ($BList$), where $BList = \{B_l = (\{Op_i \rightarrow PE_k\}, ST_{max\_opt}), \dots, B_l = (\{Op_i \rightarrow PE_k\}, ST_{max\_i}), etc.,\}$ each with a unique minimum PE displacement and ST_{max} value.

Step 3: MTTF vs. delay estimation: This step takes as input the new placements found in the previous step and estimates their MTTF based on equation 1. The result is a set of Pareto-optimal floorplans with unique MTTF vs. delay trade-offs.

V. Experimental Results

In this work we target the Renesas Electronics STP CGRRA with variable fabric size (based on the benchmark used) and use their commercial HLS, Logic Synthesis (LS) and Placement and Routing (PAR) flow called Musketeer [1]. The HLS target frequency is fixed in all cases to 200MHz. The experiments are conducted on an Intel i7-6700 (3.50 GHZ) CPU and 16 GB memory, running CentOS 7.0. HotSpot 6.0 [11] is used for thermal simulation. The ILP solver used in this paper is CPLEX [13] and we use Python 2.7-based PuLP 1.6.1 [14] to call CPLEX to manipulate the solution.

27 Synthesizable C benchmarks from different sources were selected ranging from low, medium to high in aspects of context number, CGRRA fabric size and PE usage rate. Table I shows the features and simulation results for benchmarks (B1-B27) with low, medium and high fabric usage rates, respectively. Columns 5, 9 and 13 report the highest MTTF increase ($\times$) of our proposed lifetime aware mapping method versus the original lifetime unaware mapping tool in Musketeer. The corresponding CPD increase (%) are listed in columns 5, 9 and 13. The last 2 columns list the average displacement per PE and running time of the maximum MTTF case for the benchmarks with high fabric usage rate (B19-27). Finally, Fig. 4 shows the Pareto-optimal trade off curves of

TABLE I: MTTF increase for benchmarks with **low, medium and high** fabric usage rate

		low fabric usage rate				**medium** fabric usage rate				**high** fabric usage rate					
				incr. (x)				incr. (x)				incr. (x)		avg.disp	runtime(s)
context #	fabric size		PE#	MTTF	CPD		PE#	MTTF	CPD		PE#	MTTF	CPD	per PE	
4	4*4	B1	24	1.94	0.00	B10	35	1.67	0.00	B19	52	1.52	0.00	0.58	0.57
4	8*8	B2	79	2.17	0.00	B11	148	1.71	1.00	B20	175	1.56	3.00	1.25	18.02
4	16*16	B3	192	2.28	0.00	B12	451	1.77	0.00	B21	554	1.68	6.00	1.01	998.59
8	4*4	B4	44	2.80	0.00	B13	62	2.30	0.00	B22	87	1.81	5.00	0.64	1.14
8	8*8	B5	142	2.89	0.00	B14	280	2.44	5.00	B23	327	1.91	0.00	0.54	78.90
8	16*16	B6	534	3.01	0.00	B15	1101	2.45	0.00	B24	1521	1.98	0.00	0.21	2112.10
16	4*4	B7	88	3.36	7.00	B16	147	2.75	9.00	B25	193	2.05	0.00	0.47	1.74
16	8*8	B8	259	3.63	0.00	B17	531	2.94	0.00	B26	737	2.21	3.00	0.13	33.47
16	16*16	B9	1011	3.68	1.00	B18	2165	3.04	1.00	B27	3089	2.35	2.00	0.08	3701.35
Avg.				**2.86**	**0.89**			**2.34**	**1.77**			**1.89**	**2.11**		**771.76**

our proposed method for benchmarks with low, medium and high PE fabric utilization. The x-axis is MTTF representing the lifetime of CGRRA and the y-axis is the CPD increase percent. The baseline of our method is the MTTF of the original floorplan generated by Musketeer. It can be observed that our proposed method leads to substantial MTTF increase in all 27 benchmarks, while increasing modestly the CPD in some cases, and none in most of them. The proposed method improves the MTTF by an average of 2.4× while incurring an average CPD increase of only 1.5 percent compared to the original timing-driven lifetime unaware floorplan. Our proposed method also provides an average of 2.34× MTTF increase without any CPD increase which are the cases at the lower left corner of the curves in Fig. 4.

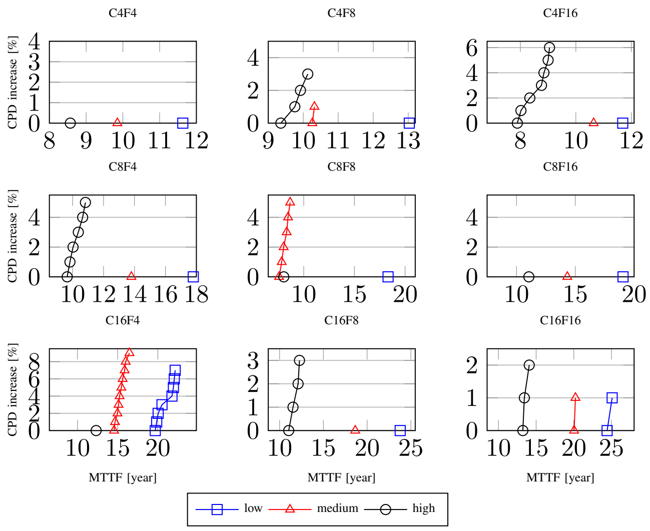

Fig. 4: MTTF vs. CPD increase percentage for benchmarks with low, medium and high PE fabric utilization.

VI. Conclusion

In this work we have presented a reliability-aware floorplanner specifically targeted for CGRRAs that extends the MTTF of these architectures by balancing PE usage while minimizing the average PE displacement. Our floorplanner is based on an efficient mixed linear and integer based programming formulation that binds operations to PEs in the CGRRA to mitigate aging-induced lifetime degradation. The floorplanner is the basis of a design space explorer that generates a variety of configurations which trade-off between MTTF and the minimized average PE displacement. Results show that the MTTF can be improved by an average of 2.4× while incurring an average CPD increase of under 2% (although most benchmarks had no delay increase) compared to the original timing-driven lifetime unaware floorplan.

References

[1] Renesas, "Stp, https://www.renesas.com/us/en/products/power-management/pmic/stp-engine.html," 2018.

[2] R. Vattikonda, W. Wang, and Y. Cao, "Modeling and minimization of pmos nbti effect for robust nanometer design," in *Design Automation Conference, 2006 43rd ACM/IEEE*. IEEE, 2006, pp. 1047–1052.

[3] S. Srinivasan, R. Krishnan, P. Mangalagiri, Y. Xie, V. Narayanan, M. J. Irwin, and K. Sarpatwari, "Toward increasing fpga lifetime," *IEEE Tran. on Dep. and Secure Comp.*, vol. 5, no. 2, pp. 115–127, 2008.

[4] J. Angermeier, D. Ziener, M. Glaß, and J. Teich, "Stress-aware module placement on reconfigurable devices," in *Field Programmable Logic and Applications (FPL), 2011 Int. Conf.* IEEE, 2011, pp. 277–281.

[5] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel, "Mapping on multi/many-core systems: survey of current and emerging trends," in *2013 50th ACM/EDAC/IEEE DAC*. IEEE, 2013, pp. 1–10.

[6] Z. Ghaderi and E. Bozorgzadeh, "Aging-aware high-level physical planning for reconfigurable systems," in *ASP-DAC, 2016 21st Asia and South Pacific*. IEEE, 2016, pp. 631–636.

[7] H. Zhang, M. A. Kochte, E. Schneider, L. Bauer, H.-J. Wunderlich, and J. Henkel, "Strap: Stress-aware placement for aging mitigation in runtime reconfigurable architectures," in *Proceedings of the IEEE/ACM Int. Conf. on CAD*. IEEE Press, 2015, pp. 38–45.

[8] S. S. Sahoo, T. D. Nguyen, B. Veeravalli, and A. Kumar, "Lifetime-aware design methodology for dynamic partially reconfigurable systems," in *Proceedings of the 23rd Asia and South Pacific Design Automation Conference*. IEEE Press, 2018, pp. 393–398.

[9] H. Afzali-Kusha, O. Akbari, M. Kamal, and M. Pedram, "Energy and reliability improvement of voltage-based, clustered, coarse-grain reconfigurable architectures by employing quality-aware mapping," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 3, pp. 480–493, 2018.

[10] J. Gu, S. Yin, and S. Wei, "Stress-aware loops mapping on cgras with considering nbti aging effect," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2017, pp. 1–6.

[11] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "Hotspot: A compact thermal modeling methodology for early-stage vlsi design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 5, pp. 501–513, 2006.

[12] H. Zhang, L. Bauer, M. A. Kochte, E. Schneider, C. Braun, M. E. Imhof, H.-J. Wunderlich, and J. Henkel, "Module diversification: Fault tolerance and aging mitigation for runtime reconfigurable architectures," in *Test Conference (ITC), 2013 IEEE International*. IEEE, 2013, pp. 1–10.

[13] I. I. CPLEX, "V12. 1: Users manual for cplex," *International Business Machines Corporation*, vol. 46, no. 53, p. 157, 2009.

[14] S. Mitchell, M. OSullivan, and I. Dunning, "PuLP: a linear programming toolkit for python," *http://www. optimization-online. org/DB_FILE/2011/09/3178. pdf*, 2011.