Luís Fernando Arcaro

INCREASING THE RELIABILITY AND APPLICABILITY OF MEASUREMENT-BASED PROBABILISTIC TIMING ANALYSIS

Tese submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas para a obtenção do Grau de Doutor em Engenharia de Automação e Sistemas.
Orientador: Prof. Dr. Rômulo Silva de Oliveira

Florianópolis

2019

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Arcaro, Luís Fernando
Increasing the Reliability and Applicability of Measurement-Based Probabilistic Timing Analysis / Luís Fernando Arcaro ; orientador, Rômulo Silva de Oliveira, 2019.
245 p.

Tese (doutorado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós Graduação em Engenharia de Automação e Sistemas, Florianópolis, 2019.

Inclui referências.

1. Engenharia de Automação e Sistemas. 2. Sistemas de Tempo Real. 3. Análise Temporal Probabilística Baseada em Medições. 4. Confiabilidade. 5. Aleatorização Temporal. I. Oliveira, Rômulo Silva de. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia de Automação e Sistemas. III. Título.

Luís Fernando Arcaro

INCREASING THE RELIABILITY AND APPLICABILITY OF MEASUREMENT-BASED PROBABILISTIC TIMING ANALYSIS

Esta Tese foi julgada aprovada para a obtenção do Título de "Doutor em Engenharia de Automação e Sistemas", e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia de Automação e Sistemas.

Florianópolis, 8 de Abril 2019.

[illegible]

Prof. Dr. Rômulo Silva de Oliveira
Orientador

[illegible]

Prof. Dr. Werner Kraus Junior
Coordenador do Programa de Pós-Graduação em Engenharia de Automação e Sistemas

Banca Examinadora:

[illegible]

Prof. Dr. Rômulo Silva de Oliveira
Presidente

[illegible]

Prof. Dr. Sandro Rigo
Universidade Estadual de Campinas (UNICAMP)

[illegible]

Prof. Dr. Luiz Cláudio Villar dos Santos
Universidade Federal de Santa Catarina (UFSC)

[illegible]

Prof. Dr. Eduardo Augusto Bezerra
Universidade Federal de Santa Catarina (UFSC)

À minha família.
Ao Bibinho.

AGRADECIMENTOS

Agradeço inicialmente a meus grandes pais, Lourenço e Izabel, pela vida, pela motivação e pela segurança familiar que permitiram minha dedicação à formação acadêmica. À minha irmã, Katia, pelos cuidados, pelos valiosos ensinamentos, e pela companhia durante toda nossa vida. À minha parceira de toda hora, Vanessa, pela companhia, paciência e compreensão no nosso movimentado dia-a-dia. Ao meu cunhado, Guilherme, por ser membro ativo e presente na nossa família e pela ótima companhia regada a cerveja e café. Aos nossos bichinhos, Mini (Bibi), Blu e Horácio, pela alegria que representam em nossas vidas. Ao meu orientador e exemplar profissional, Rômulo, agradeço grandemente pelo trabalho realizado e pela paciência em me orientar ao longo dos últimos seis anos. À minha irmã de doutorado, Karila, por compartilhar todas as dificuldades e glórias da pós-graduação – foi e continua sendo ótimo trabalhar com você! À minha também irmã de Belém, Rejane, pela grande amizade e pelos valiosos ensinamentos que levarei certamente para a vida toda. Aos amigos Jonatas, Carol e Wilson, pela amizade que permanece forte apesar da distância e dos compromissos. Aos amigos Alexandre, Flávio, Leonardo Assis, Leonardo Martins, Sidney e Thayse Christine, pela companhia diária dentro e fora da universidade e pelas discussões aleatórias regadas a chimarrão e café. Ao hermano colombiano Carlos González Aguilera, pela grande amizade e pelo boné do Cusco Voador. À amiga gaúcha Stephanie, pelos chimarrões e conversas sobre esse e outros gauchismos. Aos amigos e ciclistas incansáveis Carlise e Ruan, pelas *trips* insanas envolvendo grandes distâncias, inclinações, e cercas elétricas. Aos amigos Priscila, Guilherme e Gael, pelos fantásticos cafés com bolo e conversa. Aos demais amigos e colegas do PPGEAS, da Sala dos Doutorandos e além, por serem como uma grande família que adora café, biscoito, cerveja e churrasco. Aos professores do PPGEAS, pelo conhecimento e experiência compartilhados incansavelmente ao longo dos cursos. E, finalmente mas com igual importância, a todas as pessoas que não citei explicitamente, mas que participaram de alguma forma desses seis anos de aprendizado acadêmico, profissional e principalmente existencial. Obrigado!

Luís Fernando Arcaro

RESUMO

Conforme a complexidade das arquiteturas computacionais aumenta para melhorar desempenho ou reduzir custos, o uso de processadores modernos em Sistemas de Tempo Real (STRs) é prejudicado cada vez mais pelo surgimento de efeitos temporais que dificultam a obtenção de limites confiáveis e precisos para os *Worst-Case Execution Times* (*WCETs*) de tarefas. A Análise Temporal Probabilística Baseada em Medições (ATPBM) visa determinar limites probabilísticos de *WCET* (*i.e. pWCETs*) aplicando a Teoria de Valores Extremos (TVE) sobre medições de tempos de execução, e é portanto promissora no tratamento da complexidade de *hardware* no projeto de STRs. Processadores temporalmente aleatorizados foram recentemente propostos para tornar o comportamento temporal de sistemas computacionais mais facilmente analisável através de ferramental probabilístico, e são projetados substituindo informações internas determinísticas ou especulativas por números (pseudo-) aleatórios. A pesquisa cujos resultados são apresentados nesta tese produziu contribuições em duas frentes distintas. Em primeiro lugar, foram propostos e aplicados métodos para avaliar a confiabilidade dos *pWCETs* produzidos pela ATPBM, baseados na coleta de grandes amostras de tempos de execução e na comparação (1) dos *pWCETs* com os maiores tempos de execução observados, e (2) das densidades de excedência dos *pWCETs* com seus valores esperados. Essas avaliações indicaram que modelos probabilísticos da TVE projetados para gerar margens mais precisas podem muitas vezes levar a subestimativas de *pWCETs*, e recomendou-se então que modelos sobrestimadores devem ser utilizados para obter-se *pWCETs* mais confiáveis. Em segundo lugar, avaliou-se a hipótese de que técnicas de escalonamento aleatorizado podem beneficiar a análise temporal de tarefas executadas em *pipelines multithread* através da ATPBM, por levarem os tempos de execução produzidos a atenderem às premissas básicas de aplicabilidade da técnica. Para tal, foram considerados tanto (A) um escalonador puramente aleatório, quanto (B) um escalonador aleatorizado capaz de limitar os efeitos temporais da interferência entre *threads*, sem comprometer sua analisabilidade pela ATPBM, através de um mecanismo de regulação de elegibilidade baseado em créditos.

Palavras-chave: Sistemas de Tempo Real. Análise Temporal Probabilística Baseada em Medições. Confiabilidade. Aleatorização Temporal.

RESUMO EXPANDIDO

Introdução

Sistemas de Tempo Real (STRs) são sistemas computacionais sujeitos tanto a requisitos de natureza lógica quanto de natureza temporal, ou seja, seus resultados precisam não apenas estar logicamente corretos mas devem também ser gerados respeitando prazos (*deadlines*) estritos. Tais sistemas são classificados de acordo com a criticalidade de seus requisitos temporais. A ausência de corretude temporal em STRs críticos pode resultar em consequências catastróficas, tanto do ponto de vista econômico quanto no sentido de poder causar a perda de vidas. Garantias precisam ser fornecidas, portanto, de que seus *deadlines* não serão perdidos – mesmo no pior caso. O *Worst-Case Execution Time* (*WCET*) de tarefas de *software* representa o tempo mais longo que a plataforma-alvo de *hardware* pode possivelmente levar para executá-las. O *WCET* de uma tarefa varia em função de múltiplos fatores que precisam ser levados em conta para a determinação de seu valor exato, o que geralmente prova-se extremamente complexo, ou para estabelecer limites superiores confiáveis para seu valor. Especialmente em relação a STRs críticos, estimativas de *WCET* precisam ser seguras, *i.e.* o valor real nunca deve ser subestimado, e razoavelmente apertadas, *i.e.* sua sobrestimação deve ser minimizada a fim de reduzir o desperdício de recursos. A determinação de limites para *WCETs* é geralmente realizada através de métodos estáticos, que empregam análises conjuntas detalhadas do *software* e do *hardware*, ou baseados em medições, que analisam amostras do tempo efetivamente consumido para executar a tarefa analisada na plataforma-alvo. Este trabalho desenvolve-se no contexto da Análise Temporal Probabilística Baseada em Medições (ATPBM), que visa determinar limites probabilísticos para os *WCETs* de tarefas que compõem STRs. Esses limites, conhecidos como *pWCETs*, são compostos de um valor-limite e de uma probabilidade associada à excedência desse valor em qualquer execução da tarefa. A aplicação da ATPBM baseia-se na análise estatística dos maiores tempos de execução produzidos pelas tarefas, medidos quando de sua execução na plataforma de *hardware* real sob condições adequadas. A principal ferramenta atualmente utilizada pela ATPBM é a Teoria de Valores Extremos (TVE), um ramo da estatística destinado a estimar a probabilidade de eventos extremos raros. Através do ajuste de modelos estatísticos aos maiores valores observados de uma variável associada a

um determinado fenômeno, a TVE é capaz de determinar margens de segurança com probabilidades arbitrariamente baixas de excedência. No contexto da ATPBM, a TVE é promissora para a determinação de estimativas de *pWCET* associadas a probabilidades de excedência comparáveis, ou até mesmo mais baixas, que aquelas associadas a outros tipos de falha que precisam ser levadas em consideração no projeto de STRs críticos (*e.g.* falhas estruturais). A aleatorização temporal em *hardware* foi proposta recentemente como um meio para projetar processadores cujo comportamento temporal é influenciado por leis probabilísticas, podendo assim facilitar a análise temporal de STRs através de métodos baseados em ferramentas estatísticas como a ATPBM. O princípio básico da aleatorização temporal é a substituição de informações tipicamente especulativas utilizadas para tomar ações que influenciam o tempo de execução de tarefas por números (pseudo-)aleatórios. Consequentemente, a aleatorização temporal também desacopla parcialmente o comportamento temporal do *hardware* do histórico de execução, mitigando o surgimento sistemático de padrões patológicos associados a tempos de execução extremos. Apesar de não fornecer garantias quanto à analisabilidade dos tempos de execução produzidos, processadores temporalmente aleatorizados frequentemente mostram-se adequados para análise através da ATPBM.

Objetivos

Esta tese possui dois objetivos: (A) investigar métodos empíricos para evidenciar ou contra-provar a confiabilidade de estimativas de *pWCET* produzidas utilizando a TVE no contexto da ATPBM, e (B) investigar abordagens para o projeto de *pipelines multithread* temporalmente aleatorizados adequados para a aplicação da ATPBM. Mais especificamente, a tese a ser demonstrada em (A) é que a confiabilidade das estimativas de *pWCET* produzidas utilizando a ATPBM pode ser avaliada utilizando grandes amostras de validação e comparando seu comportamento com expectativas em relação à delimitação (*bounding*) dos maiores valores e das densidades de cauda observadas, e em (B) é que aleatorização temporal pode ser utilizada para permitir a utilização de *pipelines multithread* em STRs por produzir tempos de execução adequados à aplicação da ATPBM.

Contribuições

Duas principais contribuições são fornecidas neste trabalho. Em primeiro lugar, foi realizada uma avaliação empírica da confiabilidade de estimativas de *pWCET* produzidas com base na ATPBM através

da TVE utilizando grandes amostras de validação (por exemplo, de tamanho 10^8), considerando diferentes abordagens para aplicação da TVE e realizando um grande conjunto de replicações utilizando diferentes condições e métodos. A confiabilidade das estimativas de *pWCET* foram avaliadas com base na efetiva delimitação (1) dos maiores tempos de execução observados e (2) das densidades de excedência das caudas das distribuições empíricas de tempos de execução observadas em amostras de validação. Em segundo lugar, foi avaliada a hipótese de que técnicas de escalonamento aleatorizado de *threads* podem beneficiar a aplicação da ATPBM em múltiplas tarefas executadas simultaneamente em *pipelines multithread* (neste trabalho foi considerado apenas o caso *dual-thread*). Para isso, foram avaliados tanto (1) um escalonador puramente aleatório, que leva ao atendimento dos requisitos da ATPBM mas que não balanceia atrasos devido a interferências, e (2) um escalonador capaz de limitar os efeitos temporais de pior caso da interferência entre *threads*, utilizando para isso um mecanismo de regulação de elegibilidade baseado em créditos.

Considerações Finais
Esta tese fornece contribuições relevantes no contexto da ATPBM, tanto por avaliar e aumentar a confiabilidade de seus resultados quanto em relação à introdução de aleatorização no nível de *hardware* para facilitar sua aplicação. Em particular, os métodos de avaliação de confiabilidade propostos mostraram que a ATPBM precisa que seus modelos probabilísticos sejam utilizados de forma que as estimativas de *pWCET* potencialmente superestimem (e não apenas estimem) os máximos tempos de execução possivelmente observáveis. Além disso, o trabalho apresentado explorou caminhos originais de pesquisa envolvendo *multithreading*, uma técnica que é frequentemente evitada no projeto de STRs por potencialmente (1) levar a grande pessimismo na estimação de *WCETs* através de métodos estáticos, ou (2) requerer que garantias temporais sejam fornecidas apenas para uma *thread*, cuja execução é geralmente priorizada. Pode-se portanto considerar que o trabalho apresentado fornece contribuições relevantes que aumentam a aplicabilidade confiável da ATPBM, também promovendo portanto sua usabilidade na análise temporal de STRs críticos.

Palavras-chave: Sistemas de Tempo Real. Análise Temporal Probabilística Baseada em Medições. Confiabilidade. Aleatorização Temporal.

ABSTRACT

As the complexity of computer architectures grows in order to improve performance and/or to reduce costs, the use of modern processors in the design of Real-Time Systems (RTSs) is increasingly hampered by the emergence of timing effects that challenge determining reliable and tight bounds for tasks' Worst-Case Execution Times (WCETs). The Measurement-Based Probabilistic Timing Analysis (MBPTA) technique aims determining probabilistic WCET bounds (i.e. pWCETs) by applying Extreme Value Theory (EVT) on tasks' execution time measurements, and is hence promising in handling hardware complexity issues within RTSs' design. Hardware-level time-randomized processors were recently proposed as a means to cause computing systems' timing behaviour to become more easily analysable through probabilistic tools, and are designed replacing deterministic or speculative internal information with (pseudo-)random numbers. The scientific research whose outcomes are presented in this thesis produced contributions on two distinct fronts. In first place, we proposed and applied methods for evaluating the reliability of pWCET estimates produced using MBPTA, based on collecting large execution time samples and then comparing (1) the pWCETs against the largest observed execution times, and (2) pWCETs' exceedance densities against their expected values. These evaluations led us to conclude that EVT probabilistic models intended to yield more precise bounds may often lead to pWCET underestimations, and we hence recommended that upper-bounding models should instead be used for deriving pWCETs with increased reliability. In second place, we evaluated the hypothesis that randomized scheduling techniques can benefit the timing analysis of tasks executed on multithread pipelines through MBPTA, by causing the yielded execution times to meet the technique's basic application requirements. For that, we considered both (A) a scheduler that employs a purely random policy, and (B) a randomized scheduler capable of limiting the timing effects of inter-thread interference, without compromising analysability, by using a credit-based eligibility regulation mechanism.

Keywords: Real-Time Systems. Measurement-Based Probabilistic Timing Analysis. Reliability. Time-Randomization.

LIST OF FIGURES

Figure 1 Block Maxima (BM) approach 42
Figure 2 Peaks Over Threshold (POT) approach 43
Figure 3 Hypothesis tests' decision approaches 49
Figure 4 KS test p-value distribution comparison 49
Figure 5 Quantile plot comparison 50
Figure 6 Platform illustration 64
Figure 7 Raw data histogram 65
Figure 8 I.i.d. tests' p-values 65
Figure 9 BM artefacts 66
Figure 10 POT artefacts 66
Figure 11 pWCET derivation process 67
Figure 12 Synthetic samples' distributions 87
Figure 13 ε probability analysis 92
Figure 14 Real-hardware sample applicability evidence 93
Figure 15 Real-hardware pWCET HWM reliability 95
Figure 16 pWCET HWM reliability for $\xi = -\frac{1}{2}$ 96
Figure 17 pWCET HWM reliability for $\xi = -\frac{1}{4}$ 97
Figure 18 pWCET HWM reliability for $\xi = -\frac{1}{8}$ 98
Figure 19 pWCET HWM reliability for $\xi = 0$ 99
Figure 20 pWCET HWM reliability for $\xi = +\frac{1}{8}$ 100
Figure 21 pWCET HWM reliability for $\xi = +\frac{1}{4}$ 101
Figure 22 pWCET HWM reliability for $\xi = +\frac{1}{2}$ 102
Figure 23 Real-hardware pWCET density reliability 104
Figure 24 pWCET density reliability for $\xi = -\frac{1}{2}$ 106
Figure 25 pWCET density reliability for $\xi = -\frac{1}{4}$ 107
Figure 26 pWCET density reliability for $\xi = -\frac{1}{8}$ 108
Figure 27 pWCET density reliability for $\xi = 0$ 109
Figure 28 pWCET density reliability for $\xi = +\frac{1}{8}$ 110
Figure 29 pWCET density reliability for $\xi = +\frac{1}{4}$ 111
Figure 30 pWCET density reliability for $\xi = +\frac{1}{2}$ 112
Figure 31 EQMAE-based threshold selection plots 114
Figure 32 Simple pipeline design 120

Figure 33 Pipeline design . . . 124
Figure 34 *bsort* PRS maxima analysability analysis . . . 130
Figure 35 *bs* PRS maxima analysability analysis . . . 131
Figure 36 *cnt* PRS maxima analysability analysis . . . 131
Figure 37 *cover* PRS maxima analysability analysis . . . 131
Figure 38 PRS class timing dominance analysis . . . 132
Figure 39 PRS behavioural timing dominance analysis . . . 134
Figure 40 PRS typical scenario slowdown analysis . . . 136
Figure 41 IRS credit consumption traces . . . 141
Figure 42 *bsort* IRS maxima analysability analysis . . . 142
Figure 43 *bs* IRS maxima analysability analysis . . . 142
Figure 44 *cnt* IRS maxima analysability analysis . . . 142
Figure 45 *cover* IRS maxima analysability analysis . . . 143
Figure 46 IRS class timing dominance analysis . . . 144
Figure 47 IRS behavioural timing dominance analysis . . . 145
Figure 48 IRS typical scenario slowdown analysis . . . 147
Figure 49 Applicability evidence for *bsort* on *DPCpArrr* . . . 181
Figure 50 Applicability evidence for *bsort* on *DPArptdm* . . . 182
Figure 51 Applicability evidence for *insertsort* on *DPCpArrr* . . . 183
Figure 52 Applicability evidence for *insertsort* on *DPArptdm* . . . 184
Figure 53 Applicability evidence for *bs* on *DPCpArrr* . . . 185
Figure 54 Applicability evidence for *bs* on *DPArptdm* . . . 186
Figure 55 Applicability evidence for *expint* on *DPCpArrr* . . . 187
Figure 56 Applicability evidence for *expint* on *DPArptdm* . . . 188
Figure 57 Applicability evidence for *fdct* on *DPCpArrr* . . . 189
Figure 58 Applicability evidence for *fdct* on *DPArptdm* . . . 190
Figure 59 Applicability evidence for *crc* on *DPCpArrr* . . . 191
Figure 60 Applicability evidence for *crc* on *DPArptdm* . . . 192
Figure 61 Applicability evidence for *matmult* on *DPCpArrr* . . . 193
Figure 62 Applicability evidence for *matmult* on *DPArptdm* . . . 194
Figure 63 Applicability evidence for *fir* on *DPCpArrr* . . . 195
Figure 64 Applicability evidence for *fir* on *DPArptdm* . . . 196
Figure 65 Applicability evidence for *fibcall* on *DPCpArrr* . . . 197
Figure 66 Applicability evidence for *fibcall* on *DPArptdm* . . . 198

Figure 67 Applicability evidence for *cnt* on *DPCpArrr* 199
Figure 68 Applicability evidence for *cnt* on *DPArptdm* 200
Figure 69 pWCET HWM reliability for *bsort* on *DPCpArrr* 201
Figure 70 pWCET HWM reliability for *bsort* on *DPArptdm* 201
Figure 71 pWCET HWM reliability for *insertsort* on *DPCpArrr* 202
Figure 72 pWCET HWM reliability for *insertsort* on *DPArptdm* 202
Figure 73 pWCET HWM reliability for *bs* on *DPCpArrr* 203
Figure 74 pWCET HWM reliability for *bs* on *DPArptdm* 203
Figure 75 pWCET HWM reliability for *expint* on *DPCpArrr* 204
Figure 76 pWCET HWM reliability for *expint* on *DPArptdm* 204
Figure 77 pWCET HWM reliability for *fdct* on *DPCpArrr* 205
Figure 78 pWCET HWM reliability for *fdct* on *DPArptdm* 205
Figure 79 pWCET HWM reliability for *crc* on *DPCpArrr* 206
Figure 80 pWCET HWM reliability for *crc* on *DPArptdm* 206
Figure 81 pWCET HWM reliability for *matmult* on *DPCpArrr* . 207
Figure 82 pWCET HWM reliability for *matmult* on *DPArptdm* . 207
Figure 83 pWCET HWM reliability for *fir* on *DPCpArrr* 208
Figure 84 pWCET HWM reliability for *fir* on *DPArptdm* 208
Figure 85 pWCET HWM reliability for *fibcall* on *DPCpArrr* 209
Figure 86 pWCET HWM reliability for *fibcall* on *DPArptdm* 209
Figure 87 pWCET HWM reliability for *cnt* on *DPCpArrr* 210
Figure 88 pWCET HWM reliability for *cnt* on *DPArptdm* 210
Figure 89 pWCET density reliability for *bsort* on *DPCpArrr* 211
Figure 90 pWCET density reliability for *bsort* on *DPArptdm* 211
Figure 91 pWCET density reliability for *insertsort* on *DPCpArrr* 212
Figure 92 pWCET density reliability for *insertsort* on *DPArptdm* 212
Figure 93 pWCET density reliability for *bs* on *DPCpArrr* 213
Figure 94 pWCET density reliability for *bs* on *DPArptdm* 213
Figure 95 pWCET density reliability for *expint* on *DPCpArrr* ... 214
Figure 96 pWCET density reliability for *expint* on *DPArptdm* ... 214
Figure 97 pWCET density reliability for *fdct* on *DPCpArrr* 215
Figure 98 pWCET density reliability for *fdct* on *DPArptdm* 215
Figure 99 pWCET density reliability for *crc* on *DPCpArrr* 216
Figure 100 pWCET density reliability for *crc* on *DPArptdm* 216

Figure 101 pWCET density reliability for *matmult* on *DPCpArrr* 217
Figure 102 pWCET density reliability for *matmult* on *DPArptdm* 217
Figure 103 pWCET density reliability for *fir* on *DPCpArrr* 218
Figure 104 pWCET density reliability for *fir* on *DPArptdm* 218
Figure 105 pWCET density reliability for *fibcall* on *DPCpArrr* . . . 219
Figure 106 pWCET density reliability for *fibcall* on *DPArptdm* . . . 219
Figure 107 pWCET density reliability for *cnt* on *DPCpArrr* 220
Figure 108 pWCET density reliability for *cnt* on *DPArptdm* 220
Figure 109 *crc* PRS maxima analysability analysis 223
Figure 110 *expint* PRS maxima analysability analysis 223
Figure 111 *fdct* PRS maxima analysability analysis 223
Figure 112 *fibcall* PRS maxima analysability analysis 224
Figure 113 *fir* PRS maxima analysability analysis 224
Figure 114 *insertsort* PRS maxima analysability analysis 224
Figure 115 *janne_complex* PRS maxima analysability analysis 224
Figure 116 *matmult* PRS maxima analysability analysis 225
Figure 117 *ns* PRS maxima analysability analysis 225
Figure 118 *prime* PRS maxima analysability analysis 225
Figure 119 PRS class timing dominance analysis 226
Figure 120 PRS class timing dominance analysis 227
Figure 121 PRS behavioural timing dominance analysis 228
Figure 122 PRS behavioural timing dominance analysis 229
Figure 123 PRS typical scenario slowdown analysis 230
Figure 124 PRS typical scenario slowdown analysis 231
Figure 125 *crc* IRS maxima analysability analysis 232
Figure 126 *expint* IRS maxima analysability analysis 232
Figure 127 *fdct* IRS maxima analysability analysis 232
Figure 128 *fibcall* IRS maxima analysability analysis 233
Figure 129 *fir* IRS maxima analysability analysis 233
Figure 130 *insertsort* IRS maxima analysability analysis 233
Figure 131 *janne_complex* IRS maxima analysability analysis 233
Figure 132 *matmult* IRS maxima analysability analysis 234
Figure 133 *ns* IRS maxima analysability analysis 234
Figure 134 *prime* IRS maxima analysability analysis 234

Figure 135 IRS class timing dominance analysis 235
Figure 136 IRS class timing dominance analysis 236
Figure 137 IRS behavioural timing dominance analysis 237
Figure 138 IRS behavioural timing dominance analysis 238
Figure 139 IRS typical scenario slowdown analysis 239
Figure 140 IRS typical scenario slowdown analysis 240

LIST OF TABLES

Table 1 MBPTA results ... 68
Table 2 PRS maximum slowdown ... 135
Table 3 PRS typical slowdown ... 137
Table 4 IRS maximum slowdown ... 146
Table 5 IRS typical slowdown ... 148
Table 6 Raw dataset analysis ... 150
Table 7 BM timing analysis ... 151
Table 8 POT timing analysis ... 152
Table 9 Schedulers' hardware complexity/cost ... 153

LIST OF ACRONYMS

AD	Anderson-Darling
ALU	Arithmetic Logic Unit
BM	Block Maxima
BTB	Branch Target Buffer
CAN	Controller Area Network
CASR	Cellular Automata Shift Register
DMA	Direct Memory Access
DRAM	Dynamic Random Access Memory
EDM	Exceedance Density Metric
EQMAE	Estimated Quantiles' Mean Absolute Error
EVT	Extreme Value Theory
FIFO	First-In First-Out
FPU	Floating-Point Unit
FPGA	Field-Programmable Gate Array
FSB	Front-Side Bus
GMLE	Generalized Maximum Likelihood Estimation
GEV	Generalized Extreme Value
GP	Generalized Pareto
IoT	Internet of Things
HWM	High Water Mark
IEEE	Institute of Electrical and Electronics Engineers
I/O	Input/Output
IP	Intellectual Property

IRS	Interference-Regulated Scheduler
ISA	Instruction Set Architecture
KS	Kolmogorov-Smirnov
LB	Ljung-Box
LFSR	Linear Feedback Shift Register
LRU	Least Recently Used
MBDTA	Measurement-Based Deterministic Timing Analysis
MBPTA	Measurement-Based Probabilistic Timing Analysis
MIPS	Microprocessor without Interlocked Pipeline Stages
MLE	Maximum Likelihood Estimation
NoC	Network-on-Chip
PC	Program Counter
PCI	Peripheral Component Interconnect
PRNG	Pseudo-Random Number Generator
POT	Peaks Over Threshold
PRS	Purely Random Scheduler
pWCET	Probabilistic Worst-Case Execution Time
RAM	Random Access Memory
RISC	Reduced Instruction Set Computer
RF	Register File
RR	Round-Robin
RTS	Real-Time System
SDTA	Static Deterministic Timing Analysis
SPTA	Static Probabilistic Timing Analysis
TDM	Time Division Multiplexing

TRNG	True Random Number Generator
WCEP	Worst-Case Execution Path
WCET	Worst-Case Execution Time
WW	Wald-Wolfowitz

CONTENTS

1 INTRODUCTION **33**
1.1 MOTIVATION 37
1.2 OBJECTIVES 38
1.3 CONTRIBUTIONS 39
1.4 DOCUMENT ORGANIZATION 40
2 EXTREME VALUE THEORY **41**
2.1 APPLICATION APPROACHES 41
2.2 MODEL FITTING 44
2.3 BM BLOCK SIZE SELECTION 44
2.4 POT THRESHOLD SELECTION 45
2.5 APPLICABILITY REQUIREMENTS 46
2.5.1 Applicability Statistical Tests 47
2.5.2 Applicability Plots 50
2.5.3 Conclusion 51
3 MEASUREMENT-BASED PROBABILISTIC TIMING ANALYSIS **53**
3.1 APPLICATION 54
3.2 APPLICABILITY REQUIREMENTS 55
3.3 SAMPLE SIZE 57
3.4 MODEL FITNESS 57
3.5 EXECUTION PATHS 58
3.6 TEMPORAL ISOLATION 59
3.7 RELATED WORK 61
3.8 APPLICATION EXAMPLE 63
3.9 CONCLUSION 68
4 TIME-RANDOMIZED PROCESSORS **69**
4.1 RELATED WORK 71
4.2 CACHE MEMORIES 72
4.3 BRANCH PREDICTORS 75
4.4 BUS ARBITERS 76
4.5 NETWORKS-ON-CHIP 77
4.6 RANDOMNESS SOURCES 78
4.7 CONCLUSION 79
5 pWCET RELIABILITY EVALUATION **81**
5.1 RELATED WORK 81
5.2 EXPERIMENT DESIGN 83
5.2.1 Real-Hardware Samples 84

5.2.2 Synthetic Samples 85
5.3 EXPERIMENT OBJECTIVES 87
5.3.1 pWCET HWM Reliability 87
5.3.2 pWCET Density Reliability 87
5.4 THE PROBABILITY OF REJECTING A RELIABLE pWCET 90
5.4.1 HWM-based method 91
5.4.2 EDM-based method 91
5.5 APPLICABILITY EVIDENCE 92
5.6 pWCET HWM RELIABILITY 94
5.6.1 Real-Hardware Samples 94
5.6.2 Synthetic Samples 95
5.7 pWCET DENSITY RELIABILITY 103
5.7.1 Real-Hardware Samples 103
5.7.2 Synthetic Samples 104
5.8 EQMAE-BASED POT THRESHOLD SELECTION 113
5.9 RECOMMENDATIONS 114
5.10 CONCLUSION 117
6 EVALUATING RANDOMIZED SCHEDULING ON MULTITHREAD PIPELINES TO BENEFIT MBPTA 119
6.1 PIPELINING 119
6.2 MULTITHREADING 121
6.3 MULTITHREAD PIPELINE DESIGN 123
6.4 MAXIMUM INTERFERENCE SCENARIO 125
6.5 EVALUATION METHOD 126
6.6 PURELY RANDOM SCHEDULER (PRS) 130
6.6.1 Evaluation 130
6.6.1.1 Maxima Analysability 130
6.6.1.2 Class Timing Dominance 131
6.6.1.3 Behavioural Timing Dominance 133
6.6.1.4 Interference Balancing 133
6.6.1.5 Typical Scenario Slowdown 134
6.7 INTERFERENCE-REGULATED SCHEDULER (IRS) . . 138
6.7.1 Thread States 138
6.7.2 Inter-Thread Interference Detection 138
6.7.3 Credit-Based Schedulability Regulator 139
6.7.4 Evaluation 141
6.7.4.1 Maxima Analysability 142
6.7.4.2 Class Timing Dominance 143
6.7.4.3 Behavioural Timing Dominance 143

6.7.4.4 Interference Balancing 144
6.7.4.5 Typical Scenario Slowdown 146
6.8 TIMING ANALYSIS COMPARISON 148
6.9 IMPLEMENTATION EFFORT 153
6.10 CONCLUSION 153
7 FINAL REMARKS 155
7.1 GENERAL CONSIDERATIONS 157
7.2 PUBLICATIONS. 159
7.3 FUTURE WORK 159
7.4 ACKNOWLEDGEMENT 160
REFERENCES 161
APPENDIX A – pWCET Reliability Evaluation . . 181
APPENDIX B – Evaluating Randomized Scheduling on Multithread Pipelines to Benefit MBPTA 223
APPENDIX C – Experimentation Platforms 243

1 INTRODUCTION

Real-Time Systems (RTSs) are computer systems that are subject both to logical and temporal requirements, which means the produced results must not only be correct from a logical point of view but must also be generated at the correct time. Applications with real-time requirements are becoming increasingly common and greatly vary in size, complexity and criticality, ranging from the simple embedded controllers for household appliances to complex and critical avionics systems, for example. RTSs are classified according to the criticality of their temporal requirements. In critical or *hard real-time systems*, the lack of timing correctness may result in catastrophic consequences, both from the economic point of view and in the sense of potentially causing the loss of life. In non-critical or *soft real-time systems* the temporal requirements describe the desired behaviour, but if not met they do not invalidate the results nor have catastrophic consequences, although the application's utility is significantly reduced (LIU, 2000).

The temporal requirements to which RTSs are subject are expressed in terms of the deadlines in which the results must be generated. For critical RTSs it is necessary to build strong evidence that these deadlines are not missed, and for this purpose schedulability tests are often employed that demonstrate they are met even in the worst-case scenario. These tests are based on temporal parameters that are either imposed upon the tasks by the operation context, for example their periods T_i and deadlines D_i, or that must be derived from the software and hardware that implement them, such as their maximum execution times C_i – which are estimated considering tasks are executed continuously and exclusively on a single-core processor –, and their response times R_i – determined by taking into account interference such as the concurrent execution of other tasks, the use of resource locks (e.g. semaphores), and the occurrence of release jitter (e.g. due to disabled interrupts) (LIU, 2000; WILHELM et al., 2008).

The C_i parameters, also known as the tasks' Worst-Case Execution Times (WCETs), represent the longest time possibly taken by the target hardware platform to execute the software that implements them. A task's WCET varies depending on a multitude of factors that must be taken into account for the determination of its exact value, which can be extremely complex, or for establishing reliable bounds for it. Especially when related to critical RTSs, the WCET estimates must be safe, i.e. the real value must never be underestimated, and should be tight, i.e. its overestimation must be minimized in order to reduce resource wasting. The methods employed in deriving these estimates must be (1) sound, to allow providing reliable guarantees, (2) efficient, to be useful in practical environments, and (3) as precise as possible regarding the yielded results. The determination of WCET bounds is usually performed through methods that fit one of the following classes, although hybrid approaches are also increasingly common (LIU, 2000; WILHELM et al., 2008, 2009; ABELLA et al., 2015):

- **Static methods** are based on a detailed analysis of the task's code taking into account the architecture of the hardware platform in which it will be executed, thus generally leading to large efforts and/or computational costs but also to results known to be safe (WILHELM et al., 2008; CAZORLA et al., 2013a).
  - **Deterministic**: Static Deterministic Timing Analysis (SDTA) comprises the traditional static timing analysis methods, in which WCET bounds are derived by jointly analysing the task's code and an abstract but temporally trustworthy model of the hardware on which it will be executed. Such methods typically provide safe results by considering or being conservative regarding every possible execution condition, but (1) hardware model abstractions that are often necessary to ensure computational feasibility can lead to large WCET overestimations, and (2) even small changes in the processor architecture may require large efforts to properly adjust the hardware models employed in the timing analysis. Moreover, such methods generally lack composability characteristics needed for handling large systems that use modern processors (WILHELM et al., 2008).
  - **Probabilistic**: Static Probabilistic Timing Analysis (SPTA) determines probability density functions for timing events of the underlying hardware platform through a careful analysis of their behaviour, and progressively combines them through convolution operations in order to obtain upper-bounding densities for the execution of instruction sequences. This leads to reliable results, since the global density function is known to be valid as long as the local ones are, but potentially leads to large computational efforts for the analysis of complex applications, even if executed on simple processor architectures (ABELLA et al., 2014).

- **Measurement-based methods** perform the analysis of measurements of the task's execution times taken on the target hardware platform, which significantly reduces the hardware architecture analysis efforts but requires the determination of safety margins to account for possibly unmeasured effects.
  - **Deterministic**: Measurement-Based Deterministic Timing Analysis (MBDTA) analyses the task's code and execution environment for determining the inputs that are expected to exercise the Worst-Case Execution Paths (WCEPs), i.e. execution paths that are candidate to produce the WCET. Measurements are then taken on the target hardware to obtain these paths' execution times, whose low variability is expected to provide reasonable confidence that their real WCETs are not far from the measured values. Since the conditions for the occurrence of the task's real WCET are generally very hard to predict and reproduce, margins typically still need to be added to the maximum measured time in order to attain higher safety in relation to unwitnessed timing effects. On top of that, acceleration hardware elements capable of inducing worst-case timing effects that are hardly observable (e.g. cache memories) must be avoided or used with care (ABELLA et al., 2014).
  - **Probabilistic**: Measurement-Based Probabilistic Timing Analysis (MBPTA) produces probabilistic WCET estimates, i.e. WCET estimates associated to non-null but sufficiently small exceedance probabilities, by fitting statistical models to measurements of the analysed task's maximum execution times (CUCU-GROSJEAN et al., 2012). Such methods can in principle produce bounds with confidence levels that can be high enough even for systems requiring certification, but for that proper evidence must be supplied that the task's measured execution times (1) are representative regarding the task's worst possible timing behaviour, (2) meet the requirements of the employed statistical tools, and (3) present maxima that in fact adhere to the probability distributions used. This poses strict requirements both on the measurement collection process and on the timing behaviour of the hardware platform used (COLES, 2001; CAZORLA et al., 2016; KOSMIDIS et al., 2016).

This work focuses on the Measurement-Based Probabilistic Timing Analysis (MBPTA) technique, which targets determining probabilistic bounds for the WCETs of tasks that compose RTSs. These bounds, known as Probabilistic Worst-Case Execution Times (pWCETs), are composed of both a limiting value and an associated probability that the value is exceeded at any individual execution of the task. The application of MBPTA is based on the statistical analysis of the tasks' maximum execution times, measured while they are executed on the real target hardware platform under carefully determined conditions (CAZORLA et al., 2013a, 2016). The main tool currently employed by MBPTA is Extreme Value Theory (EVT), a branch of statistics designed to estimate the probability of rare extreme events. Through the adjustment of statistical models to the largest values observed for the outcomes of a target phenomenon, EVT is capable of determining values expected to be exceeded with a maximum probability that can (in principle) be set to arbitrarily low values (COLES, 2001; CUCU-GROSJEAN et al., 2012). Within MBPTA, EVT is promising in enabling the determination of pWCET estimates associated with exceedance probabilities comparable, or even lower, than those associated to other kinds of failures that must be considered in designing critical RTSs (e.g. structural failures) (CAZORLA et al., 2016).

Time-randomization at hardware level was recently proposed as a means for designing processors whose timing behaviour is influenced by probabilistic laws, hence potentially improving RTSs' timing analysability through methods based on statistical frameworks – such as MBPTA. The main principle of time-randomization is the replacement of speculative information typically employed in taking actions that influence execution times with (pseudo-)random numbers. Consequently, time-randomization also partially decouples hardware elements' timing behaviour from execution history, mitigating the systematic emergence of pathological patterns that could lead to extreme execution times (TRILLA et al., 2017b; AGIRRE et al., 2018). Time-randomization have been recently applied in related work, e.g., for designing cache memories (KOSMIDIS et al., 2013a), bus arbiters (JALLE et al., 2014) and Networks-on-Chip (NoCs) (SLIJEPCEVIC et al., 2016, 2017a). Despite no guarantees on execution times' analysability can be effectively provided (LIMA; DIAS; BARROS, 2016), time-randomized processors often prove suitable to be used in the context of MBPTA (CAZORLA et al., 2013a; KOSMIDIS et al., 2016; CAZORLA et al., 2016).

The determination of strict bounds for the WCETs of RTSs' tasks, with the objective of guaranteeing that their timing constraints are met, is becoming increasingly challenging as computer architectures evolve. This is so either due to the large effort and complexity of modelling modern processors' constructive details for applying static methods (WILHELM et al., 2008) or due to difficulties in reliably associating variable execution times with tasks' worst-case behaviour through measurement-based techniques (KOSMIDIS et al., 2016). At the same time, many modern applications – such as autonomous cars and devices composing the so-called Internet of Things (IoT) – are emerging which tend to increase the demand for computer architectures capable of delivering processing capacity with both scalability and affordable timing analysability characteristics. In this context, the associated application of MBPTA and of hardware-level time-randomization techniques is promising in enabling the timing analysis of RTSs' tasks executed on complex computer processors, for which traditional methods would potentially produce pessimistic results, by abstracting constructive details of the underlying hardware (KOSMIDIS et al., 2016).

1.1 MOTIVATION

Static WCET analysis techniques have known limitations regarding the complexity of processor architectures that can be handled in feasible time, for inherently lacking of composability characteristics. The continuous introduction of increasingly complex acceleration hardware elements in processors, mainly for improving performance, is resulting in architectures which are far from being analysable through static methods. Even a small set of such elements can cause static timing analysis to become intractable, due both to the complexity of modelling their behaviour (especially when different elements are combined) and to the computational efforts demanded by the analysis. The introduction of multi-core processors have especially contributed in making static analysis infeasible in modern architectures: the existence of shared hardware elements and the uncertainty regarding the behaviour of tasks lead to a situation in which either (1) unrealistically pessimistic cases must be assumed, inducing pessimism that can be large enough to negate a significant fraction of the processing capacity (NÉLIS; YOMSI; PINHO, 2016; KIM et al., 2016), or (2) all possibilities must be tested, which easily proves intractable (WILHELM et al., 2008; LIU; REINEKE; LEE, 2010; CULLMANN et al., 2010).

At the same time, industry has an increasing demand for WCET analysis, which is often met by either (1) employing simpler architectures to which timing analysis is safe and feasible (e.g. for critical systems), or (2) relying on measurements taken on a "bad-case" environment added with safety margins defined based solely on experience (e.g. for non-critical systems) (CAZORLA et al., 2016).

There is a strong tendency that the market of processors targeting RTSs will witness a fast demand increase during the next years, which is especially true assuming the autonomous cars' and IoT devices' markets keep growing. Whenever this speculation proves right, so will grow the demand for increased processing capacity, and for affordable and reliable timing analysis of tasks executed on such processors (KINNAN, 2009; NÉLIS et al., 2014; SAIDI et al., 2015). In this scenario, the timing analysis solutions currently employed in the industry tend to prove not applicable, since (1) simple processors generally do not allow large sets of tasks being integrated into single hardware platforms, and (2) the pessimism of multi-/many-core processors' analysis through traditional methods tends to prove unacceptable (KIM et al., 2016).

These trends point out that, in a near future, complex hardware platforms will be necessary for executing modern RTSs in order to cope with their increasing processing demands, and that therefore MBPTA approaches will possibly play an important role in enabling industry to derive WCET bounds in a safe and cost-effective manner. Still, there are many open questions associated with the fundamental requirements that must be met in collecting and analysing measurements for applying MBPTA, and especially regarding how reliable are the pWCET estimates it is capable of yielding. Time-randomization is a candidate technique to leverage timing analysability on increasingly complex processors, for being promising in benefiting MBPTA application.

Considering the mentioned open questions and design opportunities, this thesis tackles MBPTA-related issues in two fronts: (A) performing empirical evaluations of the reliability of pWCET estimates derived through MBPTA, and (B) evaluating the suitability for MBPTA of processors equipped with randomly-scheduled multithread pipelines.

1.2 OBJECTIVES

The objectives of this thesis are twofold: (A) investigating empirical methods for evidencing or counter-proving the reliability of pWCET estimates produced using EVT within MBPTA, and (B) investigating approaches for designing time-randomized multithread pipelines suitable for the application of MBPTA. More specifically, the thesis to be demonstrated within (A) is that the reliability of pWCET estimates produced using MBPTA can be evaluated using large validation samples and comparing their behaviour against expectations on the bounding of maximum values and of tail densities, and within (B) is that scheduling-level time-randomization can be employed to enable the use of multithread pipelines on RTSs – a design often regarded as harmful to static timing analysis – by causing yielded execution times to meet the applicability requirements of MBPTA.

1.3 CONTRIBUTIONS

The main contributions provided in this work are twofold:

Firstly, we perform an empirical evaluation of the reliability of pWCET estimates produced based on MBPTA through EVT using large validation samples (e.g. of size 10^8), considering distinct EVT application approaches and performing a large set of replications using different conditions and methods. We evaluate pWCET estimates' reliability based on the effective upper-bounding of large validation samples' (1) maximum observed execution times and (2) densities of execution time empirical distributions' tails. These contributions are covered in the following published papers:

SILVA, K. P.; ARCARO, L. F.; OLIVEIRA, R. S. de. On Using GEV or Gumbel Models when Applying EVT for Probabilistic WCET Estimation. In: **Real-Time Systems Symposium 2017 (RTSS'17)**. IEEE, 2017. p. 220–230.

ARCARO, L. F.; SILVA, K. P.; OLIVEIRA, R. S. de. On the Reliability and Tightness of GP and Exponential Models for Probabilistic WCET Estimation. **ACM Transactions on Design Automation of Electronic Systems (TODAES)**, ACM, v. 23, p. 39:1–39:27, 2018.

ARCARO, L. F.; SILVA, K. P.; OLIVEIRA, R. S. de. A Reliability Evaluation Method for Probabilistic WCET Estimates based on the Comparison of Empirical Exceedance Densities. In: **Brazilian Symposium on Computing Systems Engineering 2018 (SBESC'18) – Work-in-Progress**. IEEE, 2018.

Secondly, we evaluate the hypothesis that randomized thread scheduling techniques can be used to benefit the application of MBPTA to multiple tasks simultaneously executed on multithread pipelines (in this work we consider only the dual-thread case). For that, we evaluated both (1) a purely random scheduler, that leads the MBPTA basic requirements to be met but that does not balance delays due to interference, and (2) an interference-regulated scheduler capable of limiting the worst-case timing effects of inter-thread interference, by employing a credit-based eligibility regulation mechanism. The following paper was submitted covering these contributions:

ARCARO, L. F.; SILVA, K. P.; OLIVEIRA, R. S. de. On Using Randomized Scheduling on Multithread Pipelines for Benefiting Probabilistic Timing Analysis. **ACM Transactions on Embedded Computing Systems (TECS)**, ACM. Submitted on January 14, 2019.

1.4 DOCUMENT ORGANIZATION

The rest of this document is organized as follows. Chapter 2 introduces EVT, the statistical framework that provides the theoretical foundation used in state-of-the-art MBPTA. Chapter 3 presents details on MBPTA and its applicability, together with a step-by-step example of how it is used to obtain pWCET estimates. Chapter 4 deepens the related concepts and presents state-of-the-art hardware elements used in the design of time-randomized processors. Chapter 5 presents our first major contribution, that consists of empirical reliability evaluations of pWCET estimates produced using MBPTA. Chapter 6 introduces our second major contribution, an evaluation of a processor that employs a randomly-scheduled multithread pipeline for benefiting the application of MBPTA. Finally, Chapter 7 presents final remarks associated with the outcomes of the research presented in this thesis.

Appendixes A and B contain additional plots associated with the evaluations presented in Chapters 5 and 6, respectively. Details of the hardware elements that were developed during the herein described research are presented in Appendix C.

2 EXTREME VALUE THEORY

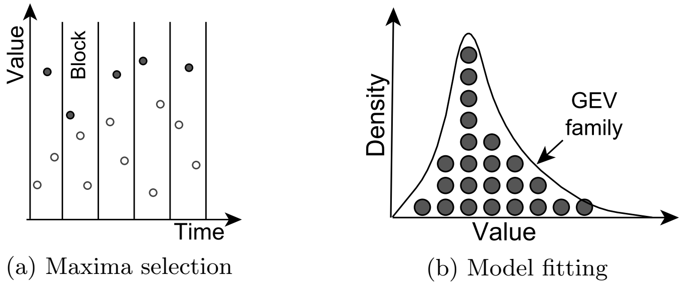
The Extreme Value Theory (EVT) statistics branch was initially designed for predicting from a probabilistic point of view the occurrence of unusual extreme events, by modelling the behaviour of the highest deviations observed for measurements associated to phenomena of interest (COLES, 2001). Throughout the years it has been applied in a variety of areas in which environmental variability is inherent and extreme events are expected to be rare but can have significant or even catastrophic impact. Among them we highlight insurance, in which relatively rare events can cause large losses, and civil engineering, where designs must resist forces that are unpredictable in the long term. In such areas, variability is known to exist, generally cannot be controlled, and determining safety margins is a necessary but challenging task, since information taken from the typical environment is not necessarily representative regarding extreme cases' deviations (COLES, 2001; BEIRLANT et al., 2004; HAAN; FERREIRA, 2006). There is a relatively common criticism on EVT with respect to the fact it extrapolates information obtained from random variables' observed behaviour to estimate the probability of unobserved events. Such criticism is yet not easily defeatable, but (1) areas in which EVT is typically employed require such extrapolations to be made, and (2) there are no alternative frameworks available yet which are capable of replacing EVT based on stronger arguments (COLES, 2001). This chapter provides an overview of EVT's key aspects. Please refer to (COLES, 2001; HAAN; FERREIRA, 2006) for further information.

2.1 APPLICATION APPROACHES

The application of EVT is based on fitting extreme value probability distributions to the maximum values observed in samples of the variable to be analysed (COLES, 2001). Two approaches can mainly be used for choosing the maximum observed values that are effectively analysed through EVT, which are known as the Block Maxima (BM) and the Peaks Over Threshold (POT) approaches.

The Block Maxima (BM) approach (HANSEN; HISSAM; MORENO, 2009; GUMBEL, 2012) is based on the Fisher-Tippett-Gnedenko theorem (FISHER; TIPPETT, 1928), which states that the maximum value of a sample may only converge in distribution to one of *Fréchet* (heavy upper tail), *Gumbel* (exponentially decreasing upper tail), or *Weibull* (bounded upper tail) extreme value distribution families, regardless of the source population distribution (LU et al., 2012). Therefore, as depicted in Figure 1, when BM is employed the collected data sample is divided into blocks (i.e. sub-samples) containing b measurements each, from which only the maximum observed value is kept for analysis' next steps. We highlight that the convergence of block maxima to the mentioned distributions is not guaranteed to be observable, and thus adherence evidence must always be supplied for supporting the EVT applicability argument (LIMA; DIAS; BARROS, 2016). Moreover, the task of determining the block size (b) to be used is not straightforward, and can prove critical in obtaining maxima that adhere to EVT models (LIMA; DIAS; BARROS, 2016; ABELLA et al., 2017). A decision on which of the *Fréchet*, *Gumbel* or *Weibull* distributions better describe the observed blocks' maxima distribution must also be taken, whereas one must either know *a priori* which of them will better fit the analysed dataset – which closely depends on the application scenario –, or try fitting all three and choose the one that best adjusts (COLES, 2001). The Generalized Extreme Value (GEV) distribution (MISES, 1936), whose cumulative distribution function is shown in Equation 2.1, combines the three families into a single one. It has three parameters (location μ, scale σ and shape ξ), and is capable of behaving exactly as the original ones depending on the value of its shape parameter, hence simplifying the fitting process by avoiding the need to manually choosing the most suitable of the three models. On top of that, GEV allows uncertainty regarding which model is more suitable to represent a certain dataset to be expressed as confidence intervals for its shape parameter's estimates (COLES, 2001; FARANDA et al., 2011; LIMA; DIAS; BARROS, 2016).

Figure 1: Block Maxima (BM) approach

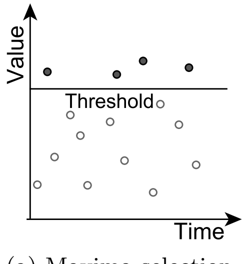


(a) Maxima selection (b) Model fitting

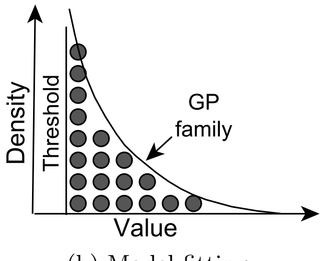
$$GEV_{CDF} = \begin{cases} exp(-(1+\xi\frac{x-\mu}{\sigma})^{-1/\xi}) & \text{if } \xi \neq 0, \\ exp(-exp(-\frac{x-\mu}{\sigma})) & \text{if } \xi = 0. \end{cases} \tag{2.1}$$

On the other hand, the Peaks Over Threshold (POT) (BEIRLANT et al., 2004) method has its foundations on the Pickands-Balkema-de Haan theorem (BALKEMA; HAAN, 1974), which states that the behaviour of a sample's values that exceed a large threshold τ can be well approximated through the Generalized Pareto (GP) distribution (PICKANDS, 1975). As illustrated in Figure 2, when the POT approach is used a value for the threshold τ must be carefully chosen, and only observed values that exceed this threshold are retained to be used in the analysis. The GP model, whose cumulative distribution function is shown in Equation 2.2, is capable of modelling different distribution tails depending on the values assigned to its scale (σ) and shape (ξ) parameters. Negative shape values lead GP to represent bounded tails (e.g. that have a maximum value), while positive ones cause it to model heavy asymptotic tails (e.g. whose density decreases polynomially). The Exponential distribution is a special case of GP with shape $\xi = 0$, and models unbounded tails whose asymptotic density decreases exponentially (ABELLA et al., 2017). Similarly to GEV within BM, confidence intervals on the GP shape parameter enable expressing uncertainty regarding the value that best represents the modelled tail's behaviour. Also analogously to BM with respect to block sizes, different threshold values (τ) can lead to maxima with distinct model adherence characteristics (SANTINELLI et al., 2014).

Figure 2: Peaks Over Threshold (POT) approach



(a) Maxima selection



(b) Model fitting

$$GP_{CDF} = \begin{cases} 1 - \left(1 + \xi \frac{x-\tau}{\sigma}\right)^{-1/\xi} & \text{if } \xi \neq 0, \\ 1 - exp(-\frac{x-\tau}{\sigma}) & \text{if } \xi = 0. \end{cases} \tag{2.2}$$

2.2 MODEL FITTING

Several methods are available for fitting maxima distributions to collected data (GILLELAND; RIBATET; STEPHENSON, 2013), among which we highlight Quantile regression (HANSEN; HISSAM; MORENO, 2009), Maximum Likelihood Estimation (MLE) (COLES; DIXON, 1999; COLES, 2001; EDGAR, 2002), Generalized Maximum Likelihood Estimation (GMLE) (MARTINS; STEDINGER, 2000), and L-moments (HOSKING, 1990). The mentioned fitting approaches present different computational costs, for instance quantile regression is based on fast numeric expressions (EDGAR, 2002) and MLE employs a relatively fast optimization process, while L-moments requires estimating parameters' confidence intervals through a bootstrap approach that may take long to complete (MARTINS; STEDINGER, 2000). They also have different applicability characteristics, for example quantile regression is only applicable to the Gumbel distribution (HANSEN; HISSAM; MORENO, 2009), MLE is known to have convergence problems when $\xi < -0.5$ and cannot be used when $\xi < -1$ (SMITH, 1985), and GMLE was shown to perform better than L-moments and MLE only for shape values $-0.4 \leq \xi \leq 0$ (MARTINS; STEDINGER, 2000). The results presented in this thesis were mainly produced using the L-moments method for fitting the GEV and GP distributions, and the MLE method for fitting the Gumbel and Exponential distributions. However, most of the experiments performed throughout our work were replicated using other methods, in order to support the generality of the drawn conclusions.

2.3 BM BLOCK SIZE SELECTION

While using the BM approach for applying EVT, a critical decision that must be taken is the size of the blocks to be used (SANTINELLI et al., 2014). Small block sizes produce relatively large maxima samples, possibly leading to higher representativeness at this level, but with each block containing only several measurements the tail modelling can be hampered due to convergence to the raw sample. On the other hand, large blocks produce smaller maxima samples, which can poorly represent the underlying population, but with each block containing numerous measurements the population maxima is potentially better characterized. Particularly, the employed block size directly affects the tightness (pessimism) of the produced safety margins and the capability of rare events being properly characterized by EVT

(SANTINELLI et al., 2014). Related works suggest using an iterative approach for determining the block size based on the results of goodness-of-fit tests, for instance by starting with the full sample (i.e. $b = 1$) and incrementing it until tests pass (i.e. good fitting is achieved) (LU et al., 2011), or by starting with e.g. $b = 100$ and doubling it until test results are accepted (HANSEN; HISSAM; MORENO, 2009). The results we present in this thesis were mainly produced using measurement blocks of size 50, but most of the experiments we have performed were replicated using larger block sizes (i.e. 100, 200 and 400), in order to support the generality of the achieved conclusions.

2.4 POT THRESHOLD SELECTION

When the POT approach is employed within EVT it becomes necessary determining the threshold τ whose exceeding peak values are to be fitted to the GP distribution. This is a critical decision, since different thresholds can lead to adjusted models that largely differ in terms of how well the tail of the distribution is represented and, consequently, can also lead to statistical inference outcomes that disagree by significant amounts (LIU; BEHNAM; NOLTE, 2013; ABELLA et al., 2017; SANTINELLI; GUET; MORIO, 2017). The POT threshold is determined in (LIU; BEHNAM; NOLTE, 2013) through an analysis of stability based on (1) the estimates of the GP distribution shape for a set of candidate thresholds, and (2) the shape of the GEV distribution fitted to the measurements' maxima selected through the BM approach. The same task is performed in (SANTINELLI; GUET; MORIO, 2017) by maximizing confidence levels associated to the p-values produced by statistical hypothesis tests of (A) model matching through the Cramer von Mises test (LAIO, 2004) and (B) of long-range independence or extremal independence (EMBRECHTS; KLÜPPELBERG; MIKOSCH, 2013). In (ABELLA et al., 2017) the threshold is selected such that its exceeding tail is likely to present shape equal or lower than zero, based on a coefficient of variation analysis, being therefore reliably upper-bounded through the Exponential model. Several other threshold selection methods are available, as described in (SCARROTT; MACDONALD, 2012).

In this work we evaluate an alternative approach, in which we select the threshold value that minimizes – in relation to the other candidate thresholds – the Estimated Quantiles' Mean Absolute Error (EQMAE) (WILLMOTT; MATSUURA, 2005) of the GP model fitted to the exceeding peaks. The EQMAE is given by $\frac{1}{n}\sum|\hat{x_i} - x_i|$, where x_i

and $\hat{x_i}$ are the n values that exceed the considered threshold and the n quantiles estimated through the GP model fitted to the exceeding peaks, respectively. Similarly to (SANTINELLI; GUET; MORIO, 2017), we only consider threshold candidates between the sample's 60% and 99% quantiles, for both (1) restricting the search to the sample's tail and (2) avoiding too high thresholds being chosen. Through this approach we are effectively selecting, based on a quantitative criterion, the adjusted model that best approximates the observed data tail's quantiles in comparison to other threshold candidates. Since the main argument on using the GP model for applying POT refers to best representing the sample's tail shape (SANTINELLI; GUET; MORIO, 2017) and EVT in fact targets deriving estimates on distribution tails' quantiles, we consider the evaluated selection approach is prone to adequately fit the purpose.

2.5 APPLICABILITY REQUIREMENTS

EVT applicability is subject to a set of requirements to which proper meeting evidence must be supplied in order to support the reliability of the yielded results. The most basic assumption EVT makes is that the outcomes of the variable of interest associated to the analysed phenomenon can be deemed independent and identically distributed. In other words, the observed values must (1) not strongly influence the probability or enable predicting the occurrence of other outcomes, and (2) adhere all to a same distribution of probabilities (COLES, 2001; KOSMIDIS et al., 2014). The independence requirement was relaxed in (LEADBETTER; LINDGREN; ROOTZÉN, 1983), by showing that dependence between variables can be admitted – to a certain extent – as long as stability and stationarity characteristics are consistently observable, i.e. as long as no strong changes in mean and variance are observed along measurements (COLES, 2001). A further requirement is that the environment in which the analysed variable's measurements are collected must be maximally representative regarding the conditions in which extreme values are expected to be produced. This requirement can be considered unmeaningful in certain application contexts, such as sea level prediction, but meeting it can prove essential and quite challenging in fields such as the one addressed in this thesis (KOSMIDIS et al., 2016). It imposes that any controllable factors influencing the analysed variable's outcomes being set such that extreme events become maximally likely to be observed during the sampling process (ABELLA et al., 2014). More details on the specific context in which EVT is applied throughout this work, i.e. for the determination of bounds for the Worst-Case Execution Times (WCETs) of Real-Time Systems (RTSs)' tasks, will be covered in detail in Chapter 3.

2.5.1 Applicability Statistical Tests

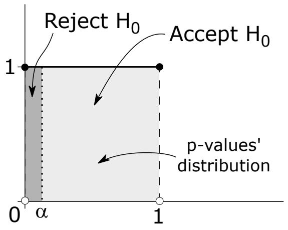
The mentioned EVT requirements demand proper statistical tests being performed in order to evidence that the collected data do not present characteristics that harm its basic assumptions. Statistical hypothesis tests are based on a null hypothesis H_0, which is assumed to be true unless sufficient evidence is found to refute it, and an opposite alternative hypothesis H_1, which is accepted if and only if H_0 is rejected through proper evidence. Hypothesis tests are subject to type I and type II errors, associated respectively to false negative and false positive results regarding the null hypothesis. A statistical hypothesis test produces a p-value that is associated with the probability of its statistical summary yielding a result that is as far or further from the expected assuming H_0 holds as that obtained in the collected sample (a type I error) (WASSERSTEIN; LAZAR, 2016). Such errors are typically controlled through the determination of a limit α to the probability of false negatives (type I errors), known as significance level, whose value is typically kept between 0.05 and 0.01 (ABELLA et al., 2014). As illustrated in Figure 3(a), by comparing a test's yielded p-value with α one can take the decision of rejecting the null hypothesis is $p < \alpha$ or not rejecting it if $p \geq \alpha$, with a confidence level given by $\gamma = 1 - \alpha$ that is hence typically between 95% and 99%.

Measured values' independence can be evidenced using the Wald-Wolfowitz (WW) and the Ljung-Box (LB) statistical tests. The WW or *runs* test, which tests the observed values' randomness null hypothesis, classifies the observed values as (H)igher or (L)ower than the sample median and then examines the normality of the lengths of the "runs" (i.e. H or L contiguous sequences). The application of the test is expected to yield p-values higher than α, thus indicating that there is no strong evidence that the data is generated by a non-random process (SANTINELLI et al., 2014; SLIJEPCEVIC et al., 2014). The LB test has as H_0 the absence of non-null correlations (trends) between observations, and is based on the detection of autocorrelations between each observed value and an arbitrary number of following measurements in a time series (i.e. lags). Its application is therefore also expected to produce p-values higher than the target significance level, indicating that there is no strong evidence that the collected measurements present dependences (LJUNG; BOX, 1978; SANTINELLI et al., 2014; KOSMIDIS et al., 2016). Similarly, identical distribution can be evidenced through the Kolmogorov-Smirnov (KS) and the k-sample Anderson-Darling (AD) statistical tests. The KS test has the null hypothesis that the observations follow the same distribution, and produces a p-value that is related to the distance between the empirical distributions of the assessed samples. Therefore, its application is expected to produce p-values higher than α, indicating that there is no strong evidence that the tested samples were drawn from different distributions (ABELLA et al., 2014; KOSMIDIS et al., 2016). The k-sample AD test is available in two versions that mainly differ by the employed empirical distribution function: one adjusts for possibly different sample sizes and the second focuses on tail differences. Similarly to the KS test it also tests the identical distribution hypothesis, and its application is expected to produce p-values higher than α (SCHOLZ; STEPHENS, 1987).

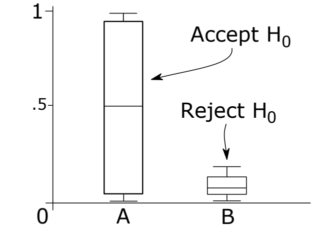
As mentioned, statistical tests can lead to erroneous conclusions with a certain probability, which is typically controlled through the use of a significance level α. Erroneous results can occur due to the variability that is inherent to the analysed data, since it can lead – possibly but improbably – to samples that present characteristics that counter-evidence the null hypothesis by chance (e.g. highly correlated values). For cases in which small samples are necessary, tackling this characteristic may require increasing the sample size in order to also increase confidence in tests' results. Since in this work we are capable of collecting large samples (e.g. of size 10^6), we address it by applying the statistical tests to a set of randomly chosen segments of the analysed data. Single-sample tests such as WW and LB are independently applied over such segments, while two-sample tests such as KS and AD are applied on independently chosen segment pairs. As depicted in Figure 3(b), statistical tests' results are presented in the form of a box and whisker plot that highlights the 0%, 5%, 50%, 95% and 100% quantiles, i.e. the minimum, the median, the maximum, and the 5% and 95% quantiles, of the p-values obtained for each test. Conclusions are then drawn with increased confidence based on the obtained results' distributions, considering the employed statistical hypothesis tests are known to produce p-values approximately uniformly distributed in the range $[0, 1)$ (or with tendency to high values) when H_0 in fact holds (MARSAGLIA; TSANG, 2002). The null hypothesis is only rejected when the yielded p-values present a clear tendency to low values.

For instance, we present a comparison of the results of the KS test on two samples of one million values each. Figure 4(a) presents the p-values obtained using samples drawn from distributions known to be identical, while Figure 4(b) shows the test's results when one of the samples is shifted by a relatively small amount for inducing distribution difference. Despite both tests generated similar minimum and maximum outcomes, an approximately uniform p-value distribution is observed for the first test while the second presents a clear tendency to small values. Such behaviour should be enough for rejecting the identical distribution hypothesis for the second test. We highlight, however, that the results observed for the first test cannot be considered a proof, but should instead be faced as evidence, that the null hypothesis holds for the evaluated data (WASSERSTEIN; LAZAR, 2016).

Figure 3: Hypothesis tests' decision approaches

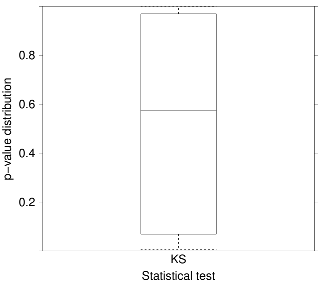


(a) Using significance level

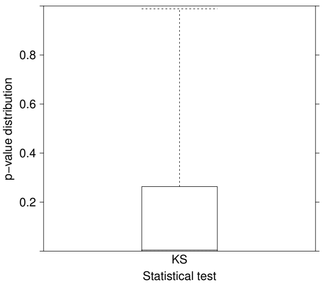


(b) Using box and whisker plot

Figure 4: KS test p-value distribution comparison



(a) Identical distributions

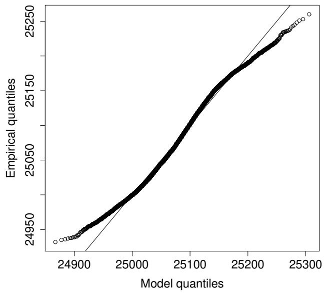


(b) Different distributions

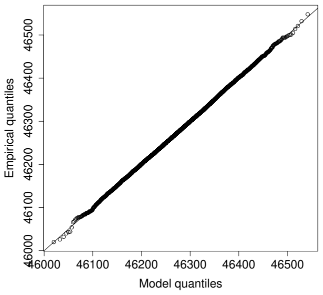
2.5.2 Applicability Plots

Besides statistical tests, plots are also employed for evidencing EVT applicability and supporting the decision on accepting or rejecting the outcomes of collected data maxima analysis. Probability and quantile comparison plots are used for evaluating how well an adjusted probability distribution (e.g. GEV or GP) fits a maxima sample (either blocks' maxima or peaks over a threshold), in order to point out whether the EVT idealized model can be confidently used for inferring over unobserved behaviour. These plots are created by sorting the sample data $x_1, x_2, \dots x_n$ in ascending order, and plotting n points $(F(x_i), \frac{i}{n+1})$ for probability plots or $(F^{-1}(\frac{i}{n+1}), x_i)$ for quantile plots – where F and F^{-1} are the cumulative distribution function and the quantile function of the adjusted distribution, respectively. Good fitting in probability and quantile plots is evidenced by the comparison points being disposed over or randomly distributed around and close to the identity line (i.e. $y = x$), meaning that the sampled values are compatible with those expected if the fitted distribution indeed represents the collected data. If, otherwise, significant systematic discrepancies are observed, the collected data maxima must be deemed not likely to be adherent to the adjusted distribution and consequently the analysis results cannot be considered trustworthy (COLES, 2001). We highlight, however, that further handling by e.g. increasing sample size or changing maxima selection parameters can often lead to distinct characteristics with respect to model fitness. We present in Figure 5(a) a quantile plot with significant model discrepancies, while Figure 5(b) shows a quantile plot with an apparently acceptable fitness behaviour.

Figure 5: Quantile plot comparison



(a) Bad fitting



(b) Good fitting

2.5.3 Conclusion

EVT is a branch of statistics that have traditionally been applied to a variety of areas as a mean for determining safety margins with low exceedance probability, supported by a reasonably strong underlying rationale, thus providing increased confidence in comparison with the definition of such margins based solely on experience. However, its applicability is subject to requirements related e.g. to the environment in which measurements are taken and to the behaviour of observed variables' outcomes. It thus requires proper evidence, e.g. statistical tests and comparison plots, being supplied to support the arguments that the obtained data can be analysed through statistical frameworks and that the adjusted extreme value models properly represent their maxima behaviour. For more information on EVT, please refer to (COLES, 2001), which presents an accessible introduction to the subject considering a number of distinct application scenarios.

3 MEASUREMENT-BASED PROBABILISTIC TIMING ANALYSIS

Measurement-Based Probabilistic Timing Analysis (MBPTA) is a recently proposed timing analysis technique that intends determining probabilistically reliable WCET bounds for RTSs' tasks, through the statistical analysis of execution time measurements using EVT. Consequently, WCET bounds produced by MBPTA – known as Probabilistic Worst-Case Execution Times (pWCETs) – are composed of both an upper-bounding value and a probability that the value is exceeded at any individual execution of the task (ABELLA et al., 2014). From the MBPTA perspective, temporal failures on computing systems are seen as just another kind of fault that must present very low occurrence probability in critical RTSs, and add up, e.g., to mechanical and structural failures. For instance, if an aircraft is designed to present a structural failure with a maximum probability of 10^{-9} per hour of operation, it should be sufficient guaranteeing that the computing system that controls it also conforms to this same maximum failure probability. The rationale behind it is that the effort on guaranteeing the control computing subsystem will never fail is not worth, since the system itself may fail with non-null probability due to a variety of unrelated reasons (CAZORLA et al., 2013a, 2016).

The probability distributions employed by EVT for modelling analysed data (execution times, in the case of MBPTA) present a right tail with decreasing slope, i.e. execution time values' probabilities are expected to decrease as they grow further from the mean value (COLES, 2001). Witnessing such behaviour in the frequency distribution of a task's maximum observed execution times gives reasons to believe that pathological values, i.e., execution times that largely exceed the maximum observed ones, should be associated to extremely or even negligibly low probabilities (CAZORLA et al., 2013a). The role of EVT within MBPTA is extending this intuition, by modelling through extreme value distributions (e.g. GEV or GP) the probabilistic behaviour of execution time samples' maxima. This enables determining WCET bounds that are expected to be exceeded with maximum probabilities that can be set to arbitrarily low values (e.g. 10^{-15}), depending on the reliability level required by the target application (COLES, 2001; CAZORLA et al., 2016).

Intuitively, MBPTA application only makes sense if the measured execution times' maxima can in fact be reliably modelled (or upper-bounded) using the probability distributions employed by EVT due to the very nature of the execution environment. In this case, the model fitting process can be considered to merely estimate the parameters of the extreme value distribution that best approximates the underlying probabilities associated to the measured execution times' maxima. However, a clear distinction between probabilities and frequencies must be made at this point. Probabilities are associated to the occurrence of phenomena outcomes, and are hardly explicit and hence generally not known beforehand, while the observation of a phenomenon's outcomes provides instead a sample of their frequencies, whose means (densities) converge to the associated probabilities as samples grow. Frequencies can hence only be regarded as evidence of the observed phenomenon's probabilistic behaviour, and should not be assumed to directly reflect the underlying probabilities. Moreover, the confidence a sample gives with respect to the population characteristics is proportional to its size, and therefore full confidence could only be achieved by researching the population – which is impossible for execution times. It follows that MBPTA uses EVT to infer on the behaviour of the (unknown) underlying execution time population, by extrapolating the limited information provided by the maxima observed on small samples (COLES, 2001; CAZORLA et al., 2013a; SANTINELLI et al., 2014; LIMA; DIAS; BARROS, 2016; GUET; SANTINELLI; MORIO, 2016).

Time-randomized processors are designed using (pseudo-)random numbers to replace internal information that, in traditional processors, are typically determined based on deterministic or speculative laws (e.g. bus arbitration and cache line replacement). This may benefit MBPTA application by causing hardware elements' latencies to be partially decoupled from execution history and from co-executing tasks' behaviour, and execution times to present variability characteristics that make their maxima more easily modellable through EVT (KOSMIDIS et al., 2016). This subject will be more thoroughly covered in Chapter 4.

3.1 APPLICATION

The application of MBPTA through EVT is done by (1) collecting a sample of the target task's execution times, (2) evidencing that the collected data is analysable through EVT, (3) selecting the maximum observed values to be employed in the rest of the analysis, (4) fitting an adequate extreme value distribution to the selected maxima, (5) producing evidence that the fitted model properly represents the statistical behaviour of the maximum observed times, and (6) determining through the adjusted model the pWCET value expected to be exceeded with a maximum probability that is sufficiently low in the application context (COLES, 2001; CAZORLA et al., 2016).

In step (1) it is necessary to (A) determine the size of the sample to be collected, such that it is large enough to characterize the execution times' behaviour but as small as possible for minimizing analysis efforts, while (B) guaranteeing that it is either representative or pessimistic with respect to the task's worst-case timing behaviour. In step (2) statistical tests are used to evidence that the data is adequate to be analysed through EVT, as described in Section 2.5.1. In step (3) the maximum values to be analysed are selected through either the BM or the POT approach, as of Section 2.1. Assuming BM is used in step (3), in step (4) a distribution of the GEV family is fitted to the selected maxima. On the other hand, if POT is employed in step (3) a distribution of the GP family should be used in step (4). In step (5), plots and statistical tests can be used to evidence that the adjusted distribution properly models the maxima behaviour, as of Sections 2.5.1 and 2.5.2.

3.2 APPLICABILITY REQUIREMENTS

As mentioned in Chapter 2, EVT fundamentally requires the analysed phenomenon outcomes to be modellable as a random variable that can be deemed independent and identically distributed. In the context of MBPTA, this means that the observed execution times should be such that (1) can be all deemed to come from the same distribution of probabilities, and that (2) the observation of any particular outcome does not affect the probability of others being observed. It is important to highlight, however, that the independence property required for MBPTA does not necessarily need to apply for every instruction's execution latency, and is instead expected to be observable for the effectively measured instruction sequences (LU et al., 2012; ABELLA et al., 2013). Consequently, individual instruction instances can present dependencies such as those induced via data (KOSMIDIS et al., 2013) and control flow (i.e. execution paths) (CUCU-GROSJEAN et al., 2012), without compromising the statistical independence required for MBPTA application (KOSMIDIS et al., 2014). Recent works on the area also claim that it is sufficient for obtaining reliable pWCET estimates through EVT that the collected execution times are stationary, i.e. present constant variance and autocovariance properties over time, and that therefore the existence of limited dependency – even between individual measurements – should not be considered to compromise MBPTA applicability (CUCU-GROSJEAN, 2013; GUET; SANTINELLI; MORIO, 2016).

As also mentioned in Chapter 2, a further requirement to apply EVT in the context of MBPTA is that the conditions under which execution times are collected must be either representative or pessimistic in relation to those expected in the environment in which the system is intended to operate. In other words, the timing behaviour of the application at analysis time must be guaranteed either to exactly match or to upper bound its timing behaviour at deployment time, which can be done either (1) deterministically, e.g. by inducing worst-case latencies being yielded during measurement collection, or (2) probabilistically, e.g. by causing latencies equal or higher than those expected at deployment time to be probabilistically predominant (GRIFFIN; BURNS, 2010; CAZORLA et al., 2013b; KOSMIDIS et al., 2016).

This poses further requirements on the measurement process and on the timing behaviour of the underlying hardware, by requiring e.g. the hardware state to be reset, randomized or carefully prepared before taking measurements, and/or hardware elements being configured to forcedly yield worst-case latencies at analysis time (PETTERS, 2002; CUCU-GROSJEAN et al., 2012; ABELLA et al., 2014). Please refer to (GRIFFIN; BURNS, 2010; MELANI; NOULARD; SANTINELLI, 2013; SANTINELLI et al., 2014; LIMA; DIAS; BARROS, 2016; GUET; SANTINELLI; MORIO, 2016) for further discussions regarding the requirements associated with MBPTA's practical applicability.

When applying MBPTA, we make the following efforts to produce execution times that meet its fundamental requirements: (1) the hardware state (e.g. of cache memories) is reset before taking measurements (CAZORLA et al., 2013b; KOSMIDIS et al., 2016), (2) time-randomized hardware elements capable of reducing dependencies are employed (e.g. cache memories) (CAZORLA et al., 2013a), and (3) data-dependent variable-latency hardware elements (e.g. the employed Arithmetic Logic Unit (ALU) design) are induced to produce the worst-case latency during measurements (KOSMIDIS et al., 2016). Independence is expected to be provided by applying (1) and (2), i.e. clearing and/or randomizing hardware elements' state, while identical distribution is expected by applying (1) and (3), i.e. using exactly the same execution environment across measurements. Proper statistical tests, such as those presented in Chapter 2, are employed for evidencing that these properties are effectively achieved (ABELLA et al., 2014).

3.3 SAMPLE SIZE

Determining the size of the samples employed in any statistical analysis is a critical task, since small samples potentially provide limited representativeness of the analysed phenomenon's populational behaviour, while collecting large ones may lead to infeasible costs. In the context of MBPTA, determining the sample sizes which are necessary for deriving reliable pWCET estimates is an open problem, because the representativeness of execution time samples is limited by multiple factors such as the variability induced both by the underlying hardware and by the measured task's control flow. The employed sample must be big enough for the effect of relatively rare timing events being individually captured, and also for suitably evidencing the smaller probabilities associated to a sufficient amount of their combinations (ABELLA et al., 2014). One possible approach for iteratively determining sample sizes was proposed in (CUCU-GROSJEAN et al., 2012), which suggests initially taking $N + N_{delta}$ measurements and increasing N of N_{delta} observations on every iteration, until the models adjusted to increasingly larger samples converge and goodness-of-fit tests' results prove acceptable. When applying MBPTA we deal with the problem of determining the sample sizes to be used by either collecting relatively large samples (e.g. of size 50000), or by evaluating experiments' results as sample sizes are increased within reasonably wide ranges (e.g. from 150 to 5000). Collecting samples whose sizes are beyond a few hundred to several thousand measurements may easily represent an undesirably large effort, especially when large applications composed of numerous and complex software tasks are to be analysed. Moreover, replications using larger sample sizes yielded no different conclusions for the evaluations we have performed in this work.

3.4 MODEL FITNESS

Achieving execution times that properly fit EVT probability distributions can be more or less challenging depending at least (1) on the timing behaviour of the underlying hardware platform, which is in general benefited by the introduction of random variability sources (see Chapter 4); (2) on the task's execution path(s) effectively measured, which can generate widely or even arbitrarily different execution time distributions; and (3) on the input data to which the tasks are subject during measurement collection, not only because of their effect on control flow but also due to internal operations' latency (e.g. ALU multiplication and division) (CUCU-GROSJEAN et al., 2012; KOSMIDIS et al., 2016; LIMA; DIAS; BARROS, 2016; LESAGE; LAW; BATE, 2018). On top of that, absolute proof on the adherence of sampled data to EVT distributions is hard (if not impossible) to be provided, since (1) samples with good or bad fitting characteristics can always occur by chance – despite their occurrence probability quickly decreases as their size is increased –, and (2) processors' timing variability is inherently not fully compatible with the statistical models to which they are expected to fit – e.g. due to clock-cycle discreteness (COLES, 2001; GRIFFIN; BURNS, 2010). When applying MBPTA we evaluate model fitness using quantile and probability comparison plots, as described in Section 2.5.2.

3.5 EXECUTION PATHS

The number of possible execution paths found in computer programs, even ones that perform simple tasks, grows extremely fast as control flow structures – especially loops – are nested. Determining the one(s) among such a multitude of execution paths are capable of generating the task's real WCET is a hard task, to which divide-and-conquer approaches are typically employed in static timing analysis (WILHELM et al., 2008). Within MBPTA, handling the existence of different execution paths in obtaining measurements that can be deemed representative regarding tasks' worst-case timing is still an open problem. The execution path problem is also directly related with the input data definition issue, since the effectively exercised execution paths mainly depend on the inputs used during the collection of measurements for applying MBPTA (LESAGE; LAW; BATE, 2018; GIL et al., 2017).

The simplest approach considered in the literature for determining input data for MBPTA is based on randomization, causing every possible input – and hence every feasible execution path – to have a non-null probability of being chosen on every execution. It leads, however, to representativity issues related to the effective measurement of execution paths that lead to the highest possible execution times (LU et al., 2012; ABELLA et al., 2014; LIMA; DIAS; BARROS, 2016). An alternative approach named Path Upper Bounding (PUB) is presented in (KOSMIDIS et al., 2014b), which creates a modified version of the analysed task in which all execution paths are padded (added with instructions) to present a timing behaviour that probabilistically upper-bounds all alternative paths. The Extended Path Coverage (EPC) technique (ZICCARDI et al., 2015; MEZZETTI et al., 2017) tackles the issue by making the measurements probabilistically path-independent, by padding their distributions considering positive effects the paths leading to individual basic blocks could have possibly caused through a time-randomized cache memory. Despite guaranteeing estimates' reliability for multi-path tasks, PUB finds limited applicability to systems requiring certification for changing the task's code, and EPC leads to pWCETs subject to overestimations for employing conservative padding.

Since we do not intend addressing the execution path problem within MBPTA, we employ benchmarks with input data fixed such that only a single long execution path is exercised. By employing single-path measurements and causing functional units with data-dependent timing behaviour (e.g. ALUs) to always yield maximum latency, we effectively remove the timing effects induced by tasks' inputs and exclusively capture the variability that arises from the hardware platform behaviour. The reasons for which we do so are twofold: (1) for creating nearly-ideal MBPTA application conditions according to the current maturity level of the technique, and (2) for evaluating the behaviour of the developed time-randomized hardware elements (see Chapter 4) in the absence of noise from other timing variability sources.

3.6 TEMPORAL ISOLATION

Most of the hardware elements that induce temporal effects on single-core processors which are hard to be predicted through static analysis methods aim to increase performance and/or utilization (e.g. cache memories and dynamic pipelines), or to reduce costs (e.g. Dynamic Random Access Memory (DRAM) memories) (WILHELM et al., 2008). In multi-core processors the effects found in single-core ones are also present, and add up to others that stem from hardware resources which are shared, for example (1) to reduce cost and/or energy consumption (e.g. buses and memories), or (2) for communication purposes (e.g. shared memories) (ČAKAREVIĆ et al., 2009). These elements cause different kinds of temporal effects: the reduction of execution times by acceleration elements – found both in single- and in multi-core processors –, and the increasing of execution times due to contention on the access to shared hardware elements – common in multi- but also found in single-core processors, e.g. when Direct Memory Access (DMA) controllers are used. Other examples of hard-to-predict timing effects are (A) timing anomalies (LUNDQVIST; STENSTRÖM, 1999; GEBHARD, 2010), that occur when local fast execution cases cause the global execution time to increase instead of decreasing (a counter-intuitive non-compositional effect), and (B) interference between cores that speed up others' execution (e.g. by fetching useful memory words into shared cache memories) (REINEKE et al., 2006; NOWOTSCH; PAULITSCH, 2012; REINEKE; WILHELM, 2014; SHAH; HUANG; KNOLL, 2014).

Particularly, timing effects that arise from shared resources in multi-core processors not only affect the executed tasks' execution times, but also make them dependent on the behaviour of other tasks executed on the same processor. In such scenarios, a slight change in any of the tasks' behaviour can cause others to temporally behave in a significantly different manner, thus potentially invalidating WCET bounds previously derived due to *temporal interference*. When such inter-task temporal effects occur due to architectural characteristics, the employed processor is said not to provide *temporal isolation* to the tasks executed on it. Accounting for these effects using static approaches often proves intractable even for relatively simple hardware platforms, and must in general be handled through conservative assumptions that often induce severe pessimism on WCET bounds (WILHELM et al., 2008; CULLMANN et al., 2010; SCHLIECKER; ERNST, 2010; KOTABA et al., 2013).

Within MBPTA the absence of temporal isolation between cores is also a harmful characteristic, because guaranteeing that measurements were taken (from a single core) under interference conditions that match or upper-bound those expected at deployment time is not straightforward. Since interference between cores don't only induce jitter but can cause execution times to vary significantly (BUI et al., 2011; KOTABA et al., 2013; CULLMANN et al., 2010), they potentially also cause measurements to present different distributions depending on other cores' behaviour, hence possibly compromising the fulfilment of the main EVT requirements. Therefore, temporal isolation absence is a characteristic that must be properly handled when tasks executing on multi-core platforms are to be analysed through MBPTA, which can be done e.g. (1) through temporally isolated resource sharing (AKESSON et al., 2011; PANIĆ et al., 2015), (2) by padding pWCETs with the worst-case interference (CAZORLA et al., 2016; KOSMIDIS et al., 2016), or (3) by guaranteeing measurements effectively capture the worst possible effects of interference (SLIJEPCEVIC et al., 2014).

3.7 RELATED WORK

The seminal works about using EVT probabilistic models to attempt solving the WCET problem are (BURNS; EDGAR, 2000; EDGAR; BURNS, 2001; EDGAR, 2002; PETTERS, 2002). They covered aspects such as (1) the basic concepts associated with probabilistic WCET analysis, (2) the higher adequacy of extreme value models in comparison with approximations based on more traditional distributions (e.g. Normal), and (3) considerations regarding the analysis of multi-path and concurrently executed tasks. The use of EVT was first mentioned explicitly in (HANSEN; HISSAM; MORENO, 2009), when maxima selection was introduced as a means of modelling execution time distributions' tails and goodness-of-fit tests were used for evidencing that the employed statistical model in fact represents the analysed data behaviour. In (GRIFFIN; BURNS, 2010) the realism of applying EVT to execution times was questioned, due to issues such as their inherent non-continuity and possible absence of statistical independence and identical distribution. Restrictions were then proposed, in order to ensure EVT's basic assumptions are observable in analysed execution times, by suggesting e.g. (A) measured tasks' inputs being restricted to the subset likely to produce the largest possible execution times (i.e. exercise the longest execution paths), and (B) the hardware state being reset before measurements for ensuring independence. Later, (CUCU-GROSJEAN et al., 2012) introduced one of the first attempts to apply EVT-based MBPTA similarly to the approach used in most of recent works. For that, methods to tackle critical issues were proposed, such as determining sample sizes in an automated manner, and further discussions regarding the use of time-randomized processors and the analysis of multi-path tasks were presented. Deeper studies on the basic requirements of EVT, and their practical impact on its use for determining trustworthy WCET upper bounds, were then presented in (CUCU-GROSJEAN, 2013; ABELLA et al., 2013; CAZORLA et al., 2013b; MELANI; NOULARD; SANTINELLI, 2013). With that, MBPTA's requirements and limitations, often related to execution times' independence and identical distribution characteristics, were more thoroughly understood. Since then, MBPTA became increasingly promising in enabling the timing analysis of complex modern systems.

Recent works were developed that (1) assessed MBPTA's practical feasibility (WARTEL et al., 2013; ABELLA et al., 2014; CONMY et al., 2015; GUET; SANTINELLI; MORIO, 2016), (2) evaluated pWCET estimates with respect to reliability and accuracy aspects (LESAGE et al., 2015; SANTINELLI; GUET; MORIO, 2017), (3) analysed the use of generalized extreme value distributions (e.g. GEV or GP) to more precisely model execution times (LIMA; DIAS; BARROS, 2016), (4) tackled the representativity issue considering e.g. the execution path coverage problem (LAW; BATE, 2016; LESAGE; LAW; BATE, 2018) and multi-mode tasks (GUET; SANTINELLI; MORIO, 2017), (5) evaluated the application of probabilistic timing analysis on traditional processors and systems (KOSMIDIS et al., 2013a; PANIĆ et al., 2015, 2017; FERNANDEZ; CAZORLA; ABELLA, 2018; SILVA et al., 2018), (6) supported deriving reliable pWCETs based on mixture execution time distributions (ABELLA et al., 2017), (7) underlined timing analysis reproducibility and execution time samples' representativity as fundamental properties for MBPTA's use (MAXIM et al., 2016), (8) discussed the effects of uncertainty sources of different nature in pWCET estimates (DAVIS; BURNS; GRIFFIN, 2017), (9) proposed hardware designs that expose and increase the observability of events affecting variable execution times for benefiting MBPTA (CAZORLA et al., 2017), (10) highlighted requirements and practices that can be considered critical for obtaining reliable pWCET estimates through MBPTA (MILUTINOVIC et al., 2017), and (11) summarized related open problems (GIL et al., 2017).

Assessments of MBPTA's results were conducted e.g. (I) through its use for the analysis of real airspace applications executing on time-randomized processors (WARTEL et al., 2013; CONMY et al., 2015), (II) by comparing the required efforts and results with those of static techniques under different hardware configurations (ABELLA et al., 2014), and (III) by introducing frameworks for evaluating results e.g. through statistical tests and fuzzy logic (GUET; SANTINELLI; MORIO, 2016). Estimates' reliability and accuracy were evaluated, e.g., by (I) comparing pWCETs with the real WCETs of synthetic tasks running on a controlled abstract platform (LESAGE et al., 2015), (II) assessing the results of stationarity, independence, identical distribution and model matching statistical hypothesis tests (SANTINELLI; GUET; MORIO, 2017), and (III) comparing the values and exceedance densities of large validation samples' maxima against pWCET estimates (see Chapter 5). Numerous recent works also covered the application of randomization at both software (BENEDICTE et al., 2016a; CROS et al., 2017) and hardware (KOSMIDIS et al., 2016) levels, in order to benefit MBPTA application (see Chapter 4).

With respect to data selection for MBPTA application, initial works (BURNS; EDGAR, 2000; EDGAR; BURNS, 2001; EDGAR, 2002; PETTERS, 2002) proposed adjusting the Gumbel distribution to raw execution time measurements. The BM approach was brought to use in the area by (HANSEN; HISSAM; MORENO, 2009), hence leading to improved applicability for not requiring raw execution times to follow any specific distribution – and requiring instead maxima convergence to extreme value distributions. To our knowledge, the POT method was first mentioned in related works in (LIU; BEHNAM; NOLTE, 2013), which evaluated its application for determining response time (i.e. latency) bounds for message transmissions on a Controller Area Network (CAN). Several posterior works (SANTINELLI et al., 2014; GUET; SANTINELLI; MORIO, 2016; SANTINELLI; GUET; MORIO, 2017) proposed and assessed novel approaches for diagnosing and applying EVT through POT to derive pWCET estimates. Challenges and potentialities on POT application to the problem are highlighted in (SANTINELLI et al., 2014), by using measurements taken from highly complex modern processors. A generalized version of EVT based on the POT approach, which relaxes some requirements of the classical methodology for increasing its applicability, was proposed in (GUET; SANTINELLI; MORIO, 2016) and extended in (SANTINELLI; GUET; MORIO, 2017).

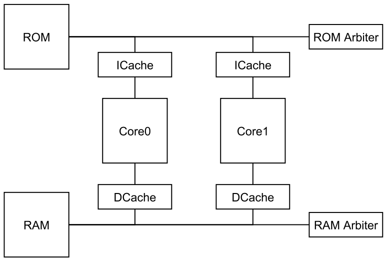
Two recent research projects funded by the European Union, namely PROARTIS (CAZORLA et al., 2013a) and PROXIMA (DAVIS et al., 2014; CAZORLA et al., 2016), targeted developing and promoting the use of MBPTA in industrial and safety-critical systems. The knowledge produced within the PROARTIS and PROXIMA projects was one of the main background references used in this thesis. Preliminary outcomes of the research presented in this thesis were shown in (SILVA; ARCARO; OLIVEIRA, 2017; ARCARO; SILVA; OLIVEIRA, 2018b, 2018a).

3.8 APPLICATION EXAMPLE

In this section we present a detailed tutorial on the application of MBPTA, using and execution time sample obtained from a well-known benchmark executing on a real time-randomized processor.

The employed time-randomized hardware platform, illustrated in Figure 6, is composed of two cores featuring a 5-stage in-order pipeline that implements the 32-bit MIPS I Instruction Set Architecture (ISA). Each core is directly connected to private and separate data and instruction caches which are 512-byte 4-word line 2-way set-associative, and implement a write-through and a purely-random replacement policy. The instruction and data caches are connected through shared buses to the respective instruction and data memory controllers.

Figure 6: Platform illustration



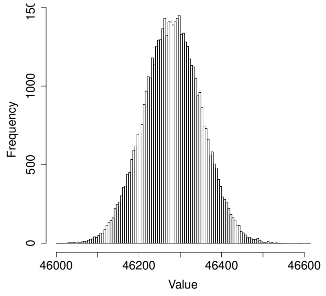
Algorithm 3.1 presents the code of the benchmark task we have employed, which consists of a sorting algorithm (bubble sort). The task's inputs are fixed such that its longest execution path in terms of the number of elementary operations is exercised, which consists of an array of integer numbers sorted in decreasing order.

Algorithm 3.1: Bubble sort benchmark

```
boolean s;
int16_t el;
int16_t il;
int16_t t;
int16_t[10] sa = { 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 };
main {
        for (el; = 0; < 10) {
                s = false;
                for (il; = 0; < (10-1)) {
                        if (sa[il] > sa[il + 1]) {
                                t = sa[il];
                                sa[il] = sa[il + 1];
                                sa[il + 1] = t;
                                s = true;
                        }
                }
                if (s == false) {
                        break;
                }
        }
}
```

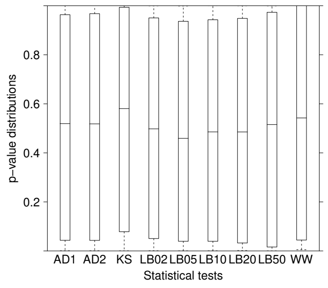
A sample of 50000 execution times was collected while the selected benchmark task was executed on the proposed time-randomized hardware platform, whose size is hence much larger than those desirable for MBPTA's practical application (CUCU-GROSJEAN et al., 2012). Figure 7 presents the histogram of the yielded execution times, in which one can observe that (1) the values concentrate around a single mode, and that (2) values' frequencies (and hence possibly their probabilities) smoothly decrease as they grow further from the mean. Despite these observations do not enable concluding whether the observed maxima can be analysed using EVT, they provide evidence that the collected data do not present characteristics that require special handling – such as the presence of mixture distributions (ABELLA et al., 2017).

Figure 7: Raw data histogram



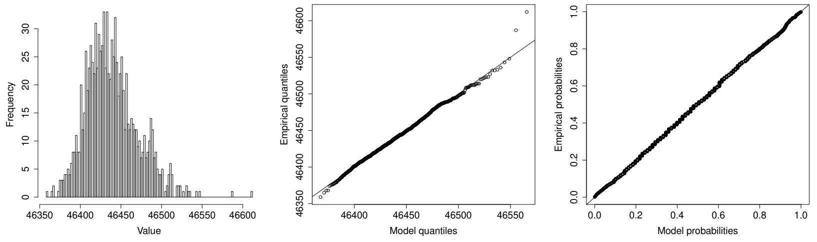
It is necessary to provide evidence that the collected measurements fulfill the basic EVT applicability assumptions, namely independence and identical distribution. For that, we present in Figure 8 the p-values yielded by applying the AD, KS, LB and WW statistical hypothesis tests on 100 random segments of 1000 measurements each – as presented in Chapter 2. Based on the approximately uniform distribution obtained for the tests' p-values we conclude, with high confidence, that the basic EVT requirements are acceptably met.

Figure 8: I.i.d. tests' p-values



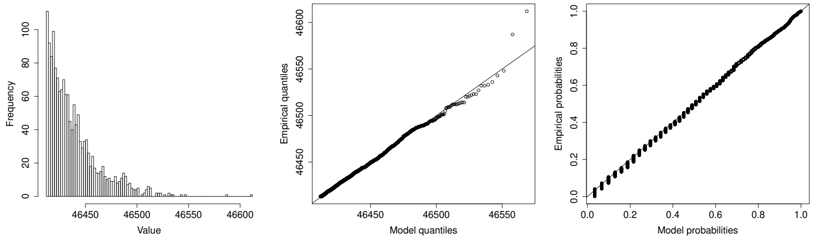
The next step consists of building the maxima samples that will be effectively analysed through EVT and adjusting proper extreme value models to them, for what we consider both the BM and the POT approaches. We present in Figure 9 the artefacts associated to EVT analysis through BM, which consist of (a) an histogram of the highest values observed on blocks of 50 measurements, and of (b) quantile and (c) probability comparison plots associated with the GEV model adjusted to the selected maxima. Similarly, Figure 10 presents the artefacts associated to the POT analysis, which consist of (a) an histogram of the observed values that exceed the threshold selected through EQMAE minimization (see Section 2.4), and of (b) quantile and (c) probability comparison plots associated with the GP model adjusted to the maxima. From these artefacts' analysis one can perceive that the collected sample's maxima present acceptable fitness to both models, since no systematic divergence is observable from the expected statistical behaviour in quantile and probability comparison plots. We highlight that, in the presence of bad fitness characteristics, further handling – such as adjusting the BM block size or using an alternative POT threshold selection approach – could lead to different results.

Figure 9: BM artefacts



(a) Maxima histogram (b) Quantile comparison (c) Probability comparison

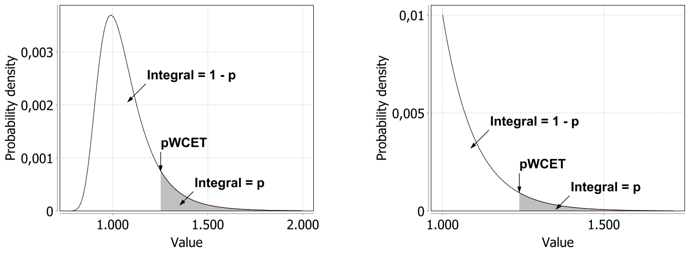
Figure 10: POT artefacts



(a) Maxima histogram (b) Quantile comparison (c) Probability comparison

At this point one can consider, with relatively high confidence, that the models adjusted to the collected data maxima can be used for deriving pWCET estimates. As illustrated in Figure 11, this is done by determining the value of the analysed variable, i.e. execution time, above which the integral of the upper tail of the fitted distribution's probability density function (in grey) is equal to or lower than the estimates' desired maximum exceedance probability (HANSEN; HISSAM; MORENO, 2009; SANTINELLI et al., 2014).

Figure 11: pWCET derivation process



(a) GEV family model (BM) (b) GP family model (POT)

We present in Table 1 a summary of the results obtained from the application of MBPTA on the collected execution time sample. We highlight that (1) these results can be considered valid only for the single execution path of the task that was exercised during measurements (MILUTINOVIC et al., 2017), and that (2) in line with this thesis' contributions presented in Chapter 5, results are also shown using the Gumbel and Exponential models in replacement to GEV and GP for applying EVT through BM and POT, respectively.

Table 1: MBPTA results

Raw execution time sample (x)
Size $length(x) = 50000$
Mean $mean(x) = 46281.80296$
Minimum $min(x) = 46002$
Maximum $max(x) = 46612$
Standard deviation $sd(x) = 69.75512$

Block Maxima sample (bm) analysis
Block size $b = 50$
Mean $mean(bm) = 46439.219$
Minimum $min(bm) = 46359$
Maximum $max(bm) = 46612$
Standard deviation $sd(bm) = 31.78732$
GEV model (95% confidence intervals)
Location $\mu \approx 46425.69579$ with $46423.78977 < \mu < 46427.54063$
Scale $\sigma \approx 27.48491$ with $26.05929 < \sigma < 28.93545$
Shape $\xi \approx -0.0934$ with $-0.14031 < \xi < -0.04868$
$pWCET(10^{-5}) \approx 46619$ with $46573 < pWCET(10^{-5}) < 46683$
$pWCET(10^{-10}) \approx 46685$ with $46602 < pWCET(10^{-10}) < 46829$
$pWCET(10^{-15}) \approx 46708$ with $46608 < pWCET(10^{-15}) < 46912$
Gumbel model (95% confidence intervals)
Location $\mu \approx 46424.29238$ with $46422.50362 < \mu < 46426.08115$
Scale $\sigma \approx 27.31963$ with $26.04845 < \sigma < 28.5908$
$pWCET(10^{-5}) \approx 46738$ with $46722 < pWCET(10^{-5}) < 46756$
$pWCET(10^{-10}) \approx 47053$ with $47022 < pWCET(10^{-10}) < 47085$
$pWCET(10^{-15}) \approx 47367$ with $47322 < pWCET(10^{-15}) < 47414$

Peak Over Threshold sample (pot) analysis
Mean $mean(pot) = 46281.80296$
Minimum $min(pot) = 46002$
Maximum $max(pot) = 46612$
Standard deviation $sd(pot) = 69.75512$
GP model (95% confidence intervals)
Threshold $\tau = 46387$
Scale $\sigma \approx 35.60811$ with $33.89807 < \sigma < 37.34064$
Shape $\xi \approx -0.17267$ with $-0.21427 < \xi < -0.13269$
$pWCET(10^{-5}) \approx 46564$ with $46532 < pWCET(10^{-5}) < 46608$
$pWCET(10^{-10}) \approx 46589$ with $46544 < pWCET(10^{-10}) < 46656$
$pWCET(10^{-15}) \approx 46592$ with $46545 < pWCET(10^{-15}) < 46666$
Exponential model (95% confidence intervals)
Threshold $\tau = 46387$
Scale $\sigma \approx 30.36507$ with $29.32701 < \sigma < 31.40313$
$pWCET(10^{-5}) \approx 46736$ with $46725 < pWCET(10^{-5}) < 46749$
$pWCET(10^{-10}) \approx 47086$ with $47062 < pWCET(10^{-10}) < 47111$
$pWCET(10^{-15}) \approx 47435$ with $47400 < pWCET(10^{-15}) < 47472$

3.9 CONCLUSION

State-of-the-art Measurement-Based Probabilistic Timing Analysis (MBPTA) techniques apply statistical techniques, such as EVT, on measurements of RTSs' tasks' execution times for producing WCET estimates with known exceedance probability. In this chapter we presented the main aspects that must be taken into consideration for applying MBPTA, together with an overview of works recently developed on the subject and a practical example of its use, considering the application of EVT through both BM and POT approaches.

4 TIME-RANDOMIZED PROCESSORS

Traditional processors' timing behaviour is generally dictated by deterministic laws built upon, e.g., recent execution history or speculations with respect to future activity. The design of such hardware is focused on achieving the smallest possible mean execution times, and therefore put no effort on making WCETs more easily analysable (NOWOTSCH; PAULITSCH, 2012). On the other hand, time-randomized processors were recently proposed, e.g., for better fitting the needs of timing analysis techniques that employ probabilistic tools, such as MBPTA. These processors replace, by design, internal information that are typically deterministic or speculative with (pseudo-)random numbers with known distribution (usually uniform). This causes hardware elements' latencies to be partially decoupled from execution history and from co-executing tasks' behaviour, and execution times to present variability characteristics that make their maxima more easily modellable through EVT (CAZORLA et al., 2013b; KOSMIDIS et al., 2016). Although the application of hardware-level time-randomization does not guarantee execution times to be analysable through EVT (LIMA; DIAS; BARROS, 2016), it was shown to often lead to execution times that better meet its basic requirements (KOSMIDIS et al., 2016).

The results yielded by EVT for WCET analysis are expected to be reliable if the collected execution time sample (1) sufficiently covers high-latency timing events and captures their most probable combinations, (2) presents frequencies that are representative with respect to the probabilities associated to the observed execution times, and (3) produces maxima distributions that fit the probabilistic models employed by EVT. Whenever high-latency timing events are associated to sufficiently low probabilities, small samples can lead EVT to produce optimistic pWCET estimates for not properly characterizing the system's worst-case temporal behaviour. Probabilities associated to such events should hence be within observable ranges (e.g. 10^{-3} or higher), for both their individual and combined effects being captured by EVT (ABELLA et al., 2014). Suppose, for instance, that the execution time of a certain code segment depends on the combination of only two independent timing events e_1 and e_2, whose exact occurrence probabilities P_{e_1} and P_{e_2} are only known to be equal (i.e. $P_{e_1} = P_{e_2}$). In this case, a certain low probability of observing the segment's WCET, e.g. $P_{WCET} = 10^{-10}$, would only occur if the individual events' probabilities were at most $P_{e_1} = P_{e_2} \leq \sqrt{P_{WCET}} = 10^{-5}$. A large sample could hence prove necessary for ensuring representativeness regarding the timing effects of the individual events, and especially of their combinations. However, if a third timing event e_3 was introduced, it would be enough for all three having occurrence probabilities $P_{e_1} = P_{e_2} = P_{e_3} \leq \sqrt[3]{P_{WCET}}$ for the assumed WCET probability being achieved. A much smaller sample could then prove sufficient to capture the effects of the individual events and of some of their combinations. In general, $P_{e_i} = \sqrt[n]{P_{WCET}}$ applies to the combination of n independent events of equal probability for achieving the target WCET probability P_{WCET}. It follows that, as the number of events that influence execution time grows, their worst-case combination quickly becomes a rare event – even if each of them is associated with relatively high probabilities. Despite time-randomized processors cannot eliminate the possibility of large execution times being produced, they benefit MBPTA at least by (A) weakening or eliminating dependence between timing events that influence execution times, and (B) potentially making the WCET rare enough to be hardly observable during the entire life cycle of a system for depending on long chains of (pseudo-)random events (CAZORLA et al., 2016; KOSMIDIS et al., 2016).

Besides benefiting probabilistic WCET analysis through EVT, hardware-level randomization is also capable of introducing interesting properties on RTSs. For instance, it is capable of (A) mitigating the possibility of hardware characteristics being exploited to systematically induce pathological execution times, (B) smoothing performance variation in comparison with traditional designs as system parameters vary, (C) mitigating hardware aging effects, and even (D) improving systems' characteristics associated with safety and security issues (TRILLA et al., 2016, 2017a, 2017b, 2018). Consequently, randomization is also potentially capable of allowing certification standards (e.g. automotive) to treat software timing faults as random, which would avoid rigorous proof processes they must currently undergo for being considered systematic faults (AGIRRE et al., 2018).

Time-randomization can only be implemented for hardware elements whose temporal behaviour is capable of being randomized without affecting the provided functionalities, or for elements that perform purely speculative functions – such as cache memories or branch predictors. Whenever time-randomization is not feasible for a certain element, it is necessary to determine whether its latency variability depends or not on input data, since in the positive case different execution time distributions could be produced depending on employed tasks' inputs. Non-randomizable hardware elements' data-dependent latencies must be either (A) ensured to present exactly the same distribution at analysis and at deployment time, granting measurements with representativeness regarding the deployment-time environment, or (B) upper-bounded either deterministically or probabilistically, thus increasing reliability at the cost of tightness for assuming the worst-case latency often/always occurs. Another possible solution is padding (i.e. augmenting) measurements based on knowledge regarding the worst-case number of variable-latency operations performed and their maximum latency. The main advantage of this approach is its applicability to commercial processors, while as disadvantages we highlight the pessimism introduced and the effort to analytically derive the maximum number of operations performed by the analysed task (CAZORLA et al., 2013b; ABELLA et al., 2014; KOSMIDIS et al., 2016).

4.1 RELATED WORK

To our knowledge, the application of randomization techniques at hardware level for increasing timing analysability through statistical methods was first suggested in (PETTERS, 2002). The proposal was that the hardware state affecting execution times should be preferably set to the worst case, but if this cannot be done (e.g. for cache memories) it should be randomized before taking measurements. Posterior works, such as (QUIÑONES et al., 2009; CUCU-GROSJEAN et al., 2012), extended the approach by proposing randomization with the introduction of (pseudo-)random number generators at hardware level for both mitigating cache-related performance anomalies and benefiting timing analysis. Several hardware elements with randomized timing behaviour were proposed in existing literature, for instance (1) cache memories with randomized placement/replacement (KOSMIDIS et al., 2013a), (2) bus arbiters based on random policies (LAHIRI; RAGHUNATHAN; LAKSHMINARAYANA, 2001; JALLE et al., 2014), and (3) Networks-on-Chip (NoCs) with probabilistic forwarding (SLIJEPCEVIC et al., 2016, 2017a). Time-randomized cache memories employ policies that randomly determine, e.g., (1) the mapping between memory addresses and the cache lines in which they should be stored, and/or (2) the cache lines to be evicted upon misses (KOSMIDIS et al., 2013a). Randomized bus arbitration policies may choose the next client to be served, e.g., (1) based solely on the value of pseudo-random numbers (LAHIRI; RAGHUNATHAN; LAKSHMINARAYANA, 2001), or (2) by randomly permuting a fixed non-work-conserving schedule (JALLE et al., 2014). Randomized NoCs can, e.g., randomly choose which among the data flits available at the routers' inputs should be forwarded to the output(s) (SLIJEPCEVIC et al., 2016), or schedule flits' forwarding in a randomly permuted order (SLIJEPCEVIC et al., 2017a). A more thorough discussion regarding hardware-level changes capable of benefiting MBPTA's application is presented in (KOSMIDIS et al., 2014).

Recent works were developed aiming, e.g., (1) improving time-randomized hardware elements by providing lower cost, complexity and/or energy consumption (KOSMIDIS et al., 2014a), (2) implementing time-randomization on hardware elements other than those previously proposed (SLIJEPCEVIC, 2017), and (3) addressing randomized cache memory issues associated e.g. to pathological replacement patterns (BENEDICTE et al., 2018) and to representativeness (MILUTINOVIC; ABELLA; CAZORLA, 2017; MILUTINOVIC et al., 2018). Several recent works targeted the development and evaluation of a probabilistic variant of the LEON3 processor, which is currently employed in aerospace systems in its traditional form, with the objective of leveraging probabilistic timing analysis for critical systems (HERNANDEZ et al., 2015; KOSMIDIS et al., 2016; FERNANDEZ et al., 2017).

4.2 CACHE MEMORIES

Caches are fast temporary memories that store instructions and/or data that were recently or will probably be accessed soon by the processor, in order to reduce the impact of using large but relatively slow RAM memories (e.g. DRAMs) with fast processors. Cached memory subsystems retrieve sets of neighbour words that contain those requested by the processor, instead of separately fetching single words at each request, and store these sets into cache memories' lines. If a word belonging to a previously retrieved line is accessed later, the cache memory serves it immediately instead of fetching it from the main memory, causing accesses to cached words to present significantly lower latency in comparison to non-cached ones (GONZALEZ; LATORRE; MAGKLIS, 2010). This affects both data and instructions access times and, consequently, also affects execution times (WILHELM et al., 2008). Since cache memories are generally much smaller than RAM memories, whenever a new cache line must be stored a previously inserted one may need to be discarded (evicted), which requires a replacement policy – e.g. First-In First-Out (FIFO) or Least Recently Used (LRU) – to be used (JACOB; NG; WANG, 2008). Specific characteristics of how objects are allocated and/or accessed in memory can cause the number of cache misses to systematically increase in a pathological manner, reducing performance and ultimately leading to execution times comparable to those expected if no cache memories were used at all (WILHELM et al., 2008). Predicting or detecting such effects is a hard task that may even require considering all possible scenarios, which is in general computationally infeasible, thus posing severe difficulties in deriving static bounds for the WCETs of tasks executed in hardware platforms that employ cached memory subsystems (QUIÑONES et al., 2009).

To our knowledge, (QUIÑONES et al., 2009) was the first work to propose the use of randomized cache memories to benefit WCET probabilistic analysis, despite a similar design was proposed much earlier for reducing cache miss rates' sensitivity to data placement (SCHLANSKER; SHAW; SIVARAMAKRISHNAN, 1993). Randomized variants can be implemented for both the placement and the replacement policies employed by cache memories. The placement policy defines the set in which memory blocks are to be stored, and the replacement policy defines which blocks are to be evicted when new ones must be stored. A simple randomized replacement policy can be implemented by choosing the line to be evicted in the target set based on a pseudo-random number (CUCU-GROSJEAN et al., 2012). A novel random replacement policy is proposed in (BENEDICTE et al., 2018), which avoids pathological patterns that are observable when purely random replacement is used by replacing the lines within cache sets in a randomly permuted order. Placement policy randomization requires the use of a proper address hashing function that randomly maps memory block addresses to cache sets, ideally in a homogeneous pattern to avoid pathological allocation cases (e.g. all addresses to a single set) (KOSMIDIS et al., 2013a; HERNANDEZ et al., 2016). A random placement policy of relatively low hardware complexity is presented in (KOSMIDIS et al., 2013a, 2014a), which supports both set-associative and directly-mapped arrangements and whose performance is comparable to that of traditional designs. It uses a hash function based on randomized rotation operations, for producing a homogeneous mapping of addresses to cache sets, using for that multiplexers and XOR gates disposed in multiple levels. Another random placement approach was proposed in (HERNANDEZ et al., 2016), which uses a hash function based on the permutation of address bits. The approach is claimed to better exploit spatial locality, by avoiding conflicts between addresses that belong to a same cache segment, and was shown to present lower latency and area costs in relation to alternative designs. An evaluation of cache memories with randomized placement and replacement policies is presented in (ANWAR, 2016).

Time-randomized caches improve probabilistic timing analysis techniques' applicability by causing hits/misses to be driven mainly by probabilistic laws (ABELLA et al., 2013), hence making the probability of any particular eviction pattern being observed (including pathological ones) extremely low (CAZORLA et al., 2013a). As mentioned, randomized caches also decouple execution times from execution history, thus also reducing the risk of both cache-induced performance anomalies and strong dependencies between execution times (QUIÑONES et al., 2009). Further decoupling between timing behaviour and execution history can be achieved using evict-on-access time-randomized cache memories, which randomly evict lines on every access, at the cost of performance reduction due to unnecessary evictions (CAZORLA et al., 2013a). Moreover, instruction and data time-randomized cache memories can be used both in multi-level and in unified configurations, without compromising or rising the cost of probabilistic timing analysis techniques (KOSMIDIS et al., 2013b). With respect to shared configurations, it was shown in (KOSMIDIS et al., 2013b) that the only information needed for bounding the interference to which a task is subject through a shared time-randomized cache is the reuse distance, i.e. the maximum number of accesses performed between two consecutive accesses to particular memory addresses. Based on that, a shared cache design was proposed in (SLIJEPCEVIC et al., 2014) that limits temporal interference by delaying misses' servicing such that the frequency of inter-core evictions is limited to a fixed threshold. This threshold can then be defined and enforced such that the application's timing behaviour during execution times' measurement probabilistically upper-bounds the one expected at deployment time.

Certain characteristics of time-randomized caches can be considered harmful when employed in critical RTSs (REINEKE, 2014), and therefore proper handling is necessary while collecting execution time measurements (MEZZETTI et al., 2015) for guaranteeing the obtained pWCETs are reliable. It was also shown in (LIMA; DIAS; BARROS, 2016) that execution times' statistical behaviour with respect to EVT modelling can degrade depending on time-randomized caches' sizes. On top of that, there are use patterns that can lead to pathological temporal behaviour when time-randomized cache memories are used under certain environments, e.g. when the cache associativity is smaller than the number of addresses which are (pseudo-)randomly mapped to cache sets (ABELLA et al., 2014). To tackle such issues (ABELLA et al., 2014) proposed a mean to detect their presence, and suggested decreasing cache memories' sizes during measurement collection for increasing the chance of effectively measuring such patterns' effects. For determining sample sizes needed to provide representativeness with respect to cache-related timing events, (A) (MILUTINOVIC; ABELLA; CAZORLA, 2016, 2017) introduced a simulation-based method to search for instruction and data cache placements that potentially lead to pathological behaviour at deployment time, (B) (BENEDICTE et al., 2016b) proposed a technique to compute the exact probabilities of cache-related events considering objects' allocation in memory, and (C) (MILUTINOVIC et al., 2018) proposed jointly applying the Path Upper-Bounding (PUB) technique (KOSMIDIS et al., 2014b) and an address conflict analysis (MILUTINOVIC et al., 2017) on multi-path tasks.

4.3 BRANCH PREDICTORS

Branch prediction mechanisms are designed to reduce the number of pipeline flushes, i.e. instruction sequences' reloads, by loading the "most probable" execution path of the program into the pipeline after branch instructions' execution (GONZALEZ; LATORRE; MAGKLIS, 2010). Their use significantly reduces the execution time of instruction sequences containing branches, since a reasonable amount of pipeline flushes is avoided, but there is no such mechanism able to correctly predict all branch targets – causing execution times to vary (WILHELM et al., 2008). Branch predictors may be either static, if branches are predicted based on static information (e.g. always taken), or dynamic, if information regarding (mis)prediction history is stored and used to improve future predictions. Some dynamic branch predictors employ buffers known as Branch Target Buffers (BTBs) to store such information, which largely improve their prediction accuracy but make branch instructions' execution times dependent on the previously executed ones' results (GONZALEZ; LATORRE; MAGKLIS, 2010; TANENBAUM; AUSTIN, 2012). These buffers are subject to effects similar to those found in cache memories in relation to the prediction of their content, hence making their analysis through static methods considerably challenging (WILHELM et al., 2008). For time-randomized platforms, branch prediction can be implemented with low hardware cost by considering the value of a single pseudo-random bit for choosing whether to predict branches as taken or not taken (BALL; LARUS, 1993).

4.4 BUS ARBITERS

Buses are wire sets used to transport data between multiple subsystems within a computing system, and are used to interconnect hardware elements such as processing cores, memories, and peripherals (NULL; LOBUR, 2003; JACOB; NG; WANG, 2008; MURDOCCA; HEURING, 1999). Buses are often shared among multiple elements and, since parallel bus accesses are in general not possible, simultaneous requests must be arbitrated over time according to a certain policy in order to be served sequentially. This causes concurrent access requests to be delayed (contained), and therefore a shared bus access latency varies at least due to: (1) contention delay until the bus can be accessed, and (2) communication delay while the bus access is performed; the former directly depends on the number of requesting devices and on the employed arbitration policy, and the latter mainly depends on the performed operation duration and on the bus' characteristics (e.g. width and speed) (DASARI et al., 2013). If the bus that connects the processor's core(s) to the main memory (i.e. the memory bus) is shared, then any access to the main memory can be delayed. In fact, many of the modern commercial processors' cores communicate with other elements (including the main memory) through a bus known as Front-Side Bus (FSB). If the FSB is shared, accounting for execution delays due to FSB access contention using static analysis methods is hard because (1) generally very limited documentation on the bus implementation is available for commercial hardware (FERNANDEZ; CAZORLA; ABELLA, 2018), and (2) if the bus protocol is not designed to be predictable (e.g. employs pipelined and/or out-of-order transactions) then the state space may prove too large to be analysed (DASARI et al., 2013). On top of that, peripheral buses such as Peripheral Component Interconnect (PCI) are employed to connect a multitude of Input/Output (I/O) devices to modern computing systems, and often use controllers that access the memory bus to perform processor-independent data transfers, thus potentially delaying accesses issued by the processor by a significant amount (NULL; LOBUR, 2003; SCHÖNBERG, 2003; PELLIZZONI; CACCAMO, 2010). These characteristics induce interference between tasks executed on the cores that access a shared bus, and can also make their timing dependent on peripherals' behaviour.

Time-randomized arbitration policies such as lottery bus (LAHIRI; RAGHUNATHAN; LAKSHMINARAYANA, 2001) and random permutation (JALLE et al., 2014) were proposed to be used in hardware platforms targeted to probabilistic timing analysis. The lottery bus policy uses pseudo-randomly generated numbers and, by taking into account the range of the generated values, grants access to clients exclusively assigned to each of the feasible value ranges. By properly reserving pseudo-random values' ranges, different service amounts can be allocated to each client which are served in a probabilistic manner. However, since the next client to be served is chosen in a purely random fashion, it relies exclusively on the fast convergence of the associated probability function to avoid starvation – i.e. starvation is possible despite being highly improbable (LAHIRI; RAGHUNATHAN; LAKSHMINARAYANA, 2001). The random permutation policy builds a servicing schedule which is randomly permuted after each arbitration round, and contains at least one time slot allocated for each client. Random permutations can be produced based on a few pseudo-randomly generated bits, by deciding based on each bit's value whether to swap or not the positions of each client (or group of clients) in the schedule. Besides inducing execution time variability, the random permutation policy also guarantees that every client is served at least once during every arbitration round, hence guaranteeing starvation absence (JALLE et al., 2014). For cases in which clients can issue bus requests of different lengths, (SLIJEPCEVIC et al., 2017b) evaluates the introduction of credit-based regulators to grant fairness in the access to shared hardware resources.

4.5 NETWORKS-ON-CHIP

Network-on-Chip (NoC) are network infrastructures based on data packets' routing that are typically used in processors to interconnect Intellectual Property (IP) cores, i.e. proprietary reusable logic units (e.g. processing cores). NoCs are composed of switches disposed and interconnected according to a certain topology, network interfaces through which data packets are sent and received by each of the IP cores, and point-to-point links between these elements. Packets transmitted in NoCs are split into *flits* (fixed-length segments) which are then routed to the destination according to a certain policy (MURALI, 2009; HENNESSY; PATTERSON, 2012; TSAI et al., 2012; MA et al., 2014). Due to their network-like structure, the time required for the execution of a transaction (transmission of a full packet) in a NoC depends on factors such as (1) its topology and the employed routing algorithm, (2) the number of channels and the bandwidth supported by the network, (3) the time required by the routing elements to store and/or forward flits or packets, and (4) the volume of packets being transmitted in the network (KOTABA et al., 2013; BUI; CACCAMO; PELLIZZONI, 2011).

A NoC based on a tree topology with probabilistic forwarding that targets all-to-one communication patterns (e.g. for memory access) is proposed in (SLIJEPCEVIC et al., 2016). Besides scaling better to high core counts in comparison to buses, the design enables asymmetric bandwidth allocation for individual cores and presents small impact on average performance due to randomization. In (SLIJEPCEVIC et al., 2017a) a time-randomized mesh NoC based on randomized wormhole routing is proposed, which induces probabilistic behaviour on contention times by employing random permutation arbitration at the flit forwarding policy (JALLE et al., 2014). It also exploits the possibility of limiting in-flight requests, which often proves not to benefit NoCs' worst-case timing analysis based on traditional approaches, for reducing WCET estimates produced through probabilistic techniques.

4.6 RANDOMNESS SOURCES

Throughout the years, random numbers found application in many areas, targeting a huge variety of objectives (GALTON, 1890). Numerous approaches were hence developed for the generation of random numbers using computers, at both software (KNUTH, 1997) and hardware levels (ALFKE, 1996), and for the assessment of characteristics they are expected to present (MARSAGLIA; TSANG, 2002). The specific method used for the generation of random numbers at hardware level must be chosen taking into account the purpose to which they are going to be employed. For instance, relatively simple pseudo-random generators may be considered sufficient for use within optimization techniques (SARMA, 1990), but such methods are not suitable for cryptographic applications that require highly unpredictable randomness sources (INTEL, 2012). A very common approach for the implementation of Pseudo-Random Number Generators (PRNGs) at hardware level is through Linear Feedback Shift Registers (LFSRs) (ALFKE, 1996), which consist of shift registers whose input bits are given as linear functions of their previous states. By properly choosing LFSRs' feedback polynomials, PRNGs can be implemented with relatively low hardware cost and with statistical characteristics that can be considered sufficient for fulfilling the needs of numerous application contexts (KOETER, 1996).

Throughout this work, we mainly employ a PRNG similar to that presented in (TKACIK, 2003), which is composed of two independent PRNGs – a LFSR and a Cellular Automata Shift Register (CASR) – whose results are XOR-combined to produce the effectively used pseudo-random values. The approach was shown to provide improved statistical behaviour in relation to the isolated use of either LFSRs or CASRs, and also presents remarkably low complexity and cost with respect to its hardware implementation. During execution the two generators (A) are independently seeded, i.e. distinct seeds are provided for the LFSR and the CASR, from a true randomness source, and (B) are clocked from two independent and non-synchronized ring oscillators. As true randomness sources, we use both (A) the RDRAND instruction, provided by Intel® processors based on the Ivy Bridge micro-architecture (INTEL, 2016), which outputs a stream of random numbers produced by a hardware PRNG that is periodically seeded with the outcome of an encryption algorithm applied on a non-deterministic thermal-noise-based entropy source (INTEL, 2012); and (B) a True Random Number Generator (TRNG) based on a ring of ring oscillators (BAETONIU, 2004).

4.7 CONCLUSION

Time-randomized processors were recently proposed as a means for improving the applicability of probabilistic timing analysis for RTSs, by partially decoupling hardware elements' latencies from execution history and causing execution times to present variability characteristics that make their maxima more easily modellable through statistical tools such as EVT. As presented in this chapter, time-randomized versions of a number of hardware elements – such as cache memories, bus arbiters and NoCs – are currently available. One of the main contributions of this thesis, which will be presented in Chapter 6, consists of an evaluation of randomized scheduling approaches on multithread pipelines for benefiting the application of MBPTA.

5 pWCET RELIABILITY EVALUATION

In this chapter we present an empirical assessment of the use of extreme value models on the analysis of RTSs' execution times through EVT, considering both BM and POT approaches. The objective of this evaluation is detecting whether the considered EVT models can be deemed suitable for determining pWCET estimates with respect to the reliability of the produced results. For that we take advantage of the possibility of collecting arbitrarily large samples of tasks' execution times, and assess whether the values and the densities of maxima observed on large validation samples can be deemed compatible with the pWCET estimates produced using extreme value models fitted to small modelling samples. More specifically, we employ a set of large execution time datasets (i.e. of size 10^8) for assessing the reliability of pWCET estimates produced through EVT employing both the BM approach, using the GEV and Gumbel models, and the POT approach, using the GP and Exponential models. We make a clear distinction between modelling samples, which are used in fitting statistical models for applying EVT, and validation samples, much larger datasets we use to obtain High Water Marks (HWMs) (i.e. maximum observed execution times) and exceedance frequencies (i.e. counts of values that exceed pWCET estimates) for evaluating the results obtained using adjusted models. We highlight that the collection of validation samples is not expected to be incorporated into the probabilistic timing analysis process, due to its relatively high cost, but it is a valuable tool in providing evidence of the overall method consistency and reliability. The main outcome of the herein presented investigation is counter-evidence to the reliability of the GEV and GP models for pWCET derivation through the BM and POT approaches, respectively, and empirical evidence of the Gumbel and Exponential models' adequacy in the same context for maxima tails with shape $\xi \leq 0$.

5.1 RELATED WORK

Both seminal (BURNS; EDGAR, 2000; EDGAR; BURNS, 2001; HANSEN; HISSAM; MORENO, 2009) and several posterior works (LU et al., 2011; CUCU-GROSJEAN et al., 2012; ABELLA et al., 2014) related to MBPTA considered that execution time distributions would present non-heavy right tails, and hence adhere to models with exponentially decreasing tails (e.g. Gumbel or Exponential). On the other hand, it was shown in (LIMA; DIAS; BARROS, 2016) that maxima adherent to any of the Weibull, Gumbel and Fréchet extreme value distributions can be produced by tasks' execution times using random input data (LU et al., 2011). With that, the use of more flexible models, such as GEV or GP, was suggested within MBPTA.

Increasing sample sizes with the objective of evaluating the outcomes of probabilistic timing analysis techniques is a relatively common practice. In (CUCU-GROSJEAN et al., 2012), samples larger than those considered sufficient based on model convergence criteria (i.e. of size 10^4) were used to evaluate the variability of pWCET estimates derived through the BM approach. Large samples (i.e. of size 10^5 to 10^7) are also employed (A) in (LIU; BEHNAM; NOLTE, 2013) to assess POT results' pessimism in comparison with those of BM, (B) in (SANTINELLI et al., 2014) for both comparing the outcomes of BM and POT and evaluating the impact of different BM block sizes and POT thresholds on MBPTA results, and (C) in (ABELLA et al., 2017) to obtain HWMs for assessing the yielded pWCET estimates' reliability and tightness.

An assessment of pWCET estimates derived using the POT approach is presented in (SANTINELLI; GUET; MORIO, 2017), which associates their reliability to the results of stationarity, independence, identical distribution and model matching statistical hypothesis tests. Several recent works have also tackled the pWCET reliability problem from the specific perspective of guaranteeing representativeness regarding events associated to cache memories, considering randomization both at hardware (MILUTINOVIC; ABELLA; CAZORLA, 2017; MILUTINOVIC et al., 2018) and at software (BENEDICTE et al., 2016a) levels. Recently proposed approaches for tackling important aspects that must be taken into consideration for ensuring pWCET estimates' reliability are highlighted in (MILUTINOVIC et al., 2017).

In this chapter, we show that the use of GEV and GP probabilistic models for determining pWCETs can lead to unreliable results even under nearly-ideal conditions, and Gumbel and Exponential are hence recommended being used instead – possibly at the cost of tightness. Preliminary results regarding these findings were presented in (SILVA; ARCARO; OLIVEIRA, 2017; ARCARO; SILVA; OLIVEIRA, 2018b, 2018a), considering EVT application through both BM and POT. In line with these suggestions, a method for deriving reliable pWCET estimates in the presence of mixture distributions is proposed in (ABELLA et al., 2017), which adjusts the Exponential model to execution times that exceed a POT threshold selected such that the modelled tail is known to decrease at least exponentially. In contrast with (SANTINELLI; GUET; MORIO, 2017), our evaluations associate pWCET reliability it with the effective upper-bounding (A) of the maximum values effectively observed in large samples, and (B) of the empirical densities observed in validation samples' tails. We, however, still require the results of EVT's typical applicability statistical tests to prove acceptable. Moreover, our work follows many of the directions proposed in (MILUTINOVIC et al., 2017), e.g. by employing time-randomized hardware and causing non-randomizable logical units to yield maximum latency, but does not intend tackling other aspects raised in the same work, and we therefore e.g. fix tasks' inputs for ruling out control flow timing effects.

5.2 EXPERIMENT DESIGN

The proposed pWCET estimates' evaluations are performed by comparing, as modelling sample sizes are increased, (A) the maximum values observed in large validation samples (HWMs) with the pWCET estimates determined through EVT for exceedance probabilities of magnitudes expected in practice (i.e. 10^{-15}), and (B) the empirical exceedance densities observed in large validation samples against their expected values, considering pWCETs with exceedance probabilities' magnitudes smaller than that of validation samples' sizes (i.e. 10^{-7}).

For the sake of conclusion generalization, the experiment was multidimensionally replicated considering (A) different EVT model fitting methods, (B) distinct modelling sample size increasing steps, and (C) multiple execution time samples. For (A), replications were made fitting the GEV and GP models through the MLE (SMITH, 1985), GMLE (MARTINS; STEDINGER, 2000) and L-moments (HOSKING, 1990) methods. Since all three produced similar outcomes, we focus on results produced using the L-moments method. The Gumbel and Exponential distributions were adjusted using the MLE method (EDGAR; BURNS, 2001). The determination of POT thresholds was performed using the EQMAE minimization approach, as presented in Section 2.4, which consists of selecting the threshold that minimizes the EQMAE metric (WILLMOTT; MATSUURA, 2005) of the GP model fitted to the exceeding execution time values. For (B), evaluation replications were made with sample sizes being increased in steps of 50, 100, 200 and 400 measurements. Since all evaluated steps led to equivalent results, the presented artefacts are those obtained using steps of size 50. We highlight that, for the BM approach, the employed measurement block sizes equal the sample size step used, while for POT evaluations the sample size step only affects the rate by which samples grow. For (C), large execution time samples were employed of both real nature, i.e. containing measurements collected from real hardware platforms executing differently behaving benchmark tasks, and of synthetic nature, i.e. produced using pseudo-random number generators. With that, models are evaluated on execution times (A) that were indeed produced by real time-randomized processors, (B) that could be yielded by systems that differ in construction and/or in complexity from those that produced the real-hardware samples, and (C) for which the correct value of the POT threshold is known beforehand.

5.2.1 Real-Hardware Samples

The real-hardware validation samples used in our research are composed of execution times collected from two different versions of a time-randomized hardware platform, both running on a Field-Programmable Gate Array (FPGA) board at 50MHz. These platforms employ dual-core processors with separate instruction and data RAM memories, each accessed through a shared bus that is arbitrated according to a randomized policy. The processing cores use simple five-stage pipelines that implement the MIPS instruction set, which produces latencies that vary depending on the internal pipeline behaviour and, consequently, on the timing of other elements such as cache memories, bus arbiters and memory controllers. A 512-byte 2-way set-associative cache memory is used which implements the modulo placement policy, a randomized line replacement policy (KOSMIDIS et al., 2013a), and the write-through update policy. Time-randomization is achieved at hardware level through a PRNG that has a single clock cycle latency and works by XOR-combining the results of two other independent generators (TKACIK, 2003), and is periodically seeded using a cryptographic-quality TRNG. One of the employed hardware platforms, to which we refer as *DPArptdm*, has no cache memories and uses a bus arbitration policy that determines the next client to be served based on a randomly permuted non-work-conserving schedule (JALLE et al., 2014). The second platform, namely *DPCpArrr*, employs one private cache memory for each core and a bus arbitration policy that chooses the next client to be served in a purely random manner. For more details on the hardware elements developed during the herein described research, please refer to Appendix C.

The software tasks we used for obtaining the real-hardware validation samples were *bsort*, *insertsort*, *bs*, *expint*, *fdct*, *crc*, *matmult*, *fir*, *fibcall* and *cnt*, all from the Mälardalen WCET Benchmarks suite (GUSTAFSSON et al., 2010). In order to cause high execution times being yielded, tasks' inputs were fixed such that only a single and long execution path (i.e. that performs a large number of elementary operations) is exercised. For instance, we apply *bsort* and *insertsort* on reverse-sorted integer-number arrays, and execute *bs* with a key that maximizes the number of divisions performed on the search interval. In line with the suggestions of (GRIFFIN; BURNS, 2010; CUCU-GROSJEAN et al., 2012; KOSMIDIS et al., 2016), we also (1) configure the processor's ALUs to produce maximum latencies during measurements, which in our case causes integer multiplication and division operations to always take 32 clock cycles to complete, and (2) reset the state of the hardware platform before executing the benchmarks, to avoid dependency being induced between measurements due to state-related effects. Consequently, the observed timing variability stems exclusively from the time-randomized processor, since we isolate effects from any other sources. We therefore consider our measurements were obtained under nearly-ideal conditions, in which MBPTA should hence yield remarkably reliable results to be considered usable in the general case. We highlight, however, that (A) these conditions are not sufficient for guaranteeing that the real WCETs are observable (e.g. due to cache effects), and that (B) the determination of the tasks' input data for the application of MBPTA is still an open problem (GIL et al., 2017).

Since all the evaluations we performed led to equivalent conclusions under all the considered scenarios (i.e. for all combinations of platforms and benchmarks), we only discuss the analysis results obtained for the execution times of the *bsort* task running on the *DPCpArrr* platform. Artefacts associated with all other scenarios' evaluations can be found in Appendix A.

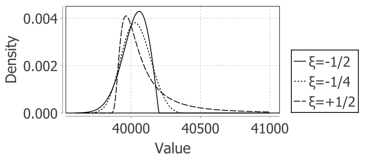
5.2.2 Synthetic Samples

We also reproduced the entire set of evaluations on synthetic (i.e. pseudo-randomly generated) samples, as a means of generalizing the conclusions drawn from real-hardware samples' analysis. Such replication enabled us extending the research coverage by assessing whether our conclusions could differ if (1) the collected samples presented distinct tail shapes due to intrinsic characteristics of the system, or (2) higher quality thresholds than those selected through EQMAE minimization could be chosen while applying EVT using POT. We therefore consider that the synthetic datasets we employed are valuable tools in corroborating the validity of our findings on domains that could prove too hard or expensive to be assessed through real-hardware samples. The synthetic datasets were generated through the *revd* function of the *R* (R, 2017) *extRemes* package (GILLELAND; KATZ, 2016), which is capable of producing arbitrarily large samples of random values adhering to specific extreme value distributions. The random draws produced through *revd* were rounded up to the nearest integer value, in order to better reproduce the behaviour of execution times whose minimum practically achievable measurement granularity is the clock cycle. We performed the analysis on two distinct sets of pseudo-randomly generated datasets, which we refer to as (1) GEV synthetic samples and (2) GP synthetic samples.

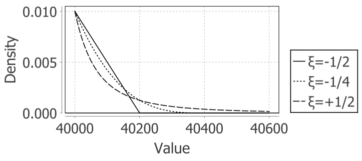
The GEV synthetic samples adhere to the GEV distribution with location $\mu = 40000$ and scale $\sigma = 100$, and each assumes a different shape such that $\xi \in \{-\frac{1}{2}, -\frac{1}{4}, -\frac{1}{8}, 0, +\frac{1}{8}, +\frac{1}{4}, +\frac{1}{2}\}$. Consequently, these are samples that mimic the times that would be produced by tasks that take around 40000 clock cycles to be executed, but which are subject to timing variability sources, such as hardware or control flow effects, that cause them to adhere to the GEV distribution with different shape values (LIMA; DIAS; BARROS, 2016). Through the analysis of the GEV synthetic samples, we evaluate whether variations in the underlying data shape could lead to conclusions that differ from those obtained on the real-hardware samples' analysis. Figure 12(a) illustrates the density distributions of a subset of the GEV synthetic samples we employ.

Since GEV synthetic samples require the threshold parameter value to be properly chosen in order to apply EVT using the POT approach – which is subject to characteristics of the method employed for that –, we also use a set of GP synthetic samples. The GP synthetic samples adhere to the GP distribution with threshold $\tau = 40000$ and scale $\sigma = 100$, and each assumes a different shape such that $\xi \in \{-\frac{1}{2}, -\frac{1}{4}, -\frac{1}{8}, 0, +\frac{1}{8}, +\frac{1}{4}, +\frac{1}{2}\}$. Consequently, these datasets simulate distribution tails with the same shapes of those of the GEV synthetic samples (COLES, 2001), but for which the ideal threshold to be used for the GP and Exponential models' adjustment is known beforehand. Through the analysis of the GP synthetic samples we show that our conclusions hold even when the correct POT parameters are available, i.e. independently of the threshold selection approach employed. Figure 12(b) illustrates the distributions of several GP synthetic samples we employ in this work.

Figure 12: Synthetic samples' distributions



(a) GEV synthetic samples



(b) GP synthetic samples

5.3 EXPERIMENT OBJECTIVES

We use the collected datasets, and the artefacts produced based on them through the proposed experiment design, to perform the evaluations presented in the following sections.

5.3.1 pWCET HWM Reliability

We assess whether the pWCET estimates derived through EVT using the GEV and the Gumbel models (for BM) and the GP and Exponential models (for the POT approach) can be deemed reliable when compared to the highest values effectively observed on large samples. For that we plot the pWCET estimates with exceedance probability 10^{-15}, and their associated 95% confidence intervals, and highlight the HWM observed in a sample of size 10^8 as an horizontal line. For the pWCET estimates being considered reliable, we expect them not to be exceeded by the effectively observed HWM, since the magnitude of the considered validation sample is much (i.e. seven orders) smaller than that of the pWCET exceedance probability used.

5.3.2 pWCET Density Reliability

We also observe whether the density of execution times that exceed pWCET estimates, whose exceedance probabilities' magnitudes are smaller than those of the large validation samples' sizes, can be deemed compatible with their associated exceedance probabilities. In such cases the observation of values that exceed the estimates are in fact expected, but the exceedance probability must be respected in the long term for pWCETs being considered potentially reliable.

The rationale behind this comparison is that, when we derive a pWCET with exceedance probability p, we are in fact looking for an execution time value w for which the expected probability (and hence density) of execution times that exceed w is closely approximated (or ideally upper-bounded) by p in the long term. Consider, for instance, we derive a pWCET w_t for a given task t intended to be exceeded with a maximum probability $p_{w_t} = 10^{-5}$. In this case, the number of execution times larger than w_t expected to be observed in a sample of $n_{t_1} = 10^5$ execution times of t equals $n_{t_1} \cdot p_{w_t} = 1$, that is, exceedances of w_t are expected in average once in every a hundred thousand executions of t. Supposing the sample size is extended e.g. to $n_{t_2} = 10^6$, one should expect in average $n_{t_2} \cdot p_{w_t} = 10$ exceedances being then observed. Large positive deviations from this expectation, especially as replications are performed and sample sizes are further increased, can be deemed incompatible with pWCETs' expected upper-bounding behaviour. Under this rationale, and taking into account that WCET underestimation is considered unacceptable for critical RTSs, not observing such upper-bounding for pWCETs with relatively high exceedance probability (e.g. 10^{-7}) raises doubts regarding the reliability of estimates associated with lower probabilities (e.g. 10^{-15}).

This new evaluation approach is justified by the fact that arbitrarily large execution times (HWMs) that exceed pWCETs may in fact occur, under certain conditions, without necessarily harming their target exceedance probabilities (even extremely low ones). Consider, for instance, a task t executing on a dual-core hardware platform that employs a shared memory bus arbitrated using a purely random policy. Also consider a pWCET estimate w_t of exceedance probability p_{w_t} is derived for task t. In a fair random arbitration policy all clients (cores) have the same probability of being chosen next for accessing the shared resource (memory bus), and since decisions are taken independently the same probabilities apply for every arbitration round. Despite the probability of a certain client never being chosen quickly converges to zero as execution proceeds, the exact zero probability is never reached. In other words, an extremely low but non-null probability exists that any one of the clients starves (temporarily or even permanently) while attempting to access the shared resource. Such an event could cause a HWM higher than w_t being yielded purely by chance, which could only be deemed to harm the reliability of w_t if effectively associated to a probability higher than p_{w_t}. However, the probability of repeatedly witnessing execution times larger than w_t by chance is substantially lower than that of witnessing a single of such exceeding HWMs by chance. Consequently, the comparison of exceedance densities can be considered a more trustworthy method for evaluating pWCET reliability than the comparison of estimates against HWMs, for being associated with a lower probability of drawing erroneous conclusions due to execution times observed purely by chance.

Similarly to the performed in traditional statistical hypothesis tests, the herein proposed method is applied by (A) assuming as true the basis hypothesis that the evaluated pWCET is in fact reliable, i.e. is only exceeded with its intended target probability, and (B) looking for empirical evidence that can be deemed sufficient for rejecting the basis hypothesis. The first step in applying it is calculating the Exceedance Density Metric (EDM) given by $edm = \frac{e}{n \cdot p}$, where e is the number of execution time values in a large sample of size n that exceed the evaluated pWCET estimate whose intended exceedance probability is p. EDM values are expected to be either close to or below the reference value $edm = 1$, respectively meaning that the empirical exceeding densities are either well-approximated or upper-bounded by the adjusted probabilistic model. More specifically, (A) whenever $edm \leq 1$ is obtained one should conclude that no empirical evidence exist at all to reject the pWCET reliability hypothesis, (B) if $edm \gg 1$ (e.g. $edm \approx 5$) is observed then clear unreliability evidence can be considered to exist, but (C) whenever $edm \gtrsim 1$ is obtained it does *not* necessarily mean sufficient evidence of unreliability exists, since execution times' randomness may cause such outcomes to be yielded purely by chance.

It is, however, possible to calculate the probability of a certain EDM value being produced purely by chance, taking as ground truth that the evaluated pWCET estimate is reliable – to which we refer simply as the ε probability (see Section 5.4). The ε probability can then be used to assess an obtained EDM value and for taking the decision, with increased confidence, on whether it is a sufficient evidence for rejecting the pWCET reliability hypothesis. Whenever relatively high values are obtained (e.g. $\varepsilon \gtrapprox 0.01$), further evaluations e.g. using replications or other evaluation methods should be performed for increasing confidence on the conclusions. If, on the other hand, low probabilities are obtained (e.g. $\varepsilon \lessapprox 10^{-7}$), one can confidently conclude that strong evidence exists that the evaluated pWCET is not reliable. The rationale is that, if it is extremely unlikely that an EDM as high as the observed one is yielded assuming that the pWCET is reliable, the evaluated pWCET should then have been exceeded with a probability higher than the intended one, being hence potentially unreliable. With that we ensure that a high confidence can be associated with the drawn conclusions, since the pWCETs' reliability hypothesis is only rejected under the demonstration of solid statistical significance. This approach resembles that of classical statistical tests, in which the tested hypothesis is only rejected if the probability of false negative errors, given as p-values (WASSERSTEIN; LAZAR, 2016), is deemed sufficiently low (STEPHENS, 2009).

5.4 THE PROBABILITY OF REJECTING A RELIABLE pWCET

A statistical experiment is called a binomial experiment if (A) it consists of n independent trials, (B) each trial can lead only to two possible outcomes: a success or a failure, and (C) the probabilities of a success and of a failure are given respectively by p and q, and are such that $p + q = 1$. In a binomial experiment, the probability of exactly k successes being observed in n trials, supposing p and q are in fact respected, can be calculated using the probability mass function of the binomial distribution given by $pmf(k; n, p) = \binom{n}{k} p^k (1 - p)^{n-k}$. Similarly, the probability of at most k successes being observed in n trials can be calculated using its cumulative distribution function $cdf(k; n, p) = \sum_{i=0}^{\lfloor k \rfloor} pmf(i; n, p)$ (FELLER, 1968).

The pWCET exceedance problem can be modelled as a binomial experiment, in which each trial is associated with an individual execution of the task and is considered successful if and only if the pWCET estimate *is* exceeded in that particular execution. The experiment's success probability hence equals the pWCET's intended exceedance probability p, and an execution time sample is then seen as a set of n independent trials of the experiment. It then follows that the ε probability, i.e. the probability of e pWCET exceedances being observed purely by chance in a sample of n execution times assuming the exceedance probability p is indeed respected, can be calculated through $pmf(e; n, p)$. Additionally, the probability of at most e pWCET exceedances being observed purely by chance in the same conditions can be calculated through $cdf(e; n, p)$.

This enables, e.g., the derivation of the probabilities with which pWCET estimates that are in fact reliable can be erroneously rejected while applying the pWCET reliability evaluation methods proposed in this work. The following sections present this and other relevant properties derived based on the binomial experiment modelling.

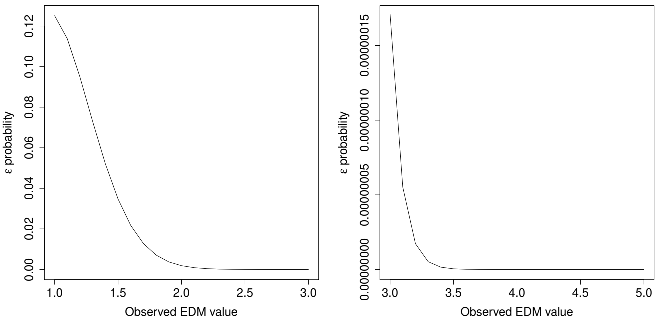
5.4.1 HWM-based method

The proposed HWM-based reliability evaluation method is applied by comparing the maximum values observed in execution time samples of size $n = 10^8$ against pWCET estimates whose target exceedance probability is $p = 10^{-15}$. The ε probability of observing at least one exceedance in this scenario, assuming the evaluated pWCET is in fact reliable, can be calculated as $\varepsilon = 1 - pmf(0; 10^8, 10^{-15}) \approx 10^{-7}$. In other words, the HWM-based reliability test proposed in this work leads to incorrect conclusions regarding pWCETs' unreliability with probability $\varepsilon = 10^{-7}$. Especially under replications, the observation of HWMs that exceed the evaluated pWCETs in the proposed experiment should hence be regarded as unreliability evidence (MAYS, 2010).

5.4.2 EDM-based method

Several relevant facts can be derived regarding the proposed EDM-based reliability evaluation method as considered in this work, i.e. using validation samples of size $n = 10^8$ to evaluate pWCET estimates of target exceedance probability $p = 10^{-7}$. For instance, the probability of observing at least one exceedance under such conditions can be calculated as $1 - pmf(0; 10^8, 10^{-7}) \approx 99,99\%$, indicating that exceedances should be observed while applying the method if the considered pWCETs are in fact exceeded with their intended probabilities. This also means that non-exceeded pWCETs are likely to present exceedance probabilities lower than 10^{-7}, being potentially both reliable and pessimistic. Moreover, the probability of observing more than $n \cdot p = 10$ exceedances in the same context is calculated as $1 - cdf(10; 10^8, 10^{-7}) \approx 42\%$. Consequently, even if the evaluated pWCETs' target probabilities are in fact respected, it is likely that EDM values higher than the reference $edm = 1$ will be often observable. However, the probability of EDM values larger than $edm = 1$ being observed purely by chance quickly converges to zero. This is evidenced in Figure 13, which shows the ε probabilities associated with the observation of all possible EDM values in the range $[1, 5]$ in the scenario considered in this work (i.e. with $n = 10^8$ and $p = 10^{-7}$), assuming that the evaluated pWCET is reliable (i.e. purely by chance).

Figure 13: ε probability analysis

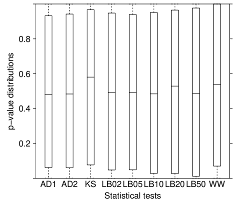


One can then easily show that the proposed EDM-based pWCET reliability evaluation method achieves higher confidence levels in comparison with the alternative HWM-based approach. In the herein considered scenario, the EDM-based method's ε probability for an EDM value as high as e.g. $edm = 3.1$ being yielded purely by chance can be calculated as $\varepsilon = pmf(31; 10^8, 10^{-7}) < 5.53 \cdot 10^{-8}$. Consequently, if it yields $edm \geq 3.1$ one can decide with confidence higher than that enabled by the HWM-based method (whose $\varepsilon \approx 10^{-7}$) on rejecting the reliability hypothesis for the evaluated pWCET estimate.

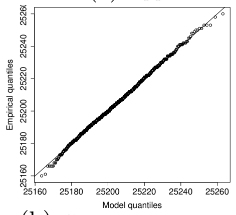
5.5 APPLICABILITY EVIDENCE

The EVT applicability diagnostic artefacts (as of Chapter 2) associated to the evaluated real-hardware sample are shown in Figure 14, which presents (1) a box and whisker plot of the p-values yielded by the independence and identical distribution statistical hypothesis tests, and (2) quantile and probability plots built by fitting a modelling sample of size 50000 to the GEV, GP, Gumbel and Exponential distributions. From the analysis of these artefacts one can conclude that (I) all statistical tests' results are acceptable, since the produced p-values are approximately uniformly distributed in the $[0, 1)$ range, (II) the GEV and GP models adjusted to the maximum observed execution times are capable of properly representing their behaviour, and (III) the Gumbel and Exponential models do not represent well the maxima behaviour, but lead to potentially reliable upper bounds for consistently overestimating high quantiles. The 95% confidence interval produced through the GEV and GP model fitting indicated that the sample's tail shape is approximately $-0.139 \leq \xi \leq -0.055$ for GEV and $-0.154 \leq \xi \leq -0.125$ for GP. We can hence conclude with confidence higher than 95% that its right tail shape is negative (i.e. $\xi < 0$), due to the fact the confidence intervals' upper estimates are significantly smaller than zero considering the typical variation ranges of the shape parameter.

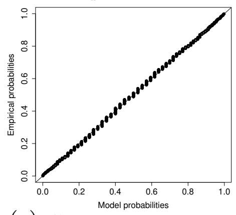
Figure 14: Real-hardware sample applicability evidence



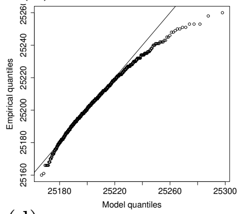
(a) Applicability statistical tests' p-values



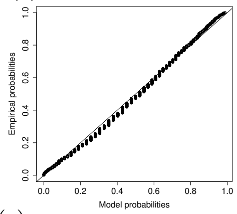
(b) GEV quantile plot



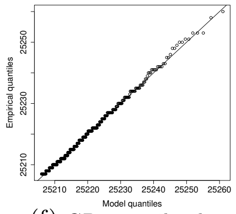
(c) GEV probability plot



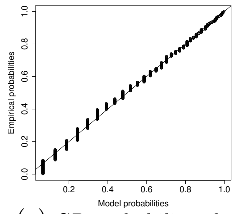
(d) Gumbel quantile plot



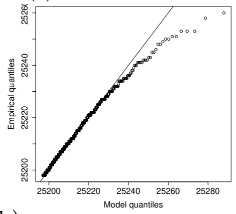
(e) Gumbel probability plot



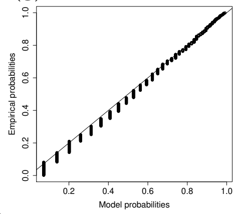
(f) GP quantile plot



(g) GP probability plot



(h) Exponential quantile plot



(i) Exponential probability plot

Acceptable results were also yielded for the synthetic samples by both the independence and identical distribution statistical hypothesis tests and the model adherence evaluation plots. This was expected, since (1) independence and identical distribution are requirements for random numbers that adhere to a specific probability distribution (KNUTH, 1997; MARSAGLIA; TSANG, 2002), and (2) data that adhere to the GEV distribution present tails that are known to adhere to the GP model with the same shape value (COLES, 2001). For this reason, we suppress the synthetic samples' applicability diagnostic artefacts.

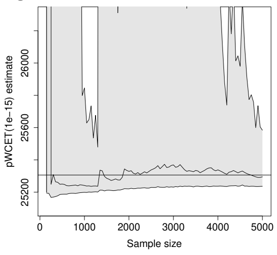
5.6 pWCET HWM RELIABILITY

In this section we evaluate the HWM reliability assessment plots for the selected real-hardware sample and for the synthetic samples, for both the GEV/Gumbel (BM) and the GP/Exponential (POT) models. The plots show the behaviour of the pWCET estimates with exceedance probability of 10^{-15}, and their associated 95% confidence intervals (grey region), as modelling sample sizes are increased from 150 (i.e. 3 sampling steps) to 5000 (i.e. 100 sampling steps) in steps of 50 measurements. The horizontal line indicates the HWMs, i.e. the maximum values, that were observed in the validation samples of size 10^8.

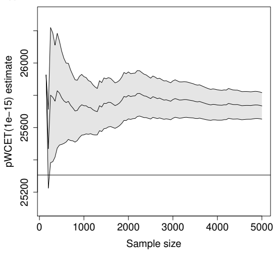
5.6.1 Real-Hardware Samples

Figures 15(a) and 15(c) show that the pWCET estimates yielded by the GEV and GP models consistently converge to values close or lower than the 10^8 HWM, a result that can be deemed as unreliability evidence considering the large difference between the magnitudes of the estimates' exceedance probability and of the considered validation sample size (i.e. seven orders). On the other hand, Figures 15(b) and 15(d) show that, for the Gumbel and Exponential models, all pWCET(10^{-15}) and their associated confidence intervals stay consistently above the 10^8 HWM for reasonable sample sizes, being hence deemed potentially reliable.

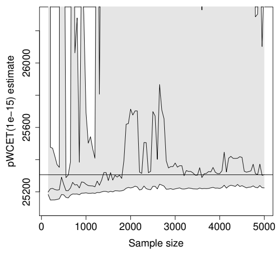
Figure 15: Real-hardware pWCET HWM reliability



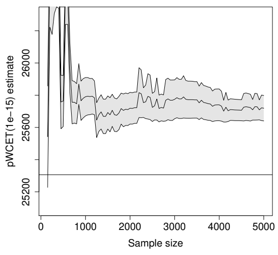
(a) GEV



(b) Gumbel



(c) GP



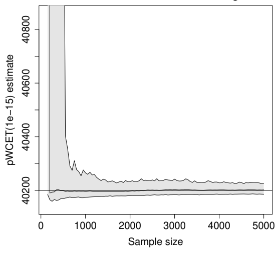
(d) Exponential

5.6.2 Synthetic Samples

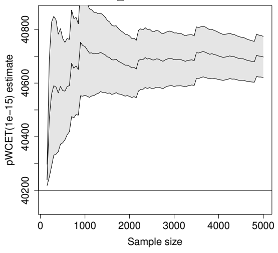
Figure 16 presents the HWM reliability assessment plots for the synthetic samples with $\xi = -\frac{1}{2}$. Based on their analysis, one can observe that the GEV and GP models' reliability can be considered doubtful, since (1) the pWCET(10^{-15}) estimates are very close and often slightly smaller than the observed 10^8 HWM even after convergence, and (2) the HWM is within the pWCETs' confidence intervals, indicating the correct pWCET value could in fact be lower than the HWM. On the other hand, the Gumbel and Exponential models present no unreliability evidence, for providing pWCET estimates which are, together with their associated confidence intervals, consistently higher than the highest values observed in samples of 10^8 measurements.

Figure 16: pWCET HWM reliability for $\xi = -\frac{1}{2}$

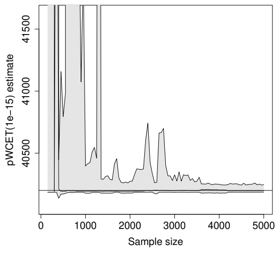
GEV synthetic sample



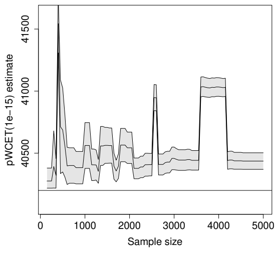
(a) GEV



(b) Gumbel

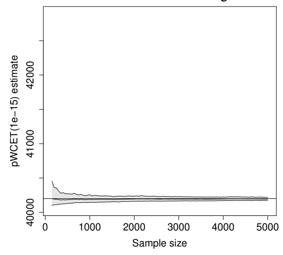


(c) GP

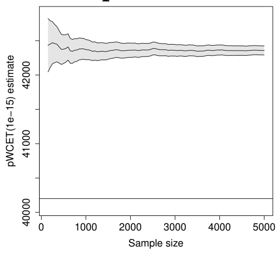


(d) Exponential

GP synthetic sample



(e) GP

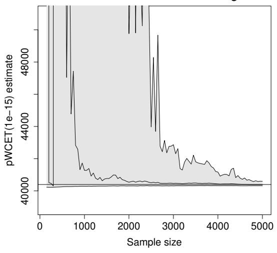


(f) Exponential

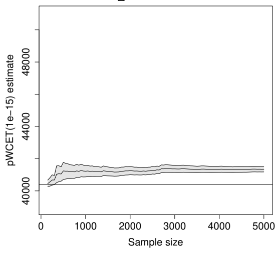
Figures 17 and 18 present HWM reliability assessment plots for the synthetic samples with $\xi = -\frac{1}{4}$ and $\xi = -\frac{1}{8}$, respectively. The conclusions drawn from these plots' analysis are similar to those obtained for the $\xi = -\frac{1}{2}$ case, but it is remarkable that the pWCET(10^{-15}) estimates provided by the GEV and GP models still converge to values very close to – and are indeed often lower than – the 10^8 HWM as the underlying data shape value is increased. Such behaviour is not expected, due to the large difference between the magnitudes of the estimates' exceedance probability and of the validation samples' size. One should expect instead the pWCET(10^{-15}) estimates to be consistently higher than the HWM observed for a sample of size 10^8, which is in fact observable for both the Gumbel and the Exponential models.

Figure 17: pWCET HWM reliability for $\xi = -\frac{1}{4}$

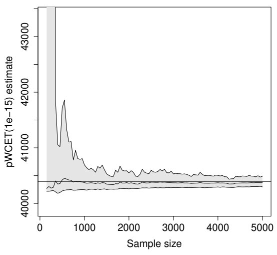
GEV synthetic sample



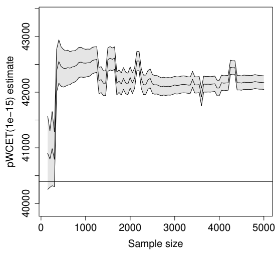
(a) GEV



(b) Gumbel

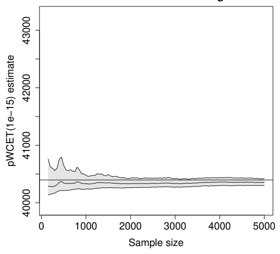


(c) GP

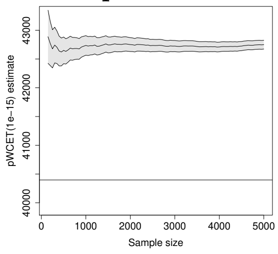


(d) Exponential

GP synthetic sample



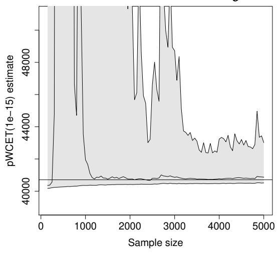
(e) GP



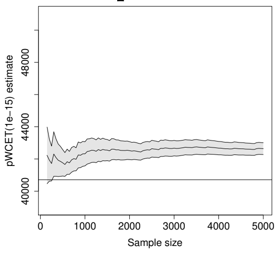
(f) Exponential

Figure 18: pWCET HWM reliability for $\xi = -\frac{1}{8}$

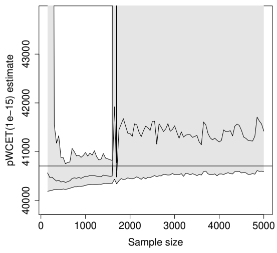
GEV synthetic sample



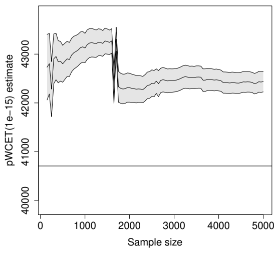
(a) GEV



(b) Gumbel

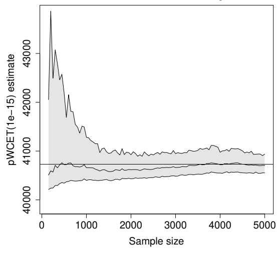


(c) GP

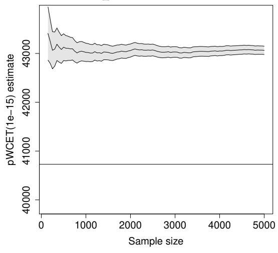


(d) Exponential

GP synthetic sample



(e) GP

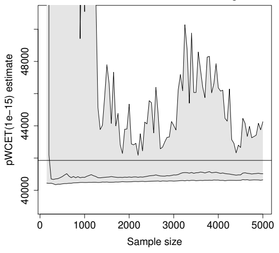


(f) Exponential

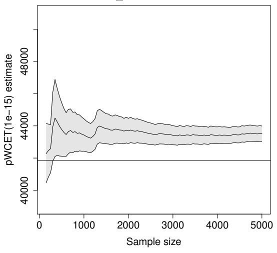
Figure 19 presents the HWM reliability assessment plots for the synthetic sample with $\xi = 0$, in which the analysed maxima is known to adhere to the Gumbel distribution and, therefore, tails are known to adhere to the Exponential distribution. Based on their analysis, one can conclude that the GEV and GP models produced what could be regarded as clearly unreliable results, since the 10^8 HWM observed on the validation sample consistently exceeds the yielded pWCET(10^{-15}) estimates and are contained within their confidence intervals. On the other hand, the Gumbel and Exponential models produced results that can be deemed reliable, since the pWCETs and their associated confidence intervals remain consistently above the same HWM.

Figure 19: pWCET HWM reliability for $\xi = 0$

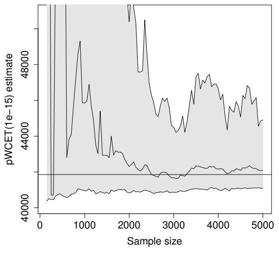
GEV synthetic sample



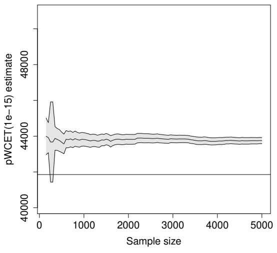
(a) GEV



(b) Gumbel

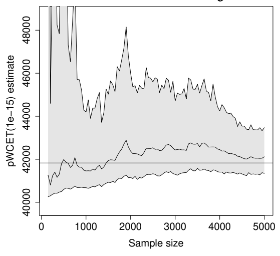


(c) GP

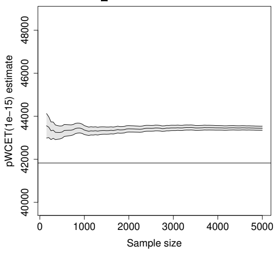


(d) Exponential

GP synthetic sample



(e) GP

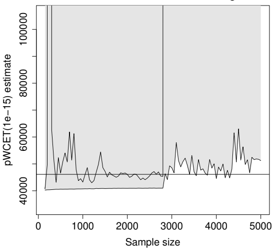


(f) Exponential

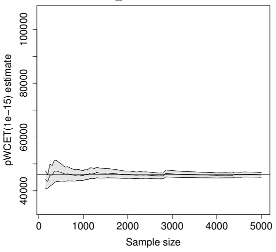
Figures 20, 21 and 22 present the HWM reliability assessment plots for the synthetic samples with $\xi \in \{+\frac{1}{8}, +\frac{1}{4}, +\frac{1}{2}\}$, respectively. The estimates derived through GEV and GP in these cases proved either (1) unreliable or doubtfully reliable, for being often exceeded or for being associated to confidence intervals that contain the HWMs observed on the validation samples, or (2) useless in practice, for providing values that are extremely higher than typical WCET safety margins – i.e. 20% (CAZORLA et al., 2016). The Gumbel and Exponential models, which are in fact not applicable if $\xi > 0$, also produced unreliable results but still provided more consistent and stable pWCET estimates in comparison with GEV and GP. Despite also being useless in practice, the consistent estimates yielded by the Gumbel and Exponential models make it easier to detect cases in which their application cannot be deemed reliable, e.g. through the methods proposed in this work.

Figure 20: pWCET HWM reliability for $\xi = +\frac{1}{8}$

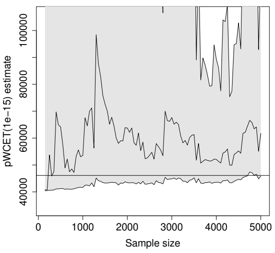
GEV synthetic sample



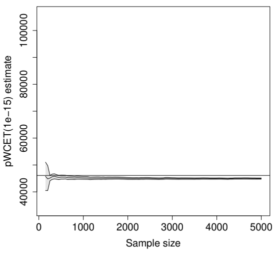
(a) GEV



(b) Gumbel

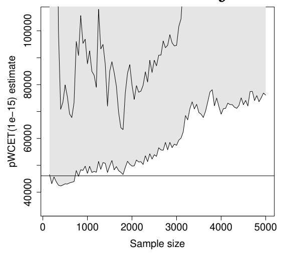


(c) GP

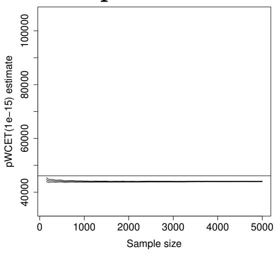


(d) Exponential

GP synthetic sample



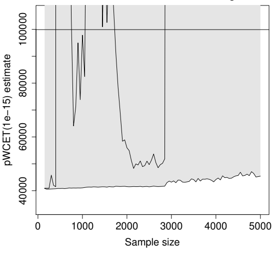
(e) GP



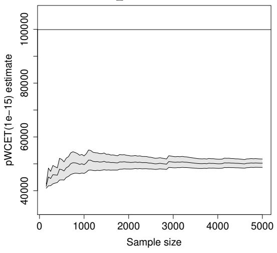
(f) Exponential

Figure 21: pWCET HWM reliability for $\xi = +\frac{1}{4}$

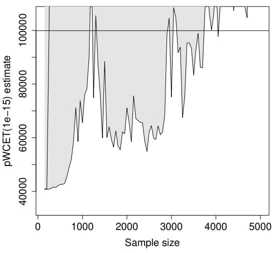
GEV synthetic sample



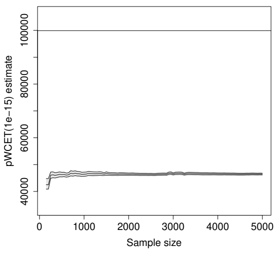
(a) GEV



(b) Gumbel

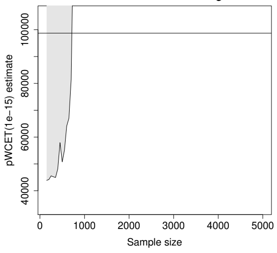


(c) GP

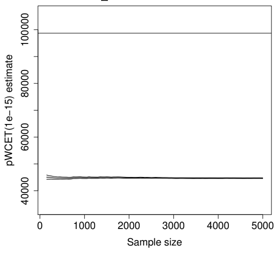


(d) Exponential

GP synthetic sample



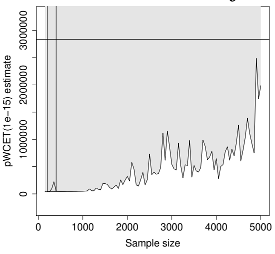
(e) GP



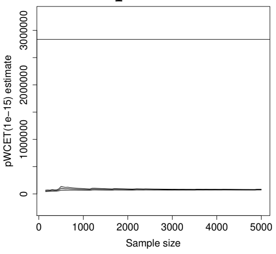
(f) Exponential

Figure 22: pWCET HWM reliability for $\xi = +\frac{1}{2}$

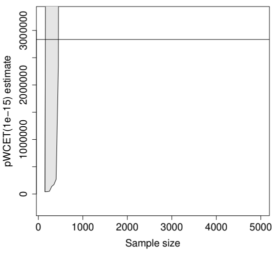
GEV synthetic sample



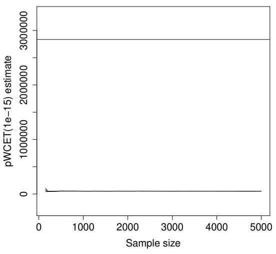
(a) GEV



(b) Gumbel

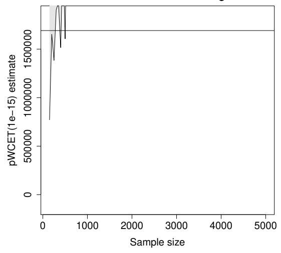


(c) GP

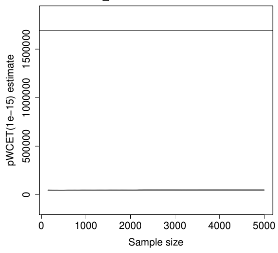


(d) Exponential

GP synthetic sample



(e) GP



(f) Exponential

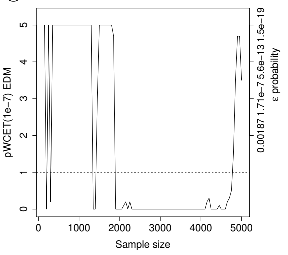
5.7 pWCET DENSITY RELIABILITY

This section presents an evaluation of the density reliability assessment plots produced based on the selected real-hardware sample and on the synthetic samples, for both the BM (GEV/Gumbel) and the POT (GP/Exponential) probabilistic models. The plots show the behaviour of the EDM, as described in Section 5.3.2, for pWCET estimates associated with a target exceedance probability of 10^{-7}, as modelling sample sizes are increased from 150 to 5000 in steps of 50 measurements (i.e. from 3 to 100 sampling steps). The EDM value is limited to the range $[0, 5]$ for improving plots' readability, the horizontal dotted line ($edm = 1$) indicates the reference value which the metric is intended to be compared with, and the ε probabilities associated with the corresponding EDM values are also indicated.

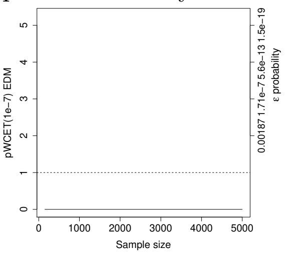
5.7.1 Real-Hardware Samples

Figures 23(a) and 23(c) show that the pWCET estimates produced using either the GEV or the GP models are often exceeded, even for relatively large modelling sample sizes, with densities many (at least five) times larger than the expected taking into account their associated exceedance probabilities. An analysis of the associated ε probabilities reveals that such high EDM values are extremely unlikely, i.e. their observation is associated with probabilities as low as $1.5 \cdot 10^{-19}$, assuming the evaluated pWCET estimates are in fact reliable (i.e. exceeded only with the intended probability). Consequently, one should consider that unreliability evidence exist for pWCET estimates derived through the GEV and GP models. On the other hand, Figures 23(b) and 23(d) show that, under the same conditions, the pWCETs produced through the Gumbel and Exponential models consistently upper-bound all execution time values observed in the large validation sample, since EDM equals zero for all considered modelling sample sizes. We hence consider unreliability evidence exists for pWCET estimates produced using the GEV and GP models, which are not observable when the Gumbel and Exponential models are used to the same purpose.

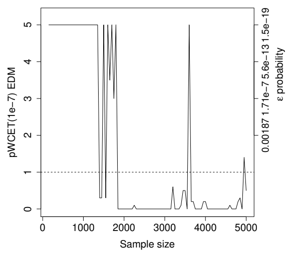
Figure 23: Real-hardware pWCET density reliability



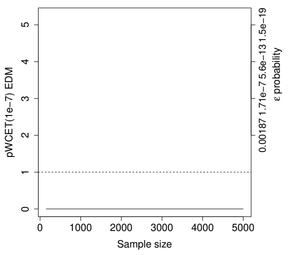
(a) GEV



(b) Gumbel



(c) GP



(d) Exponential

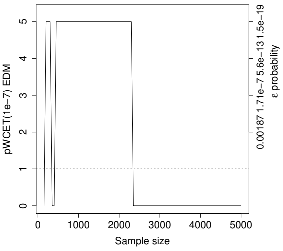
5.7.2 Synthetic Samples

The density reliability plots associated to GEV and GP pWCET estimates, shown in Figures 24 to 30, indicate that the EDM behaviour varies significantly for pWCET estimates derived through the GEV and GP models. There are cases in which the EDM reaches unacceptably high values (A) eventually, as in Figures 25(a), 26(a), 28(e) and 29(e), (B) frequently, as in Figures 24(a), 24(c), 25(c), 26(c), 27(c), 28(a), 28(c), 29(a), 29(c) and 30(e), and even (C) consistently, as in Figures 24(e), 25(e), 26(e), 27(a) and 27(e). The only cases in which the EDM behaviour for the GEV and GP models could be deemed acceptable was for the GEV synthetic samples with $\xi = +\frac{1}{2}$, shown in Figures 30(a) and 30(c), for quickly converging to values nearby or lower than the reference value. However, in this specific case the yielded pWCETs often prove useless in the context of MBPTA for leading to extremely pessimistic bounds (see Figure 22). We highlight the case of the samples with shape $\xi = 0$, shown in Figure 27, in which most EDM values reach the plots' upper limit (i.e. $edm = 5$). The ε probabilities' analysis shows that such EDM values are extremely improbable assuming the evaluated pWCETs are in fact reliable, i.e. are associated with probabilities as low as $1.5 \cdot 10^{-19}$, indicating that it is highly unlikely that such high exceedance densities occurred purely by chance. Such behaviour can hence be deemed as unreliability evidence, since it indicates the pWCET estimates produced through the GEV and GP models are often exceeded with densities that largely diverge from those expected when their target exceedance probabilities are respected.

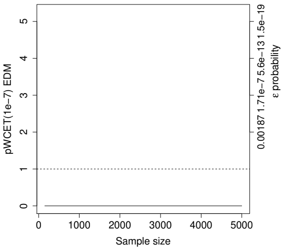
On the other hand, the density reliability plots associated to Gumbel and Exponential pWCET estimates indicate null exceedance densities are consistently observable for all synthetic samples with tail shapes $\xi < 0$, as of Figures 24, 25 and 26. Conversely, for samples whose tails present shape $\xi > 0$ the pWCETs' exceedance densities prove consistently unacceptable, as shown in Figures 28, 29 and 30. Both conclusions were expected, since the Gumbel and Exponential models are in fact known (A) to upper-bound the densities of maxima tails with shape $\xi < 0$, and (B) to be not applicable for maxima tails with shape $\xi > 0$ (COLES, 2001). Moreover, nearly-ideal EDM behaviour was observed for synthetic samples with tail shape $\xi = 0$, which are known to be modellable using the Gumbel/Exponential distributions for presenting tails with exponentially decreasing slopes. As shown in Figure 27, for these samples the EDM is either within acceptable ranges for reasonable modelling sample sizes or presents a clear convergence pattern to the reference value $edm = 1$. Such consistent behaviour hence provides evidence that, for samples whose maxima present tails with shape $\xi \leq 0$, the Gumbel and Exponential models yield pWCET estimates that can be deemed potentially reliable.

Figure 24: pWCET density reliability for $\xi = -\frac{1}{2}$

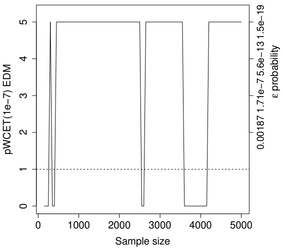
GEV synthetic sample



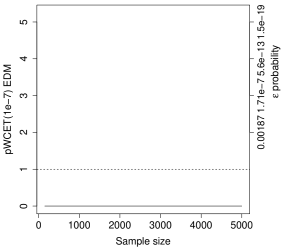
(a) GEV



(b) Gumbel

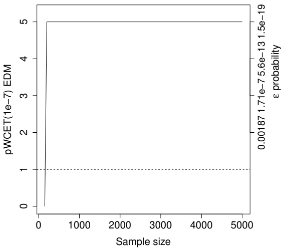


(c) GP

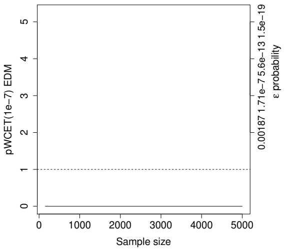


(d) Exponential

GP synthetic sample



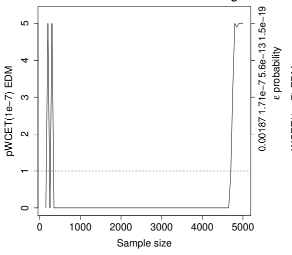
(e) GP



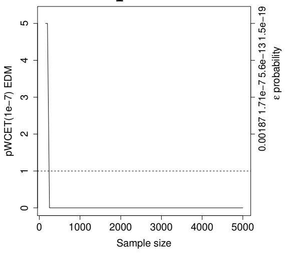
(f) Exponential

Figure 25: pWCET density reliability for $\xi = -\frac{1}{4}$

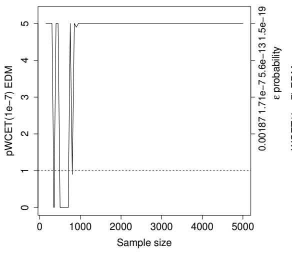
GEV synthetic sample



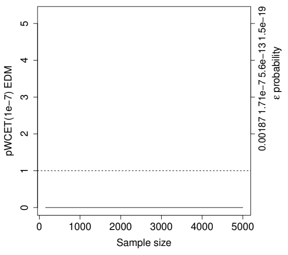
(a) GEV



(b) Gumbel

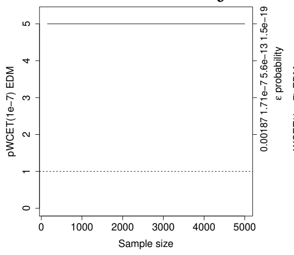


(c) GP

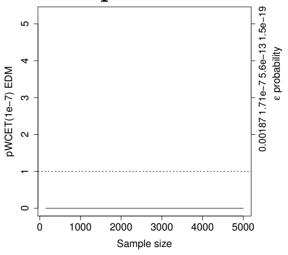


(d) Exponential

GP synthetic sample



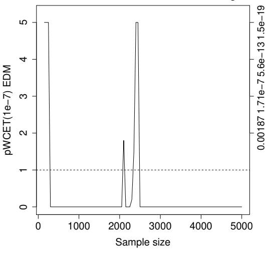
(e) GP



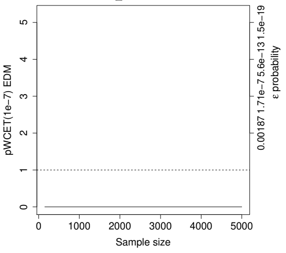
(f) Exponential

Figure 26: pWCET density reliability for $\xi = -\frac{1}{8}$

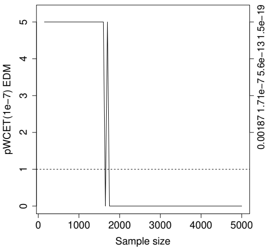
GEV synthetic sample



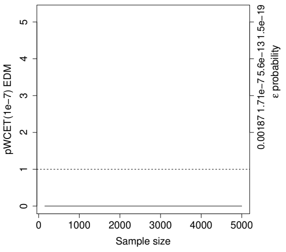
(a) GEV



(b) Gumbel

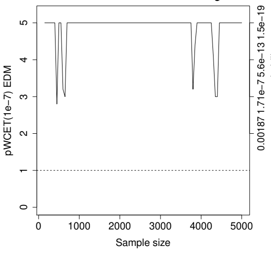


(c) GP

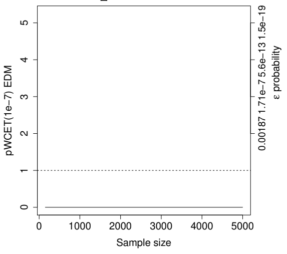


(d) Exponential

GP synthetic sample



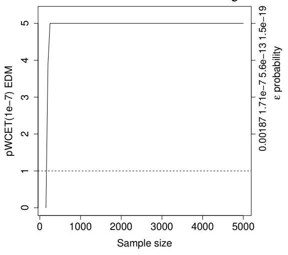
(e) GP



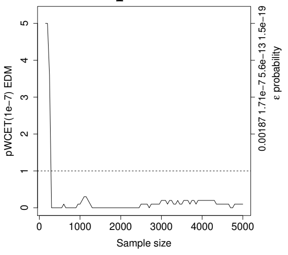
(f) Exponential

Figure 27: pWCET density reliability for $\xi = 0$

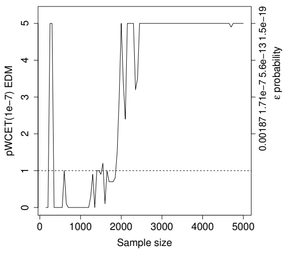
GEV synthetic sample



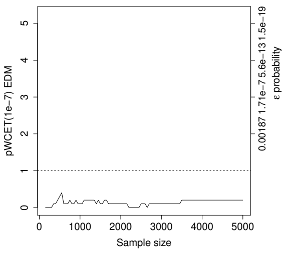
(a) GEV



(b) Gumbel

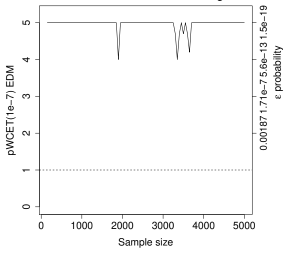


(c) GP

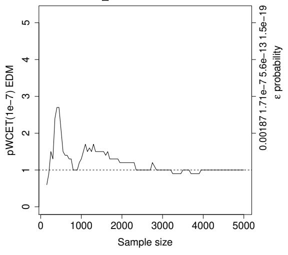


(d) Exponential

GP synthetic sample



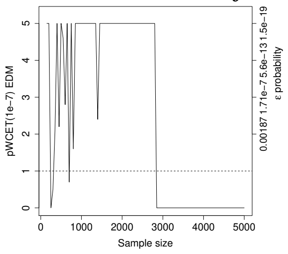
(e) GP



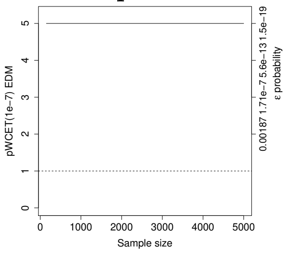
(f) Exponential

Figure 28: pWCET density reliability for $\xi = +\frac{1}{8}$

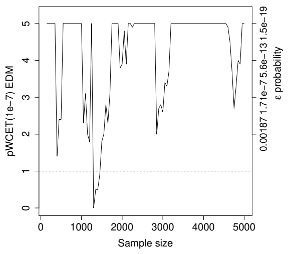
GEV synthetic sample



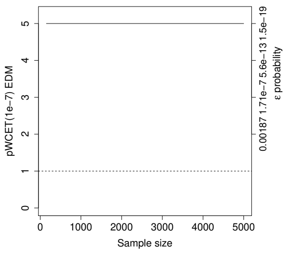
(a) GEV



(b) Gumbel

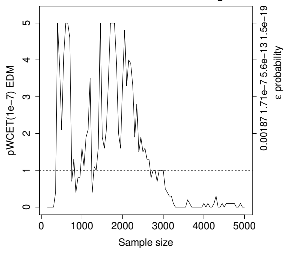


(c) GP

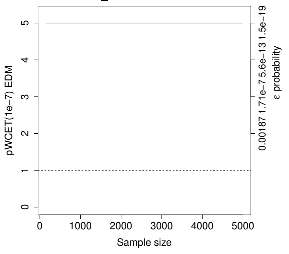


(d) Exponential

GP synthetic sample



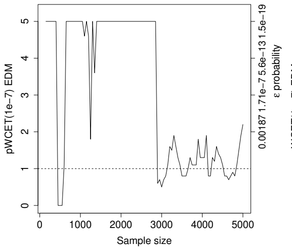
(e) GP



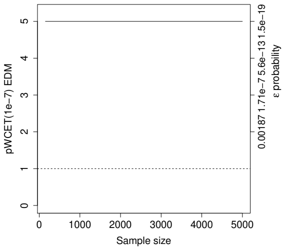
(f) Exponential

Figure 29: pWCET density reliability for $\xi = +\frac{1}{4}$

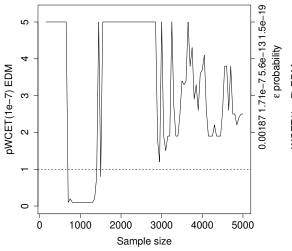
GEV synthetic sample



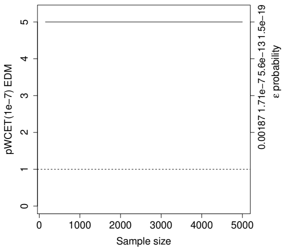
(a) GEV



(b) Gumbel

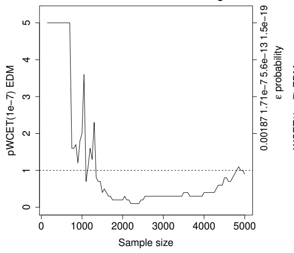


(c) GP

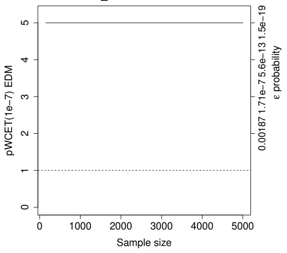


(d) Exponential

GP synthetic sample



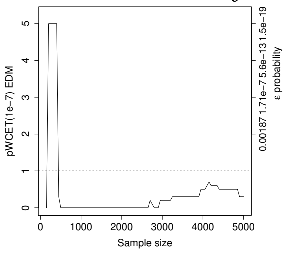
(e) GP



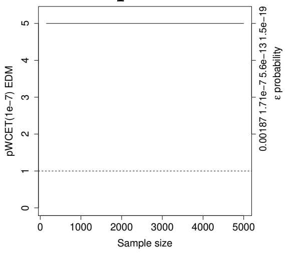
(f) Exponential

Figure 30: pWCET density reliability for $\xi = +\frac{1}{2}$

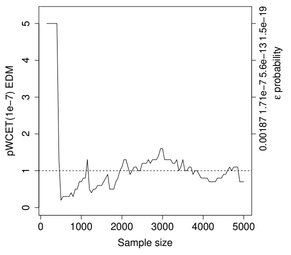
GEV synthetic sample



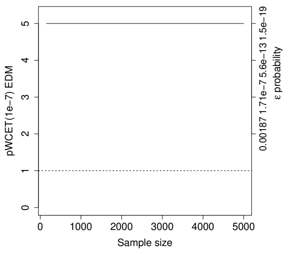
(a) GEV



(b) Gumbel

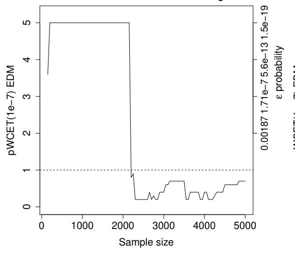


(c) GP

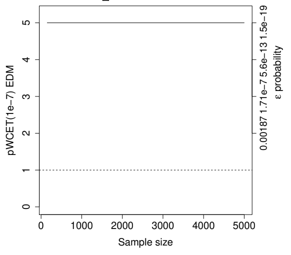


(d) Exponential

GP synthetic sample



(e) GP



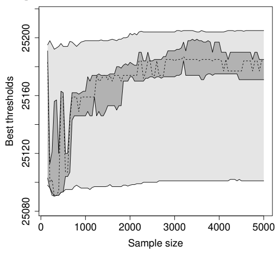
(f) Exponential

5.8 EQMAE-BASED POT THRESHOLD SELECTION

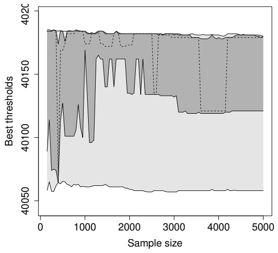
All the previously presented POT-related reliability assessment plots were produced by fitting the GP and Exponential models to values exceeding thresholds determined through the minimization of the EQMAE metric, as described in Section 2.4. In this section we present a brief evaluation of this approach considering a threshold selection method as adequate if, as the modelling sample size is increased, (1) the chosen threshold converges, (2) the range in which the best threshold candidate values are contained also presents convergence pattern, and especially (3) the produced pWCET estimates are stable and reliable. For that, we present in Figure 31 a collection of threshold selection plots in which are shown, as modelling sample sizes are increased from 150 to 5000 in steps of 50 values, (1) in light grey the range of candidate threshold values, given by the 60% to 99% quantiles of the analysed data, (2) in dark grey the interval in which the ten best threshold candidates (taking EQMAE as criterion) are contained, and (3) the effectively chosen threshold as a dotted line.

From the analysis of these plots one can conclude that no convergence pattern is generally observable for the thresholds chosen through the EQMAE minimization method, since even for large sample sizes both the best candidates' interval and the chosen threshold often vary significantly with respect to the range of candidate values. Through a joint analysis of these artefacts with their associated reliability counterparts presented in Sections 5.6 and 5.7, we also observe that the pWCET estimates' variability is strongly correlated to threshold selection, which – as expected – makes it clear that the estimates are sensible to the threshold used. We highlight, however, that (1) no estimate derived through the Exponential model for reasonable modelling sample sizes with thresholds determined using the EQMAE minimization method could be deemed unreliable according to the assessments performed in this work, and (2) the GP model yields unreliable results even when the threshold value is known beforehand, as evidenced in our analysis using GP synthetic samples.

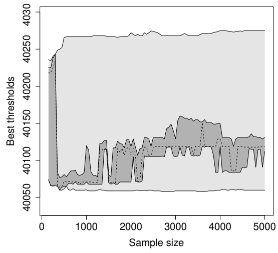
Figure 31: EQMAE-based threshold selection plots



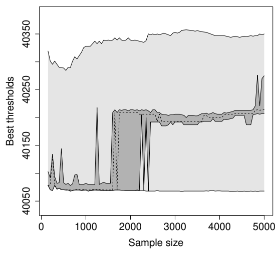
(a) *bsort* on *DPCpArrr*



(b) GEV with $\xi = -\frac{1}{2}$



(c) GEV with $\xi = -\frac{1}{4}$



(d) GEV with $\xi = -\frac{1}{8}$

5.9 RECOMMENDATIONS

We conclude from the assessments presented in the previous sections that the Gumbel and Exponential models seem consistent and stable with respect to the produced pWCET estimates, which gives confidence on the yielded results whenever their applicability conditions hold. On the other hand, the GEV and GP models appear to add a degree of uncertainty to the analysis process, often yielding unreliable pWCET estimates, hence reducing the confidence on the produced results. Our conclusions support the idea that only the Gumbel and Exponential models could be deemed acceptably reliable for sample sizes feasible in practical environments, i.e. in the order of hundreds (CUCU-GROSJEAN et al., 2012; WARTEL et al., 2015), and that therefore GEV and GP should not be used to derive pWCET estimates unless approaches for increasing confidence on the produced results becomes available. We also conclude that the POT threshold selection approach based on the EQMAE metric minimization, as implemented and evaluated in this work, does not appear to present convergence pattern as modelling sample sizes are increased. We hence recommend that:

- The shape parameter (ξ) of the observed data distribution tail should be estimated using a (set of) proper method(s), for which we suggest (1) fitting the GEV and/or GP models and assessing the shape parameter confidence interval, (2) fitting the Gumbel and/or Exponential models and assessing the diagnostic quantile plot – which should suggest high quantiles are not underestimated (see Section 5.5) –, and/or (3) employing the tail diagnostic method proposed in (ABELLA et al., 2017), and then:
  - If $\xi < 0$ (Weibull tail), the Gumbel and Exponential models introduce some pessimism but can apparently be used with reasonable confidence on the reliability of the yielded results.
  - If $\xi = 0$ (Gumbel tail), the Gumbel and Exponential models appear to be acceptably reliable but must be carefully used, e.g. associated to proper evidence on estimates' reliability.
  - If $\xi > 0$ (Fréchet tail), the Gumbel and Exponential models yield estimates that are consistent but clearly unreliable, since they are in fact not applicable to such scenarios. To our knowledge, no model capable of producing reliable pWCET estimates in such scenarios is currently available.

Additionally, we highlight that:

- Most of the real-hardware execution time samples we collected present tails that consistently adhere to the GEV and GP models with shape $\xi < 0$, hence being potentially analysable through the Gumbel and Exponential models with good confidence on the reliability of the produced pWCETs.
- pWCET confidence intervals' upper limits can be used to reduce the probability of faults due to parameter estimation errors only for the Gumbel and Exponential models, since for GEV and GP this could lead to huge pessimism in cases the shape parameter's upper limit exceeds zero (see e.g. Figure 18).
- The presented assessments were replicated using the POT methodology for applying EVT as proposed in (GUET; SANTINELLI; MORIO, 2016; SANTINELLI; GUET; MORIO, 2017), and the obtained results consistently corroborated all achieved conclusions.

- The works presented in (MILUTINOVIC et al., 2017; ABELLA et al., 2017) point out several reasons for which the Gumbel and Exponential models should be used for applying EVT within MBPTA, considering specific aspects of its phenomenon of interest (i.e. execution times). The evidence provided in this work add empirical arguments to support this recommendation, by showing that models that are supposed to be more precise may in fact lead to underestimations even under nearly-ideal conditions.

Finally, we highlight that the safety margins yielded using the GEV and GP models can be regarded as valid and quite precise as estimates, i.e. approximations that can be subject to both positive errors (overestimation) and negative errors (underestimation) in relation to the real value. This is evidenced in this work by the relative proximity of the analysed pWCET estimates to the effectively observed HWMs, especially for synthetic samples whose shape values lead to bounded tails (see e.g. Figures 16 and 17). However, the fact that underestimation is not acceptable while deriving pWCETs for building critical RTSs requires care regarding estimates that are prone to negative errors. Consequently, a high criticality becomes associated to the estimation of the shape parameter of GEV and GP models within MBPTA, since (A) underestimations can easily lead to unreliable pWCETs, and (B) overestimations can produce extremely pessimistic (useless in practice) pWCETs. On the other hand, the Gumbel and Exponential models do not require the estimation of a shape parameter, for presenting right tails that decrease exponentially and are therefore equivalent to GEV and GP with shape $\xi = 0$, respectively. As a consequence and in line with the herein presented analysis, Gumbel and Exponential can only be reliably applied for producing safety margins if the modelled maxima adhere to GEV and/or GP with shape $\xi \leq 0$. Moreover, the pessimism that arises from this approach for maxima with shape $\xi \ll 0$ appears to be still within acceptable amounts in comparison with typically employed safety margins. The drawn recommendations can hence be considered to effectively envision *upper-bounding the shape* of the analysed maxima, in order to increase reliability on the produced pWCETs, still providing reasonable confidence that no extreme pessimism is induced.

5.10 CONCLUSION

In the work presented in this chapter we have performed an empirical evaluation of the GEV/Gumbel and of the GP/Exponential models, with respect to the reliability of the pWCET estimates they yield when used for applying EVT in the context of MBPTA through the BM and the POT approaches, respectively.

For that, we employed a set of large execution time datasets (i.e. of size 10^8) of both real (i.e. measured on real hardware platforms) and synthetic (i.e. pseudo-randomly generated) nature. We then compared (A) the pWCET estimates produced through the assessed models with the highest values (i.e. HWMs) effectively observed in the large validation samples, and (B) the empirical exceedance densities observed in the validation samples against their expected values taking into account the pWCETs' target exceedance probabilities.

We observed that (1) the GEV and GP models lead to pWCET estimates often exceeded by large samples' maxima, and that, on the other hand, (2) the Gumbel and Exponential models produce potentially reliable upper-bounds in all evaluated cases to which they are expected to be applicable (i.e. when $\xi \leq 0$). In fact, GEV and GP estimates appear to be often exceeded with densities many (at least five) times larger than the expected according to the specified pWCET exceedance probabilities, while both the Gumbel and the Exponential models appear to consistently upper-bound empirical tails' densities when $\xi \leq 0$.

We hence conclude that GEV and GP do not present key characteristics required for probabilistic models employed in deriving pWCET estimates within MBPTA, and therefore recommend that the Gumbel and Exponential should instead be used for the purpose until the observed unreliable behaviour is better understood and handling methods become available. For that, we suggest properly diagnosing scenarios in which pWCET estimates can be reliably derived through the Gumbel and Exponential models, and rejecting the analysis or performing further reliability assessments whenever the necessary conditions do not clearly hold. Our objective with these recommendations is translating the outcomes of our research into measures that enable increasing confidence on the reliability of pWCET estimates produced through MBPTA.

6 EVALUATING RANDOMIZED SCHEDULING ON MULTITHREAD PIPELINES TO BENEFIT MBPTA

In this chapter we evaluate the hypothesis that randomized thread scheduling can benefit the timing analysis of tasks executed on multithread pipelines, by causing their execution times to meet MBPTA's basic application requirements. For that we employ a processing core equipped with a simple multithread pipeline, and then (A) derive the conditions in which execution time measurements can be taken under maximum inter-thread interference, (B) establish a method for evaluating key properties such processors are expected to present to support MBPTA's applicability, and (C) assess two distinct randomized thread scheduling approaches. Within (C) we evaluate both (C.1) a simple thread scheduler that employs a purely random policy, which meets MBPTA's basic requirements but does not balance inter-thread interference delays, and (C.2) a credit-based interference-regulated thread scheduler, capable of limiting interference timing effects without compromising execution times' analysability through MBPTA.

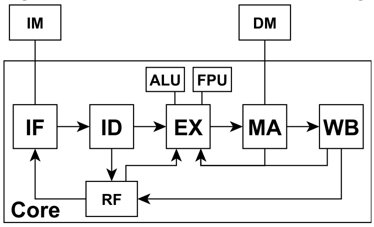
6.1 PIPELINING

In the context of computer processors, *pipelining* is a core implementation approach in which instructions' execution is divided into a set of stages, such that a new instruction may enter a stage as soon as the previous one leaves it. This enables the execution of multiple instructions being overlapped, hence allowing instruction-level parallelism to be exploited at some level and significantly increasing processing throughput (i.e. the frequency with which instructions are completed) in comparison to cores that execute instructions sequentially. Pipelining also enables increasing processors' clock frequencies, since it allows certain complex operations that require deep combinational logic being decomposed into simpler steps (GONZALEZ; LATORRE; MAGKLIS, 2010; PATTERSON; HENNESSY, 2011; HENNESSY; PATTERSON, 2012).

The application of pipelining techniques requires proper control logic being added, for guaranteeing instructions' correct execution. First, some operations executed by the stages take longer than others, e.g. multiplications may take longer than additions, which may require stages to block the previous ones until long operations complete. Moreover, some instructions use as input the information produced by previously executed ones (i.e. are subject to data dependencies), requiring stages being stalled (i.e. temporarily stopped) until the necessary data produced by the following stages is ready – such issues are known as *hazards*. Bypassing logic is also used between some stages for avoiding long stalls, e.g. by forwarding data read at the memory stage to the instruction at the execution stage that is possibly (and often) awaiting it to be available. This avoids a delay that would be necessary for the execution stage to read the memory word from the register to which the memory stage intends to write it (in load/store instruction sets). Additional buffering can also be added between pipeline stages to allow more in-flight instructions. However, their presence also requires additional logic being added, e.g. to handle data dependencies between instructions within the stages and in the buffers between them. Finally, branch instructions may require the content of the Program Counter (PC) register being changed, hence requiring instructions in the pipeline to be flushed and new ones being then loaded (GONZALEZ; LATORRE; MAGKLIS, 2010; PATTERSON; HENNESSY, 2011; HENNESSY; PATTERSON, 2012).

Figure 32 depicts the basic structure of a simple five-stage instruction processing pipeline, which splits instructions' execution in five stages: Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Memory Access (MA) and Write-Back (WB). The IF stage retrieves the next instruction to be executed, which is pointed out by the PC register of the Register File (RF), from the Instruction Memory (IM). The ID stage parses the instruction's machine code and transforms it into signals semantically meaningful for the next stages. The EX stage performs instructions' logic and arithmetic operations, e.g. using an ALU or a Floating-Point Unit (FPU). The MA stage performs read/write accesses to the Data Memory (DM). Finally, the WB stage writes the instructions' results into the work registers located in the RF. The arrows indicate information flow between the units (PATTERSON; HENNESSY, 2011; HENNESSY; PATTERSON, 2012).

Figure 32: Simple pipeline design



We highlight that modern processors generally employ very complex pipelines which use, for instance, out-of-order execution techniques, i.e. process instructions in an order that can differ from that of the program binary, or superscalar designs, i.e. stages capable of issuing more than one instruction per clock cycle. Such techniques enable higher instruction-level parallelism (and average performance) being achieved, but require for that highly complex logic (GONZALEZ; LATORRE; MAGKLIS, 2010; PATTERSON; HENNESSY, 2011; SHEN; LIPASTI, 2013). Such complexity at the hardware level also leads to big challenges in employing modern processors in critical RTSs, since static approaches generally lack composability characteristics that are necessary for their timing analysis (WILHELM et al., 2008; NOWOTSCH; PAULITSCH, 2012). In this context, MBPTA emerges as a promising approach for handling the hardware complexity issue in estimating WCET bounds for RTSs' tasks (KOSMIDIS et al., 2016).

6.2 MULTITHREADING

In the computing context, a *thread (of execution)* is associated to a set of registers that hold the execution state of a program (software) within a processor. Simple processors generally use cores that support the processing of a single thread of execution. On the other hand, multithread cores are capable of simultaneously holding the execution state of several threads at hardware level, and executing their instructions in an overlapped manner while sharing most of the processing elements between them. A multithread pipeline is hence capable of exploiting, at some level, both instruction-level parallelism – by overlapping instructions' execution – and thread-level parallelism – by processing instructions from different threads simultaneously. For handling multiple threads, a processing core requires not only a separate register file for each thread, but it must also include logic for scheduling and controlling threads' execution (GONZALEZ; LATORRE; MAGKLIS, 2010; PATTERSON; HENNESSY, 2011; HENNESSY; PATTERSON, 2012).

A serious drawback of multithreading, which is particularly harmful to its applicability in processors that target RTSs, is the intrinsic existence of hardware-level timing interference between threads. Such interference occurs because, in these pipelines, many hardware processing elements (i.e. functional units) are shared between threads and must be accessed exclusively. Moreover, the delay induced between threads depends on the specific operations performed by each of them, since functional units' latency may vary depending on the performed operations and on the operated data. This causes the use of multithread processing cores to be often discouraged in the context of RTSs, since the determination of WCET estimates for tasks executing on them through static methods often proves very challenging and/or leads to extremely pessimistic upper-bounds (WILHELM et al., 2008; NOWOTSCH; PAULITSCH, 2012; KOTABA et al., 2013).

A critical aspect of multithreading is the thread scheduling policy employed, i.e., the decision on when to place instructions of each active thread into the pipeline for being processed. Thread scheduling is typically either (1) fine-grained, when switches occur at every clock cycle, (2) coarse-grained, when switches only occur when costly operations take place, or (3) simultaneous, when a dynamically scheduled superscalar pipeline executes in parallel several instructions from different threads (HENNESSY; PATTERSON, 2012). The employed scheduling policy may also play an important role with respect to the limitation of timing interference between threads, since it can be tailored to grant some level of fairness regarding the amount of processing resources effectively provided for each of them (MARKOVIC, 2015).

The pipelining and multithreading techniques are well-established, and related material is widely available, in the field of high-performance computing (PATTERSON; HENNESSY, 2011; HENNESSY; PATTERSON, 2012). Several previous works were developed that aimed enabling static timing analysis for tasks executed on processors equipped with multithread pipelines. A multithread pipeline was proposed in (ZIMMER et al., 2014) which interleaves threads' instructions such that a new instruction of a same thread only enters the pipeline when the last one already left it, thus eliminating both hazards and the need for forwarding logic. As a consequence, static analysis of tasks executed on it becomes much easier in comparison with designs that employ other scheduling approaches. Similarly, (UNGERER et al., 2010) presents a pipeline design in which one single real-time thread executes together but with high priority over several non-real-time others. The approach enables performance guarantees for the real-time thread being easily provided, since it executes exactly as if it was the only existing thread and interference is only observable over the non-real-time ones. To the best of our knowledge, no previous work evaluated the utilization of randomized scheduling in the context of multithread pipelines for benefiting the application of probabilistic timing analysis techniques.

6.3 MULTITHREAD PIPELINE DESIGN

A high-level graphical representation of the multithread pipeline we employ in this work is presented in Figure 33. Similarly to the example presented in Section 6.1, it decomposes instructions' execution into five stages: IF, ID, EX, MA and WB. It implements the 32-bit MIPS I RISC instruction set (PRICE, 1995) added with a thread halting instruction (HALT), and can be classified as a scalar, in-order, single-issue, statically-allocated multithread instruction pipeline (PATTERSON; HENNESSY, 2011). Moreover, it employs an additional instruction buffer between each pair of stages, enabling a single instruction being stored before proceeding to the next stage. Its RFs are composed of 35 registers each and have two read/write ports, for supporting MIPS instructions that read or write two registers in parallel.

A scheduler (SCH) feeds the pipeline's first stage with the identifier of the thread whose next instruction is to be processed, which is determined based on the policies presented in Sections 6.6 and 6.7. The thread identifier enters the pipeline and proceeds, together with the other information of the instruction to be processed, throughout the stages. Once an instruction is scheduled, it only leaves the pipeline if a previous branch or halt instruction of the same thread causes it to be flushed. Flushes only affect in-flight instructions belonging to the thread that performed the branch or halt, and no branch prediction mechanism is used (i.e. instruction fetching always proceeds in memory order). Only an ALU is currently available for use in the EX stage, and consequently only integer operations are supported. The RAM memories containing instructions (IM) and data (DM) are connected to the IF and MA stages, respectively. The RFs are selectively connected to the IF, EX and WB stages, according to the identifier of the thread being currently processed on each of them.

The PRNG we employ for time-randomization purposes at hardware level (A) has a single clock cycle latency and works by XORing the results of two other different generators running on independent and non-synchronized oscillators (TKACIK, 2003), and (B) is periodically seeded using a TRNG based on a ring of ring oscillators (BAETONIU, 2004) for ensuring high-quality randomness is achieved. This approach is similar to that used by Intel® processors' RDRAND instruction, which accesses a high-throughput PRNG that is periodically seeded through a TRNG based on a thermal noise entropy source (INTEL, 2012). For more details on implementation aspects of other hardware elements developed during the research described in this thesis, please refer to Appendix C.

We also use a thread dispatcher (TD) which is capable of automatically triggering periodic, aperiodic and sporadic tasks executed on the pipeline's threads, by monitoring and controlling their respective reset and halt signals. It is assumed that (A) a single task is executed on every thread, and (B) a task signals the end of its execution through a HALT instruction. Tasks' dispatching is configured by defining the following parameters for each of the threads:

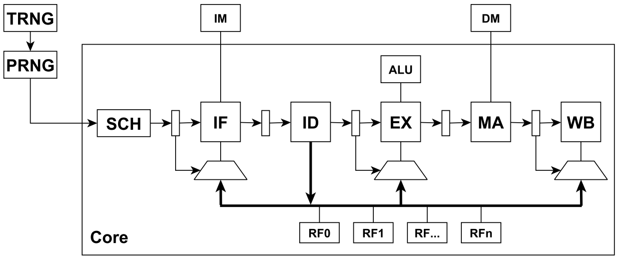
Permanent (logical) Determines whether tasks should be continuously released, and is used for inducing maximum interference during the measurement collection process (see Section 6.4).

Period (numeric) Defines the period between individual releases of the tasks, where zero indicates aperiodicity.

Interval (numeric) Establishes a minimum interval to be respected between consecutive releases of tasks configured as aperiodic, hence causing them to behave as sporadic tasks.

Jitter width Determines the width (in bits) of a random uniform release jitter that can be added to the tasks' dispatching, and is mainly used for behaviour testing purposes.

Figure 33: Pipeline design



This design differs from (ZIMMER et al., 2014) by (1) being subject to hazards and hence requiring forwarding and interlocking logic, since random scheduling can lead the pipeline to be loaded with sequences of instructions of a same thread, and (2) supporting less than $s-1$ threads in execution (where s is the number of pipeline stages), since it does not need to schedule a thread only when its previous instruction has already left the pipeline. It also differs from the pipeline proposed in (UNGERER et al., 2010) by not requiring real-time threads to be executed with higher priority in relation to non-real-time ones, hence potentially enabling the provision of probabilistic real-time guarantees for all threads.

6.4 MAXIMUM INTERFERENCE SCENARIO

A critical step in collecting samples for performing MBPTA is guaranteeing that the execution times observed during the analysis either exactly match or upper bound those produced in operation (CAZORLA et al., 2016). One possible approach to perform this step when multithread pipelines are to be employed involves establishing a *maximum interference scenario.* In such a scenario, the thread under analysis should experience temporal interference from other threads that can be considered – and consistently evidenced – to upper-bound the interference observed in normal operation.

To enable the establishment of a maximum interference scenario for the multithread pipeline used in this work, hardware elements coupled to it that influence its timing behaviour must support both a *measurement mode* and an *interference mode.* In measurement mode, data-dependent latencies must be forced to their highest possible value, such that normal operation ones are upper-bounded regardless of the operated data (KOSMIDIS et al., 2016). In interference mode, timing behaviour that influence inter-thread interference must be forced to the highest possible, such that interference over other threads is effectively maximized. Within the multithread pipeline design used in this work, both modes exclusively cause the ALU to always produce its maximum possible latencies. This is because (A) a deterministic memory hierarchy is used in our evaluations, and (B) the ALU latencies are in fact the main source of inter-thread interference in the employed pipeline.

Coupled with such hardware elements, the pipeline we use allows individual threads being forced into *interference mode*, which causes them to (A) repeatedly execute ghost instructions, i.e. instructions that have no effect except presenting temporal behaviour compatible with that of real ones, and (B) set the memory hierarchy and the ALU into *interference mode*, for maximizing inter-thread interference timing effects. With respect to (A), the MIPS I instruction set is composed of instructions that, according to the potential interference they can induce on other threads' timing due to their latency in traversing each stage of the pipeline, can be classified as follows:

F Interfere only in the IF stage, for taking a single clock cycle to traverse all other stages of the pipeline, and therefore includes the NOOP, SLTI(U), LUI, MOV(Z/N), M(F/T)HI, M(F/T)LO, SLT(U) and HALT instructions.

FF Interfere only in the IF stage, but cause the owner thread's in-flight instructions to be flushed out of the pipeline when the WB stage is reached, comprising the branching instructions J(AL), J(AL)R, B(LT/GE)Z(AL), B(EQ/NE) and B(LE/GT)Z.

FM Interfere in the IF and MA stages, comprising the memory instructions LB(U), LH(U), LW, SB, SH and SW.

FEAL Interfere in the IF and EX stages and use the ALU to perform a low-latency (typically logic) operation, containing the ADDI(U), ANDI, ORI, XORI, SLL, SRL, SRA, SLLV, SRLV, SRAV, ADD(U), SUB(U), AND, OR, XOR and NOR instructions.

FEAH Interfere in the IF and EX stages and use the ALU to perform a high-latency (typically arithmetic) operation, comprising the MULT(U) and DIV(U) instructions. For containing the instructions of highest single-stage latencies, it is expected to induce the highest potential inter-thread interference.

A maximum-interference measurement scenario can hence be built by (1) setting the thread to be analysed into *measurement mode*, and (2) setting all other threads into *interference mode* permanently executing ghost instructions of the FEAH interference class. This scenario is expected to cause the analysed thread to experience the maximum possible interference from others, hence providing a suitable condition for collecting upper-bounding measurements to perform MBPTA.

6.5 EVALUATION METHOD

Time-randomized multithread pipelines are expected to present certain key properties for being considered usable for probabilistic timing analysis through MBPTA. The *maxima analysability* property is associated with the production of execution times that (A) can be deemed independent and identically distributed, and (B) present maxima that can be evidenced to adhere to one of the extreme value distributions employed by EVT. The *timing dominance* property is associated with the existence, knowledge and possibility of consistently evidencing, the measurement conditions that yield the worst-case timing behaviour of a certain thread with respect to interference from others. A non-fundamental but rather important property is *interference balancing*, which means that tasks' execution times in the maximum interference scenario should not increase, in comparison with those of solo execution, by a factor that largely exceeds the number of threads t. Interference balancing can also be seen as the provision of approximately $\frac{1}{t}$ of the processor's capacity for each of the t threads in the maximum interference scenario. An additional assessment worth being performed regards the *typical scenario slowdown*, in order to evaluate how severe are the effects of inter-thread interference tasks are likely to be subject under typical conditions (i.e. when real tasks are executed on the interfering thread). Threads' slowdown factor in the typical execution scenario is also expected to approximate t in relation to solo execution, since large slowdowns are only expected under conditions that are extremely unlikely to occur in typical situations. This is so because, during normal execution, potential interference is reduced by (1) functional units not yielding worst-case latencies, and (2) interfering tasks not performing only high-latency operations. In this work, we evaluate the aforementioned properties as follows:

Maxima analysability is evaluated by submitting the execution times produced in the maximum interference scenario to the typical MBPTA applicability assessments presented in Section 2.5, i.e. by applying statistical tests and creating diagnostic plots for evaluating (1) independence and identical distribution and (2) maxima adherence to extreme value models. In this work we evaluate maxima analysability considering EVT application through both approaches, i.e., BM – with blocks of size 50 as performed in Chapter 5 – and POT – defining thresholds through the minimization of the GP distribution estimated quantiles' mean absolute error as also performed in Chapter 5.

Timing dominance is evaluated from two distinct perspectives:

- *Class dominance* aims evidencing that the execution times produced in the maximum interference scenario dominate (i.e. are consistently higher than) those produced under interference of instructions that belong to other interference classes. This is necessary because the introduction of interference regulation mechanisms, e.g. based on thread suspension, may change instructions' timing dominance characteristics. For instance, by over-suspending highly-interfering tasks the low-interference ones can be scheduled frequently enough to induce larger slowdowns than high-interference tasks. Class dominance is evidenced by the maximum interference scenario execution time distributions being composed of values consistently higher than those observed on samples obtained under interference of other instruction classes (see Figures 38 and 46).

- *Behavioural dominance* evidences that the measurements taken in the maximum interference scenario can be deemed to dominate those produced under different execution conditions. For that we measure execution times of several benchmarks in numerous environments, composed by applying a large number of distinct combinations of behavioural configurations to the interfering thread. The ≈ 25000 evaluated scenarios are built by using (A) measurement mode active/inactive, (B) execution of distinct randomly-generated mixes of ghost instructions with different class proportion combinations, (C) intermittent behaviour, given by periodic and sporadic releases using different periods and intervals, and (D) release jitter varying from zero to several thousand clock cycles. The evaluation hence evidences timing dominance of the maximum interference scenario independently of both the specific instructions that compose interfering tasks and of the interfering tasks' configuration and behaviour. Behavioural dominance is evidenced by the maximum-interference execution times' distributions being composed of values consistently higher than those observed on all distinct execution conditions considered (see Figures 39 and 47).

Interference balancing is evaluated by comparing the distribution of the execution time measurements obtained while a set of benchmarks are executed solo, against those of their execution in the maximum interference scenario. Interference balancing is evidenced by the increase factor between the two compared conditions not largely exceeding the number of threads t.

Typical scenario slowdown is evaluated by comparing the execution times observed for a set of benchmark tasks while other real tasks are repeatedly executed on the interfering thread (we use the same benchmarks employed in the evaluations) with those obtained in solo execution. The slowdown factor between solo and typical execution is expected to approximate the number of threads t.

The experimental processor we used to perform the proposed evaluations is a single-core, equipped with a dual-thread instance of the multithread pipeline presented in Section 6.3, running at 50MHz on an FPGA. It uses separate instruction and data RAM memories, which are directly accessed by the processor. The benchmark tasks used in the evaluations were *bsort*, *bs*, *cnt*, *cover*, *crc*, *expint*, *fdct*, *fibcall*, *fir*, *insertsort*, *janne_complex*, *matmult*, *ns* and *prime* from the Mälardalen WCET Benchmarks suite (GUSTAFSSON et al., 2010), which are often used in comparisons and assessments of WCET-related tools. Since these are multi-path tasks, we fix their inputs such that only a long execution path, i.e. that performs a large number of elementary operations, is exercised (e.g. we use reverse-sorted integer arrays for the sorting algorithms). As mentioned and in line with the suggestions of (GRIFFIN; BURNS, 2010; CUCU-GROSJEAN et al., 2012; KOSMIDIS et al., 2016), we also reset the state of the hardware platform before executions, for avoiding dependency between measurements due to state-related effects. Consequently, the observed timing variability stems exclusively from the time-randomized multithread pipeline, since we isolate effects from any other sources. We highlight, however, that (A) these conditions are not sufficient for guaranteeing that the real WCETs are effectively observable, and (B) the determination of tasks' input data for the reliable application of MBPTA must be performed in a per-scenario basis and is still considered an open problem (GIL et al., 2017).

The herein presented evaluations were also replicated using a cached memory hierarchy, which increases instruction and data memories' access throughput by fetching consecutive words in a pipelined fashion. A time-randomized two-way set-associative thread-partitioned cache memory was used (KOSMIDIS et al., 2013a), which implements the modulo placement policy, the write-through update policy, a randomized line replacement policy, and a simple write-to-all-partitions coherency mechanism. Partitioning is used for avoiding inter-thread timing interference due to cache effects, whose worst-case behaviour prediction and/or reproduction is hard (ALTMEYER et al., 2014), but the approach presented in (SLIJEPCEVIC et al., 2014) could also be used for handling cache sharing between threads. In interference mode the employed cache memories assume reads always miss, in order to maximize interference induced on other threads while instructions traverse the IF and MA stages. The conclusions achieved in this replication were consistent with those drawn without cache memories.

6.6 PURELY RANDOM SCHEDULER (PRS)

The first randomized thread scheduling approach we evaluate is the Purely Random Scheduler (PRS), which selects the next active thread to be scheduled based solely on (pseudo-)random numbers that follow an approximately uniform distribution. In other words, PRS randomly selects one between the active threads such that all have approximately the same probability of being chosen, and schedules it whenever a new instruction can be inserted into the pipeline.

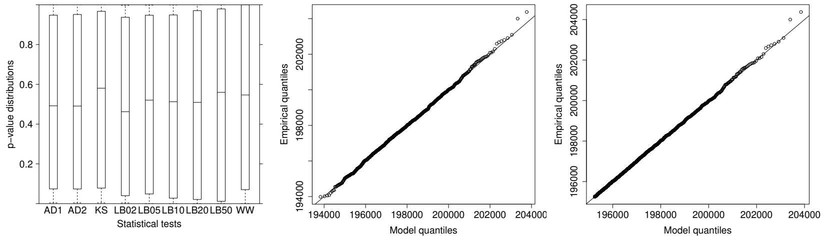
6.6.1 Evaluation

This section presents the results obtained by applying the evaluation method proposed in Section 6.5 on the execution times of benchmark tasks executed on the multithread pipeline using a PRS.

6.6.1.1 Maxima Analysability

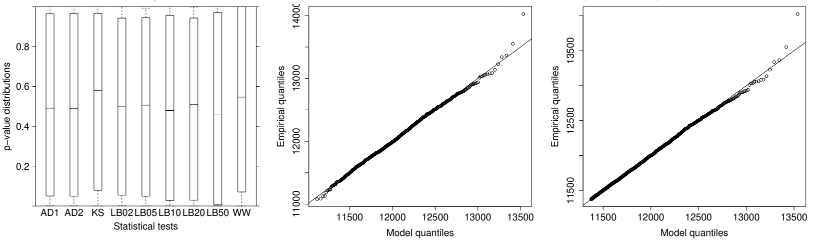
Figures 34 to 37 show the MBPTA applicability evaluation artefacts presented in Section 2.5, for samples of 50000 measurements of the benchmarks' execution times. These plots' analysis evidence that the maxima analysability property can be considered to hold when the PRS is used, since in the maximum interference scenario all MBPTA applicability requirements can be deemed being acceptably met. Equivalent analysability evaluation plots associated with the remaining benchmarks can be found in Appendix B.

Figure 34: *bsort* PRS maxima analysability analysis



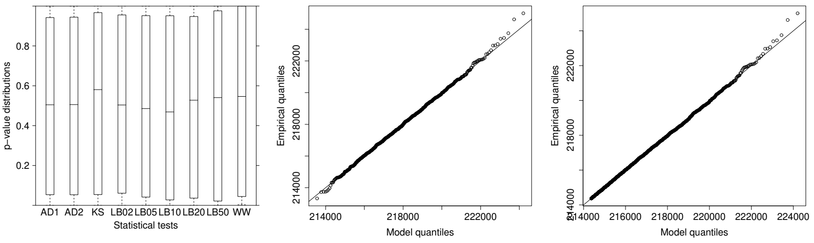
(a) I.i.d. statistical tests (b) GEV quantile plot (c) GP quantile plot

Figure 35: *bs* PRS maxima analysability analysis



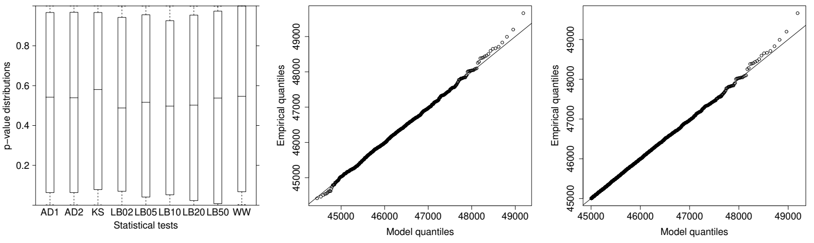
(a) I.i.d. statistical tests (b) GEV quantile plot (c) GP quantile plot

Figure 36: *cnt* PRS maxima analysability analysis



(a) I.i.d. statistical tests (b) GEV quantile plot (c) GP quantile plot

Figure 37: *cover* PRS maxima analysability analysis

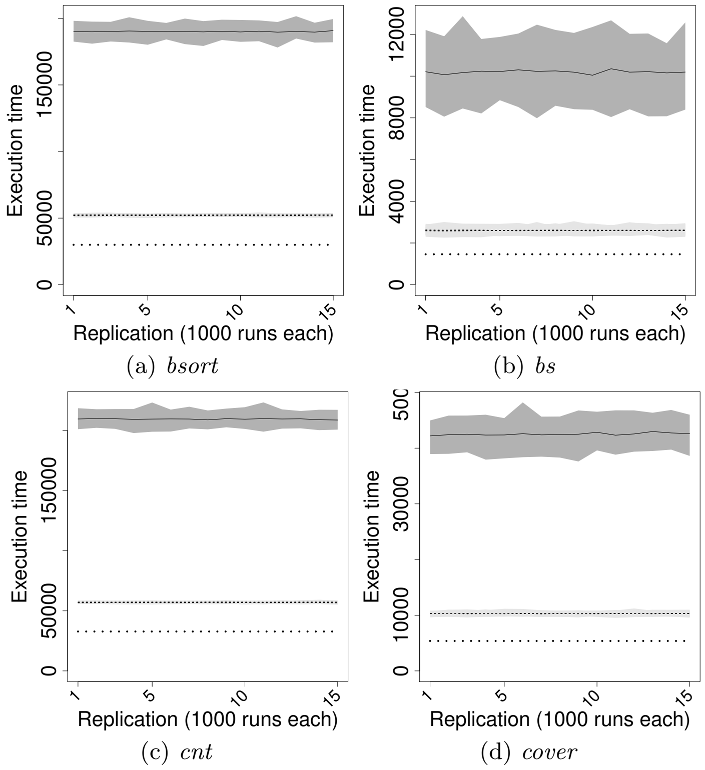


(a) I.i.d. statistical tests (b) GEV quantile plot (c) GP quantile plot

6.6.1.2 Class Timing Dominance

The plots presented in Figure 38 show the mean, the minimum and the maximum execution times observed while the benchmark tasks execute (1) in the maximum interference scenario (i.e. under FEAH instructions' interference), as a continuous line in a dark grey region, and (2) under interference of instructions of classes other than FEAH, as superposed dashed lines on light grey regions. The benchmark tasks' solo execution times are highlighted as a straight dotted line, for slowdown evaluation purposes. Each of the employed execution time samples is composed of 1000 measurements and was replicated 15 times. The plots evidence that the class timing dominance property holds when the PRS is used, since benchmarks' execution times produced in the maximum interference scenario consistently dominate those yielded under interference of instructions of classes other than FEAH. Class timing dominance plots associated with the other benchmark tasks considered in this work can be found in Appendix B.

Figure 38: PRS class timing dominance analysis



(a) *bsort* (b) *bs* (c) *cnt* (d) *cover*

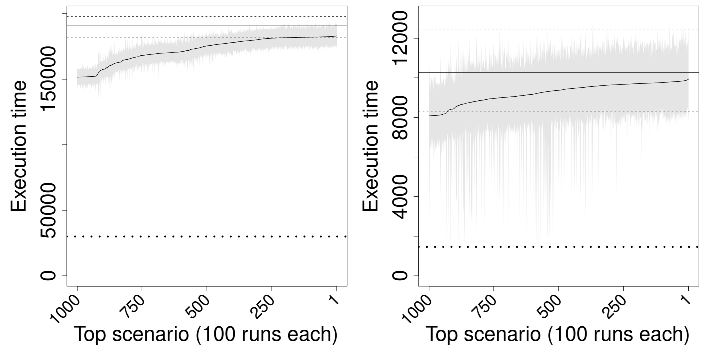
6.6.1.3 Behavioural Timing Dominance

The plots shown in Figure 39 present the mean, the minimum and the maximum execution times observed while the benchmark tasks are executed (1) in a large number of distinct interference conditions (see Section 6.5 for details), as a continuous line in a light grey region, and (2) in the maximum interference scenario, as a straight continuous line surrounded by dashed lines. Approximately 25000 scenarios were evaluated using samples composed of 100 execution times each, and therefore only those that produce the top-1000 highest mean execution times are presented for improving plots' readability. Benchmark tasks' solo execution time is highlighted as a straight dotted line, for slowdown evaluation purposes. The plots evidence that the behavioural timing dominance property holds when the PRS is used, since benchmarks' execution time distributions produced in the maximum interference scenario consistently dominate those observed in numerous distinct execution conditions. Behavioural timing dominance plots associated with the other benchmarks can be found in Appendix B.

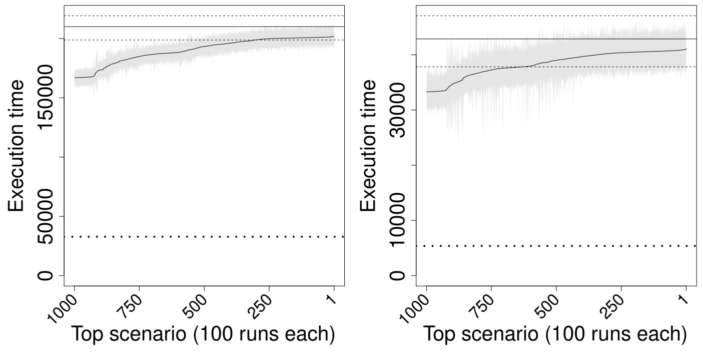
6.6.1.4 Interference Balancing

Table 2 presents, for each of the considered benchmark tasks, its solo execution time and the slowdown factor observed for its average execution times in the maximum interference scenario in relation to solo execution. This information is presented for both the processor executing in measurement mode, i.e. with logical units producing maximum latencies, and under normal execution conditions. Based on its analysis, it is evident that the interference balancing property does not acceptably hold when the PRS is employed. This is so because the slowdown factors experienced by the measured thread in the maximum interference scenario largely exceed the total number of threads. Consider, for instance, the execution times yielded for the *bsort* benchmark, whose solo execution takes $\approx$ 30000 clock cycles to complete. Taking into account that a multithread design with t threads enables t tasks being executed on the same core, the ideal maximum slowdown factor for each of the threads also equals t. For this reason, if the interference balancing property held ideally in the evaluated dual-thread pipeline, then the *bsort* benchmark would take $\approx$ 60000 clock cycles to execute in the maximum interference scenario. However, its mean execution time reaches values as high as $\approx$ 190000 cycles. Such slowdown is observable

Figure 39: PRS behavioural timing dominance analysis



(a) *bsort* (b) *bs*



(c) *cnt* (d) *cover*

because, despite scheduling is uniform among threads, some may execute low-interference instructions (e.g. NOOPs) while others execute highly-interfering ones (e.g. MULTs). This fact causes the processing provision to become unbalanced, allowing threads' execution times to be largely increased due to uncontrolled inter-thread interference.

6.6.1.5 Typical Scenario Slowdown

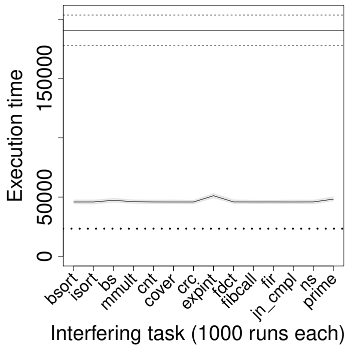
The plots shown in Figure 40, whose reading is similar to that of the behavioural timing dominance plots of Section 6.6.1.3, compare the distributions of 1000 execution times obtained while each benchmark executes under interference of real tasks (the same set of benchmarks) against those of solo execution and of the maximum interference scenario. Their analysis shows that the slowdown factor threads experience under

Task	Measurement mode		Execution mode	
	Solo execution	Maximum slowdown	Solo execution	Maximum slowdown
bsort	29967	6.35	23352	8.15
isort	25946	6.45	20762	8.06
bs	1460	7.00	1310	7.80
mmult	54292	5.68	36692	8.40
cnt	32802	6.40	25602	8.20
cover	5377	7.93	5377	7.93
crc	10092	7.31	9372	7.87
expint	36896	6.51	31588	7.61
fdct	56074	5.99	41540	8.09
fibcall	4457	7.87	4457	7.87
fir	65640	7.21	60614	7.81
jn_cmpl	4916	7.74	4812	7.91
ns	29244	6.39	22437	8.33
prime	16017	6.71	14392	7.47

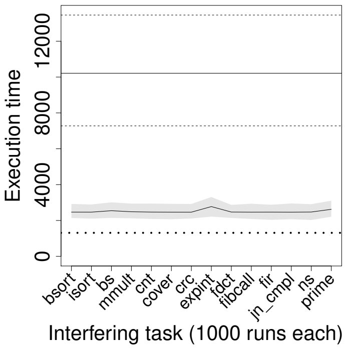
Table 2: PRS maximum slowdown

typical execution conditions while using the PRS is acceptable, i.e. approximates the ideal value t, despite it proves higher in the maximum interference scenario. Consider for instance the *bsort* benchmark task (see Figure 40(a)), which takes ≈ 30000 clock cycles to execute solo. Its average execution time in typical execution conditions remain close to ≈ 51000 cycles, and its typical slowdown factor is hence potentially lower than t. Similar plots, associated with other benchmark tasks, can be found in Appendix B. Summarized information on typical scenario slowdowns is presented in Table 3, which shows, for each benchmark task, its solo execution time and the maximum slowdown factor it was observed to experience in the evaluated typical scenarios – considering their execution both in measurement mode and in normal mode.

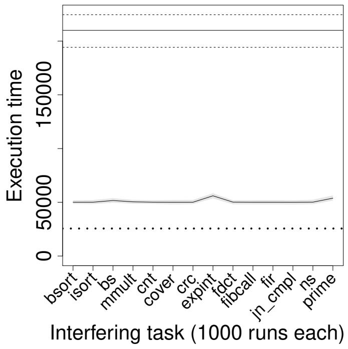
Figure 40: PRS typical scenario slowdown analysis



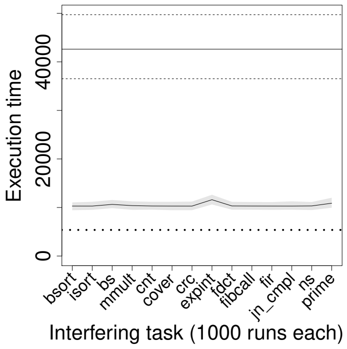
(a) *bsort*



(b) *bs*



(c) *cnt*



(d) *cover*

Task	Measurement mode		Execution mode	
	Solo execution	Typical slowdown	Solo execution	Typical slowdown
bsort	29967	1.71	23352	2.19
isort	25946	1.76	20762	2.20
bs	1460	1.90	1310	2.12
mmult	54292	1.49	36692	2.20
cnt	32802	1.71	25602	2.19
cover	5377	2.16	5377	2.16
crc	10092	2.03	9372	2.18
expint	36896	1.81	31588	2.11
fdct	56074	1.65	41540	2.23
fibcall	4457	2.17	4457	2.17
fir	65640	2.00	60614	2.16
jn_cmpl	4916	2.11	4812	2.16
ns	29244	1.67	22437	2.17
prime	16017	1.84	14392	2.05

Table 3: PRS typical slowdown

6.7 INTERFERENCE-REGULATED SCHEDULER (IRS)

For tackling the interference balancing issue observed for the PRS in Section 6.6.1.4, we propose and evaluate in this section a novel randomized thread scheduler which is expected to (1) limit the timing effects of maximum inter-thread interference, while ideally (2) conserving typical scenario slowdowns observed using the PRS. The proposed thread scheduler, referred to as Interference-Regulated Scheduler (IRS), is based on an inter-thread interference detection logic coupled to a credit-based schedulability regulator. The interference detection logic it uses identifies in-flight instructions' states, i.e. pipeline stages' and inter-stage buffers' states, that characterize inter-thread interference. A credit-based eligibility regulator then uses this detection logic to account for interference as execution evolves, and temporarily suspends the (randomized) selection of high-interfering threads for balancing the provision of processing resources. The following sections present relevant concepts and details of the IRS implementation.

6.7.1 Thread States

We consider that (A) a *ready thread* is a thread which was released and is therefore eligible to be scheduled into the pipeline, (B) a *suspended thread* is a ready thread that was suspended by the scheduler for regulation purposes, (C) a *halted thread* is one that executed a HALT instruction and is therefore temporally or permanently reset due to task completion, and (D) a *stopped thread* is one that is either not yet released or has been halted, and is therefore not eligible for scheduling.

6.7.2 Inter-Thread Interference Detection

The interference detection logic used by the IRS is formally defined according to the following set of statements:

- A stage s_i is considered to be *blocked* when its processing is finished, its output buffer is full, and the following stage s_{i+1} is currently either in busy or in blocked state.
- A thread t_i is considered to be *blocked on a stage* s_i when s_i is blocked while executing an instruction of t_i.
- A thread t_i is said to be *blocking a stage* s_i when one of its instructions is causing s_i to be blocked, either directly or indirectly. Indirect blocking refers to stages whose processing is blocked because a contiguous chain of posterior stages are blocked.

- Finally, a thread t_2 is considered to be *interfering with* a thread t_1 when an instruction of t_2 is being executed on a stage s_2 while an instruction of t_1 is executing on a stage s_1 previous to s_2 which is currently being blocked by t_2, either directly or indirectly.

We highlight the existence of a recurrence characteristic in the proposed interference detection logic, since we consider interference occurs when a thread's instruction is avoiding another thread's instruction to proceed in the pipeline due to direct or indirect stage blocking. Let, for instance, s_1 and s_2 be two consecutive pipeline stages executing instructions of threads t_1 and t_2, respectively. We consider s_1 is blocked if (1) its processing is completed, (2) its output buffer is full, and (3) s_2 is busy or blocked by a posterior stage. If s_2 is currently busy then t_1 is considered to be under interference of t_2, but if s_2 is currently blocked then t_1 is considered to be under interference of the thread that is, either directly or indirectly, blocking thread t_2.

6.7.3 Credit-Based Schedulability Regulator

Credit-based regulators are often used to achieve fairness in sharing hardware elements among multiple clients (e.g. buses). Such regulators periodically allocate credits for each client, and account for their consumption as utilization evolves for deciding on granting or delaying requests (AKESSON; STEFFENS; GOOSSENS, 2009; SLIJEPCEVIC et al., 2017b). The credit-based regulator used in our scheduler consists of a set of t credit accounting and schedulability decisioning mechanisms running in parallel, one for each of the possible number of active threads (excluding the zero case), where t is the total number of threads. It is characterized, for each number of active threads $a \in [1..t]$, by credit periods $Cprd_a$, initial credit values $Cini_a$, minimum and maximum credit values $Cmin_a$ and $Cmax_a$, and budget credits $Cbdg_a$.

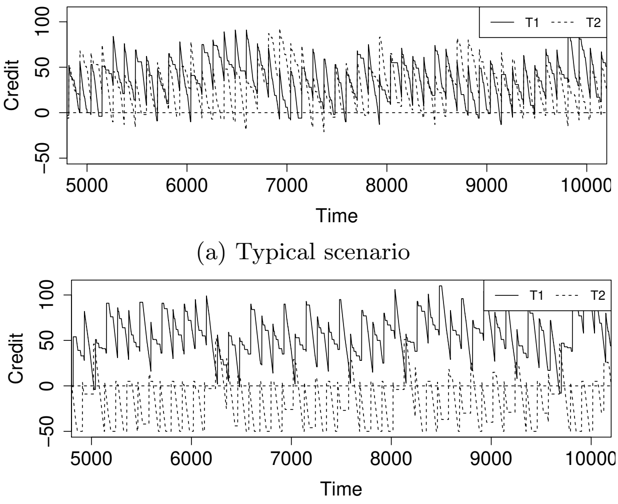
Each thread $z \in [1..t]$ is associated to a set of t credit accumulator registers $Cacc_{a,z}$, which are (A) reinitialized to $Cini_a$ when thread z is stopped, (B) permanently enforced to assume values such that $Cmin_a \leq Cacc_{a,z} \leq Cmax_a$, (C) decremented on every clock cycle in which thread z is not stopped and is interfering with any thread, and (D) increased by $Cbdg_a$ every time the $Cprd_a$ regulation period elapses while the associated thread is not stopped. Whenever $Cacc_{a,z} \leq 0$, thread z is suspended if and only if a threads are currently active. Among the eligible threads, scheduling is performed based on random numbers with approximately uniform distribution – i.e. all have the same approximate chance of being scheduled next into the pipeline.

Each parameter of the IRS affects its functioning in a distinct manner. $Cbdg_a$ and $Cprd_a$ affect the speed and frequency with which threads' credits are restored, respectively. $Cini_a$ defines the threads' credit at the moment they are released. $Cmin_a$ influences the credit deficits threads can be subject due to pre-consumption (which occur because suspensions do not flush in-flight instructions), after their budget is restored – where zero indicates the threads obtain their full budgets on restores and negative values allow their budgets being pre-consumed on suspensions. Finally, $Cmax_a$ affects the highest credit a thread can accumulate when low consumption is observed.

Most of the IRS's parameters should be either fixed or derived in function of others' values, for granting threads' timing certain fundamental properties. $Cini_a$ should be set to the same value as $Cmin_a$, for ensuring threads cannot receive budgets higher than others' by frequently restarting execution. $Cmax_a$ should be set to $Cprd_a$, for ensuring low-interference threads can accumulate credit for spending at most an entire credit period interfering on others. Finally, $Cbdg_a$ should be set as $Cbdg_a = \frac{Cprd_a}{a}$ to ensure that, when a threads are active, any individual thread can receive budget for spending at most $\frac{1}{a}$ of the credit period inducing interference. Moreover, when a single thread is active (i.e. when $a = 1$) it should be continuously scheduled into the pipeline, which can be done by assigning $Cmin_1 = Cprd_1 = 1$. Consequently, for the dual-thread case (i.e. with $t = 2$) only parameters $Cmin_2$ and $Cprd_2$ values must be defined.

We show in Figure 41 traces of the IRS credit consumption and restoration patterns, for threads T1 and T2, observed during 5000-cycle slices of simulated executions on the multithread pipeline. The software task used is *bsort* and scheduler's parameters are set as $Cmin_2 = -50$ and $Cprd_2 = 110$, hence (1) every $Cprd_2 = 110$ clock cycles each thread receives $Cbdg_2 = 55$ credit units, (2) threads can accumulate up to $Cmax_2 = 110$ credit units, and (3) threads can pre-consume up to $-Cmin_2 = 50$ credit units while suspended (since in-flight instructions are not flushed on suspensions). In the trace shown in Figure 41(a) the *bsort* task is executed on both threads, which can be regarded as a typical execution scenario since real tasks are being executed on both threads. The trace shows that, during most of the analysed period, both threads remain schedulable (have credit greater than zero). This indicates that, in this case, random thread selection is sufficient for balancing the allocation of processing among threads. In the trace shown in Figure 41(b) the *bsort* task is executed on T1 in the maximum interference scenario, i.e. with T2 permanently executing ghost instructions of the FEAH interference class. The trace shows that T1 remains almost permanently schedulable, since it does not induce strong interference on T2's execution. On the other hand, T2 quickly consumes its credit as soon as it is restored, remaining non-eligible for scheduling during most of the analysed period, hence limiting the interference T2 is capable of inducing on T1's execution.

Figure 41: IRS credit consumption traces



(a) Typical scenario

(b) Maximum interference scenario

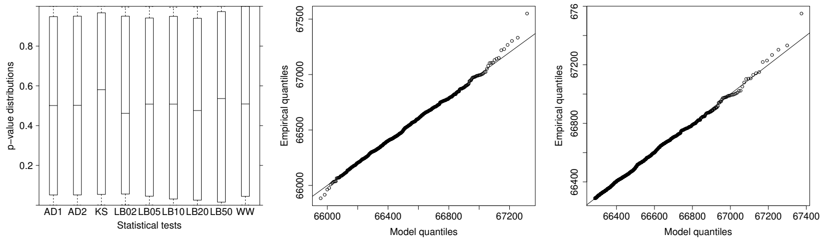
6.7.4 Evaluation

This section shows the results obtained by applying the evaluation method proposed in Section 6.5 on the execution times of benchmark tasks executed on the multithread pipeline with the IRS, together with comparisons with those obtained using the PRS. The measurements for this section's evaluations were taken using parameters $Cmin_2 = -50$ and $Cprd_2 = 110$. These values were defined based on an empirical assessment that mainly envisioned the production of MBPTA-analysable execution time distributions. Optimized parameter calibration and detailed analyses of the sensitivity of execution times' analysability to IRS parameters are subjects for future works.

6.7.4.1 Maxima Analysability

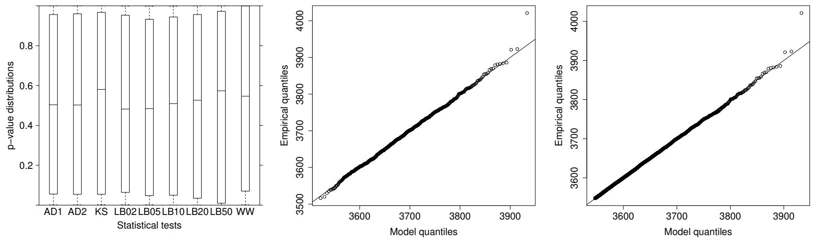
Figures 42 to 45 show the MBPTA applicability evaluation artefacts presented in Section 2.5, for samples of 50000 measurements of the benchmarks' execution times. These plots' analysis evidence that the maxima analysability property can be considered to hold when the IRS is used, since in the maximum interference scenario all MBPTA applicability requirements can be deemed being acceptably met. Equivalent analysability evaluation plots associated with other benchmark tasks can be found in Appendix B.

Figure 42: *bsort* IRS maxima analysability analysis



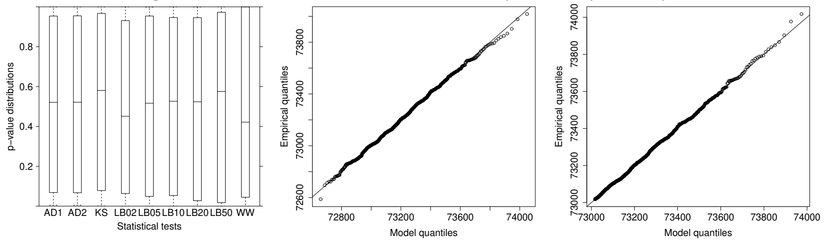
(a) I.i.d. statistical tests (b) GEV quantile plot (c) GP quantile plot

Figure 43: *bs* IRS maxima analysability analysis



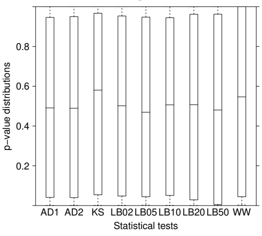
(a) I.i.d. statistical tests (b) GEV quantile plot (c) GP quantile plot

Figure 44: *cnt* IRS maxima analysability analysis

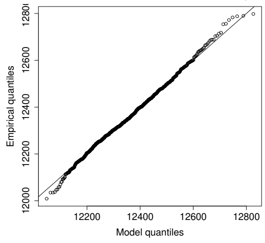


(a) I.i.d. statistical tests (b) GEV quantile plot (c) GP quantile plot

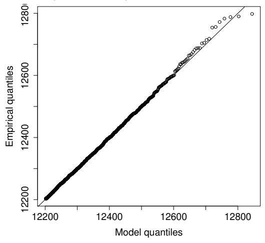
Figure 45: *cover* IRS maxima analysability analysis



(a) I.i.d. statistical tests



(b) GEV quantile plot



(c) GP quantile plot

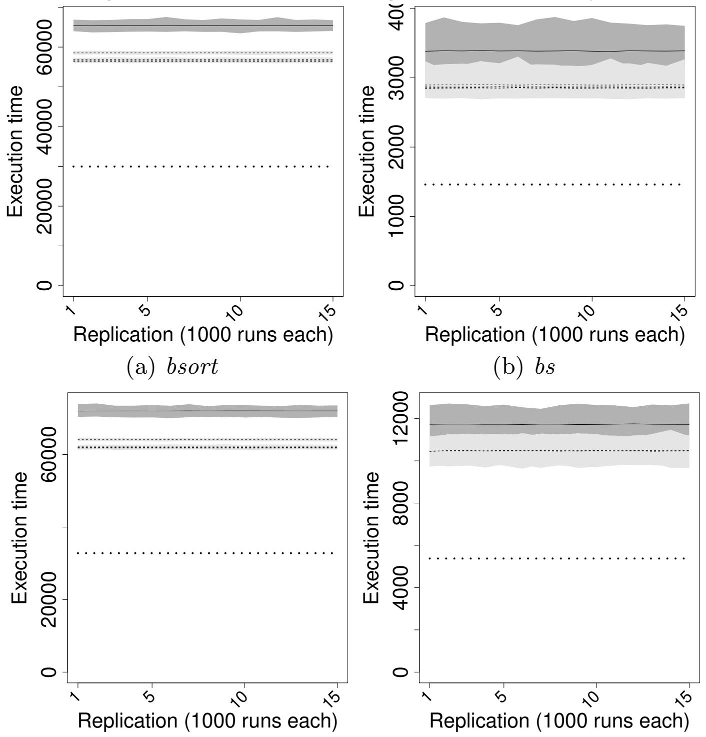
6.7.4.2 Class Timing Dominance

The plots shown in Figure 46 present the mean, the minimum and the maximum execution times observed while the benchmarks execute (1) in the maximum interference scenario (i.e. under interference of FEAH instructions), as a continuous line in a dark grey region, and (2) under interference of instructions of classes other than FEAH, as superposed dashed lines on light grey regions. The benchmarks' solo execution times are highlighted as a straight dotted line, for slowdown evaluation purposes. Each of the employed execution time samples is composed of 1000 measurements and was replicated 15 times. The plots provide evidence that the class timing dominance property holds when the IRS is used, since tasks' execution times yielded under interference of instructions of classes other than FEAH are dominated by those produced in the maximum interference scenario. Class timing dominance plots associated with other benchmarks can be found in Appendix B.

6.7.4.3 Behavioural Timing Dominance

The plots presented in Figure 47 show the mean, the minimum and the maximum execution times observed while the benchmark tasks are executed (1) in a large number of distinct interference conditions (see Section 6.5 for details), as a continuous line in a light grey region, and (2) in the maximum interference scenario, as a straight continuous line surrounded by dashed lines. Approximately 25000 scenarios were evaluated using samples composed of 100 execution times, and therefore only those that produced the top-1000 highest mean execution times are considered in order to improve plots' readability. Benchmark tasks' solo execution time is highlighted as a straight dotted line, for slowdown evaluation purposes. The plots provide evidence that the

Figure 46: IRS class timing dominance analysis



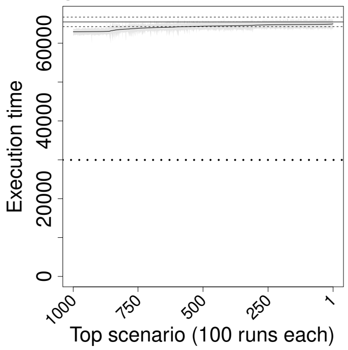
(a) *bsort* (b) *bs* (c) *cnt* (d) *cover*

behavioural timing dominance property holds when the IRS is used, since benchmarks' execution time distributions produced in numerous execution conditions are dominated by those observed in the maximum interference scenario. Behavioural timing dominance plots associated with all other benchmark tasks can be found in Appendix B.

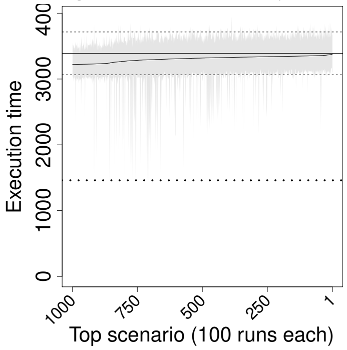
6.7.4.4 Interference Balancing

Table 4 presents, for each of the considered benchmark tasks, its solo execution time and the slowdown factor observed for its average execution times in the maximum interference scenario in relation to solo execution. This information is presented both with the processor executing in measurement mode, i.e. with logical units producing maximum latencies, and under normal execution conditions. Based

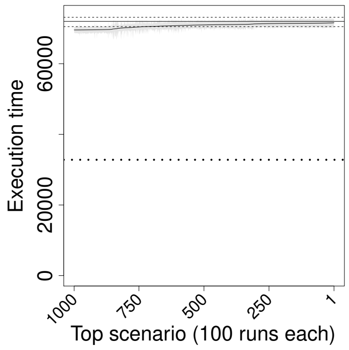
Figure 47: IRS behavioural timing dominance analysis



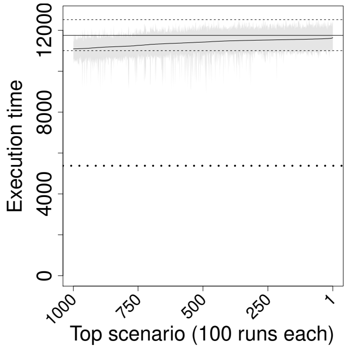
(a) *bsort*



(b) *bs*



(c) *cnt*



(d) *cover*

on its analysis we observe that the IRS can be deemed to provide improved interference balancing characteristics, in comparison with the PRS approach taken as baseline. Consider for instance the *bsort* benchmark, which executes solo in ≈ 30000 clock cycles and takes in average ≈ 70000 cycles to execute in the maximum interference scenario. The slowdown factor of ≈ 2.5 using IRS can be considered reasonable, if compared with the $t = 2$ ideal case, and is in fact much smaller than that of ≈ 6 observed with PRS (as shown in Section 6.6.1.4). We hence conclude that the proposed IRS is capable of limiting inter-thread interference effects in the maximum interference scenario, and does so without compromising execution times' analysability through MBPTA.

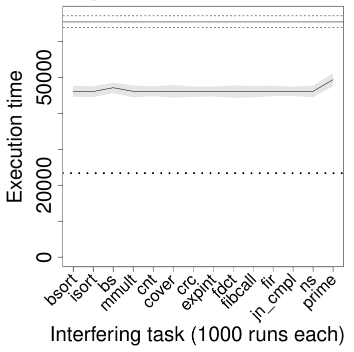
Task	Measurement mode		Execution mode	
	Solo execution	Maximum slowdown	Solo execution	Maximum slowdown
bsort	29967	2.18	23352	2.80
isort	25946	2.18	20762	2.72
bs	1460	2.32	1310	2.58
mmult	54292	2.26	36692	3.34
cnt	32802	2.20	25602	2.81
cover	5377	2.18	5377	2.18
crc	10092	2.12	9372	2.29
expint	36896	2.11	31588	2.47
fdct	56074	2.17	41540	2.93
fibcall	4457	2.17	4457	2.17
fir	65640	2.13	60614	2.31
jn_cmpl	4916	2.15	4812	2.20
ns	29244	2.21	22437	2.89
prime	16017	2.15	14392	2.39

Table 4: IRS maximum slowdown

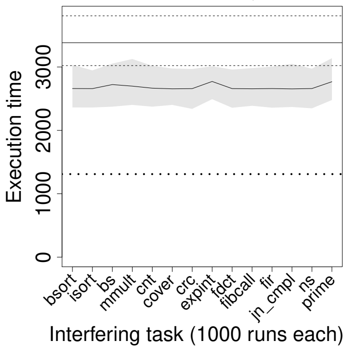
6.7.4.5 Typical Scenario Slowdown

The plots shown in Figure 48, whose reading is similar to that of the behavioural timing dominance plots of Section 6.7.4.3, compare the distributions of 1000 execution times yielded while each benchmark executes under interference of real tasks (the same set of benchmarks) against those of solo execution and of the maximum interference scenario. These plots' analysis shows that the slowdown factor threads experience under typical conditions while the IRS is employed proves acceptable (i.e. approximates the ideal value t), despite in the maximum interference scenario it is slightly higher. Moreover, it remains very similar to those witnessed when the PRS is employed instead (see Section 6.6.1.5). Consider for instance the *bsort* benchmark task (see Figure 48(a)), which takes $\approx$ 30000 clock cycles to execute solo. Its mean execution time in typical execution conditions remain (A) around $\approx$ 51000 cycles when PRS is used, and (B) close to $\approx$ 50000 cycles using IRS. We hence conclude that the IRS is capable of limiting inter-thread timing interference without negatively affecting the execution time distributions observed in typical scenarios. Similar plots, associated with the other considered benchmark tasks, can be found in Appendix B. Summarized information on IRS's typical scenario slowdowns is found in Table 5, which shows, for each of the benchmark tasks, its solo execution time and the maximum slowdown factor it was observed to experience in the evaluated typical scenarios – considering their execution both in measurement mode and in normal mode.

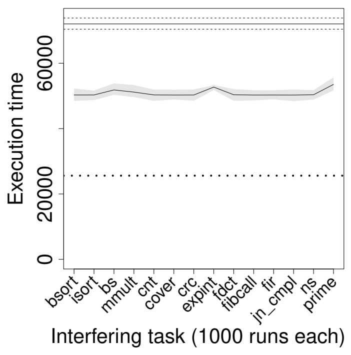
Figure 48: IRS typical scenario slowdown analysis



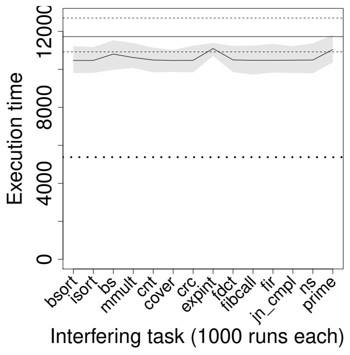
(a) *bsort*



(b) *bs*



(c) *cnt*



(d) *cover*

Task	Measurement mode		Execution mode	
	Solo execution	Typical slowdown	Solo execution	Typical slowdown
bsort	29967	1.65	23352	2.11
isort	25946	1.67	20762	2.08
bs	1460	1.90	1310	2.12
mmult	54292	1.41	36692	2.09
cnt	32802	1.63	25602	2.09
cover	5377	2.06	5377	2.06
crc	10092	1.93	9372	2.08
expint	36896	1.74	31588	2.03
fdct	56074	1.56	41540	2.11
fibcall	4457	2.09	4457	2.09
fir	65640	1.88	60614	2.04
jn_cmpl	4916	2.02	4812	2.06
ns	29244	1.60	22437	2.09
prime	16017	1.79	14392	1.99

Table 5: IRS typical slowdown

6.8 TIMING ANALYSIS COMPARISON

In this section we present a comparison of the previously evaluated randomized thread scheduling approaches, namely PRS and IRS, with respect to the timing analysis of execution times produced in the maximum interference scenario. For that we assess characteristics of the samples of size 50000 used in Sections 6.6.1.1 and 6.7.4.1, and the results obtained from their analysis through MBPTA considering the application of EVT using both BM and POT.

Table 6 presents, for each benchmark task executing in the maximum interference scenario using both PRS and IRS, (1) the average, minimum and maximum execution times observed, (2) the standard deviation of the produced execution times, and (3) the reduction factor observed for IRS in relation to PRS with respect to execution times' average value and standard deviation. From its analysis we conclude that the proposed IRS doesn't only produce average execution times that are smaller in relation to those of PRS by a factor that often proves higher than ≈ 3, but it is also capable of reducing their standard deviation (dispersion) by factors as high as ≈ 6.

Tables 7 and 8 show information associated with the application of MBPTA, using BM and POT respectively, on the benchmark tasks' execution times yielded in the maximum interference scenario using both PRS and IRS. In line with the recommendations of Chapter 5, we apply EVT within MBPTA by (1) fitting GEV and GP models by estimating their location (μ)/threshold (τ), scale (σ) and shape (ξ) parameters, for evaluating whether execution times' maxima present right tails that decrease at least exponentially (i.e. $\xi < 0$), and (2) using the Gumbel and Exponential models for effectively producing pWCET estimates (for BM and POT approaches, respectively). Despite this approach being likely to introduce some level of pessimism, it also grants pWCET estimates with higher reliability.

Table 7 presents (1) estimates for the location (μ) and scale (σ), and 95% confidence intervals for the shape (ξ), of the GEV distribution adjusted to the samples' maxima using blocks of size 50, (2) the resulting pWCET estimates (γ) with exceedance probability 10^{-15}, and (3) the reduction factors of μ, σ, ξ and γ observed for IRS in relation to PRS. Similarly, Table 8 presents (1) estimates for the scale (σ) and 95% confidence intervals for the shape (ξ) of the GP distribution adjusted to the samples' maxima that exceed the threshold (τ) defined through GP estimated quantiles' mean absolute error minimization (see Section 2.4), (2) the resulting pWCET estimates (γ) with exceedance probability 10^{-15}, and (3) the reduction factors of τ, σ, ξ and γ observed for IRS in comparison with the PRS taken as baseline.

From the analysis of Tables 7 and 8, we observe that (A) the maxima shape (ξ) remains consistently negative and often present no significant changes for IRS in relation to PRS, and that (B) the reduction in the execution times' average values and dispersion IRS provides in relation to PRS also causes the location (μ)/threshold (τ) and the scale (σ) of the produced maxima distributions to be reduced by significant amounts – i.e. by factors as high as ≈ 3 and ≈ 8, respectively. From (A), it follows that MBPTA application with increased reliability, as recommended in Chapter 5, is feasible using either PRS or IRS. As a consequence of both (A) and (B), the yielded pWCET estimates (γ) are reduced by factors between ≈ 2 and ≈ 4 for IRS in relation to PRS.

Task	Purely Random Scheduler (PRS)				Interference-Regulated Scheduler (IRS)				Average reduction factor	Std. dev. reduction factor
	Minimum	Average	Maximum	Standard deviation	Minimum	Average	Maximum	Standard deviation		
bsort	177042	190162.90	204373	3328.52	63543	65389.40	67550	496.88	2.91	6.70
isort	153374	167215.53	180977	3152.86	54732	56447.08	58207	464.95	2.96	6.78
bs	7051	10198.62	14026	762.97	2980	3384.42	4021	120.40	3.01	6.34
mmult	290141	307864.95	324885	4212.46	119981	122453.11	125423	863.58	2.51	4.88
cnt	196120	209719.26	225005	3506.17	69946	72023.83	74017	543.62	2.91	6.45
cover	36584	42564.45	49662	1594.51	10698	11731.79	12798	266.60	3.63	5.98
crc	64958	73651.28	83278	2113.36	20131	21443.13	22980	349.24	3.43	6.05
expint	221710	240016.50	256634	3732.31	75395	77892.78	80578	573.00	3.08	6.51
fdct	316381	335702.78	353981	4433.85	119135	121912.48	124838	815.28	2.75	5.44
fibcall	28962	35005.33	41565	1441.93	8731	9671.49	10683	236.53	3.62	6.10
fir	451360	472750.09	494177	5297.45	136084	139858.20	143692	859.87	3.38	6.16
jn_cmpl	32162	38022.17	44528	1507.45	9588	10585.87	11707	249.40	3.59	6.04
ns	173568	186781.04	199818	3242.42	62910	64730.35	66745	500.65	2.89	6.48
prime	96192	107327.08	118281	2482.29	32795	34454.97	36032	372.65	3.11	6.66

Table 6: Raw dataset analysis

Task	Purely Random Scheduler (PRS)				Interference-Regulated Scheduler (IRS)				μ	σ	ξ	γ
	Location	Scale	Shape	pWCET	Location	Scale	Shape	pWCET	reduction	reduction	reduction	reduction
	(μ)	(σ)	(ξ)	(γ)	(μ)	(σ)	(ξ)	(γ)	factor	factor	factor	factor
bsort	197043.18	1467.38	$-0.18<-0.13<-0.08$	246500	66391.32	199.17	$-0.17<-0.12<-0.09$	73162	2.97	7.37	1.03	3.37
isort	173774.89	1391.37	$-0.15<-0.11<-0.06$	220363	57384.97	178.99	$-0.15<-0.10<-0.06$	63365	3.03	7.77	1.05	3.48
bs	11840.89	339.32	$-0.14<-0.10<-0.06$	23376	3650.04	59.14	$-0.15<-0.11<-0.06$	5640	3.24	5.74	0.88	4.14
mmult	316602.25	1791.97	$-0.22<-0.17<-0.13$	376640	124096.95	236.27	$-0.20<-0.16<-0.12$	132109	2.55	7.58	1.09	2.85
cnt	216978.71	1563.71	$-0.17<-0.13<-0.08$	269921	73101.39	202.60	$-0.16<-0.12<-0.09$	79891	2.97	7.72	1.03	3.38
cover	45890.64	672.54	$-0.15<-0.11<-0.06$	68715	12287.71	111.24	$-0.15<-0.11<-0.07$	16163	3.73	6.05	0.96	4.25
crc	78055.75	896.29	$-0.19<-0.14<-0.10$	108346	22171.84	148.41	$-0.15<-0.11<-0.06$	27152	3.52	6.04	1.33	3.99
expint	247709.50	1612.69	$-0.18<-0.13<-0.09$	302618	79079.32	254.26	$-0.16<-0.11<-0.07$	87546	3.13	6.34	1.19	3.46
fdct	344925.97	1834.47	$-0.15<-0.11<-0.07$	406950	123460.69	256.84	$-0.17<-0.13<-0.08$	132338	2.79	7.14	0.86	3.08
fibcall	38022.77	608.41	$-0.14<-0.10<-0.06$	58455	10166.17	100.13	$-0.14<-0.10<-0.05$	13617	3.74	6.08	1.01	4.29
fir	483741.73	2175.20	$-0.16<-0.11<-0.07$	556511	141621.63	363.04	$-0.14<-0.09<-0.05$	153912	3.42	5.99	1.24	3.62
jn_cmpl	41187.75	615.58	$-0.14<-0.09<-0.05$	62420	11103.45	107.70	$-0.15<-0.11<-0.07$	14776	3.71	5.72	0.85	4.22
ns	193587.44	1409.31	$-0.20<-0.16<-0.11$	240208	65742.60	201.77	$-0.16<-0.12<-0.08$	72666	2.94	6.98	1.36	3.31
prime	112454.55	1063.74	$-0.15<-0.10<-0.06$	148482	35232.14	152.20	$-0.16<-0.11<-0.06$	40433	3.19	6.99	0.90	3.67

Table 7: BM timing analysis

Task	Purely Random Scheduler (PRS)				Interference-Regulated Scheduler (IRS)				τ	σ	ξ	γ
	Threshold (τ)	Scale (σ)	Shape (ξ)	pWCET (γ)	Threshold (τ)	Scale (σ)	Shape (ξ)	pWCET (γ)	reduction factor	reduction factor	reduction factor	reduction factor
bsort	195246	1701.61	$-0.16<-0.13<-0.09$	247437	66287	196.26	$-0.14<-0.09<-0.03$	74036	2.95	8.67	1.47	3.34
isort	171258	1708.18	$-0.15<-0.12<-0.08$	224021	57176	208.00	$-0.16<-0.12<-0.07$	63632	3.00	8.21	1.02	3.52
bs	11376	408.17	$-0.16<-0.12<-0.08$	24519	3547	74.32	$-0.18<-0.14<-0.10$	5799	3.21	5.49	0.84	4.23
mmult	313515	2276.27	$-0.20<-0.16<-0.13$	381277	124137	212.83	$-0.18<-0.10<-0.03$	133365	2.53	10.70	1.66	2.86
cnt	214369	1895.63	$-0.17<-0.13<-0.10$	272602	73017	227.78	$-0.23<-0.16<-0.10$	79783	2.94	8.32	0.79	3.42
cover	45005	798.88	$-0.16<-0.12<-0.08$	69738	12202	117.38	$-0.15<-0.09<-0.04$	15778	3.69	6.81	1.26	4.42
crc	76189	1224.98	$-0.20<-0.17<-0.14$	112525	22019	170.88	$-0.18<-0.13<-0.09$	28170	3.46	7.17	1.27	3.99
expint	246914	1597.18	$-0.15<-0.09<-0.03$	312745	78622	305.45	$-0.15<-0.11<-0.08$	88096	3.14	5.23	0.82	3.55
fdct	342108	2307.94	$-0.18<-0.14<-0.10$	412783	123136	302.83	$-0.17<-0.13<-0.09$	132377	2.78	7.62	1.03	3.12
fibcall	37119	738.30	$-0.16<-0.12<-0.09$	59767	10053	115.53	$-0.16<-0.12<-0.07$	13629	3.69	6.39	1.06	4.39
fir	481733	2418.72	$-0.17<-0.12<-0.07$	556434	141217	399.72	$-0.13<-0.09<-0.05$	153868	3.41	6.05	1.28	3.62
jn_cmpl	40634	704.29	$-0.16<-0.11<-0.06$	64138	11016	114.98	$-0.15<-0.10<-0.06$	14618	3.69	6.13	1.08	4.39
ns	190354	1952.15	$-0.19<-0.16<-0.13$	248432	65418	238.96	$-0.15<-0.11<-0.08$	72840	2.91	8.17	1.44	3.41
prime	111370	1154.89	$-0.14<-0.10<-0.05$	147752	35200	147.99	$-0.14<-0.07<-0.01$	41952	3.16	7.80	1.37	3.52

Table 8: POT timing analysis

6.9 IMPLEMENTATION EFFORT

Developing both the maximum interference scenario and the interference detection logic we employ in this work have demanded careful analysis and adaptations. Establishing the maximum interference scenario required classifying the processor's instructions with respect to their maximum potential inter-thread interference, which depends on the timing of hardware elements used both within and outside the processing core (e.g. ALU and memory hierarchy). It has also required hardware adaptations, e.g. for enabling the execution of ghost instructions and for fixing elements' latencies to effectively induce maximum interference. In turn, implementing the interference detection logic required a careful analysis of the pipeline's internal dynamics while traversed by instructions belonging to different threads. These tasks are critical, and their execution can also prove quite challenging, for implementing the approaches proposed in this work in processors equipped with hardware elements of higher complexity.

In terms of hardware complexity/cost, summary information and a simplified comparison between PRS and IRS is presented in Table 9. The table shows the number and the percentage increase, for IRS in relation to PRS, of logic elements, combinational functions and logic registers used for synthesizing the employed multithread processor with the proposed thread schedulers, for execution on Altera FPGAs.

Hardware elements	Scheduler		Increase
	PRS	IRS	
Logic elements	19256	20107	4,42%
Combinational functions	17490	18243	4,30%
Logic registers	7648	7988	4,45%

Table 9: Schedulers' hardware complexity/cost

6.10 CONCLUSION

In this chapter we evaluate whether the application of randomization techniques to perform thread scheduling on multithread pipelines can benefit their timing analysis through MBPTA. For that we (A) designed a simple multithread pipelined core, (B) established the conditions in which maximum inter-thread interference is observable, (C) proposed a method to evaluate key properties MBPTA-targeted multithread processors must present, and (D) assessed two distinct randomized thread scheduling approaches. We first evaluated PRS, a simple purely-random scheduler, and observed that (i) it leads to execution times that meet the basic MBPTA applicability requirements, but also that (ii) it allows threads being severely slowed down in the maximum inter-thread interference scenario. We then introduced IRS, a scheduler that employs a credit-based eligibility regulation mechanism coupled with an inter-thread interference detection logic, and then observed that (I) it also produces execution times that meet MBPTA's basic requirements, (II) it is capable of limiting the delays experienced by threads due to inter-thread interference, and consequently (III) the resulting pWCET estimates produced through MBPTA when IRS is used are also substantially reduced. We have also observed that, when randomized scheduling is employed, the threads' slowdown in typical scenarios – i.e. under interference of real tasks – remains within acceptable ranges when using either PRS or IRS.

With that, we conclude that (A) time-randomization is, in principle, capable of benefiting the analysability of multithread pipelines through MBPTA when applied at the thread scheduling level, by leading to execution times that meet its basic requirements in the scenario of maximum inter-thread interference; (B) in the presence of instructions whose latencies in traversing a randomly scheduled multithread pipeline largely differ (e.g. NOOPs and MULTs), a great unbalancing in the allocation of processing power is observed if no mechanism is employed to limit inter-thread interference effects; and (C) credit-based eligibility regulation is a plausible approach for limiting the delays experienced by threads due to inter-thread interference, which was evidenced to reduce the mean execution times and to limit their dispersion in comparison with the baseline purely random scheduling approach (even in the maximum inter-thread interference scenario).

We highlight that processors whose architectures differ from the one employed in this work can produce execution times whose characteristics deviate from those we have observed. For this reason, we strongly recommend the evaluation method proposed in Section 6.5 being applied on future multithread pipeline designs targeted to probabilistic timing analysis, as a means to produce evidence that their behaviour is in fact compatible with their utilization within MBPTA. We also underline that the analysability evidence shown in this work cannot be used as general proof for supporting the use of MBPTA under similar conditions. The production of applicability evidence is an integral part of MBPTA, and it must hence be performed on a per-application basis.

7 FINAL REMARKS

The complexity of RTSs is growing fast as modern applications emerge, hence causing the demand for WCET-analysable processing power to increase accordingly. Modern processors capable of delivering such high processing capacity employ numerous acceleration hardware elements, which cause their behavioural complexity to easily prove intractable within traditional timing analysis techniques. This is so because these methods must deal with hardware models that either (1) present complexity that grow fast as acceleration mechanisms are introduced, which easily hampers such techniques' computational feasibility, or (2) must conservatively abstract hardware constructive details for ensuring computational feasibility, which potentially leads to high pessimism on the produced WCET estimates. Static approaches are also extremely sensitive to hardware changes, since even small design improvements require large efforts for adjusting the models and the analysis process accordingly (WILHELM et al., 2008).

A recently proposed methodology, called Measurement-Based Probabilistic Timing Analysis (MBPTA), promises improving this scenario by replacing the joint hardware-software analysis of static approaches with probabilistic analyses of execution time measurements. Its basic principle is the application of Extreme Value Theory (EVT), a statistics framework designed to estimate the probabilities associated with the occurrence of unusual extreme events, on the execution times of RTSs' tasks. One of the most attractive aspects of MBPTA is its potentiality to abstract hardware complexity, which could ultimately enable arbitrarily complex processors being used within RTSs. It is hence emerging as an attractive solution for industry, also due to its relatively low cost and complexity in comparison with traditional approaches (HERNANDEZ et al., 2015; CAZORLA et al., 2016). In particular, the emergence of modern applications such as autonomous cars and Internet of Things (IoT) devices may lead to significant increases on the demand for high-performance processors suitable for running RTSs, i.e. to which reliable and affordable timing analysis techniques are readily available (SLIJEPCEVIC et al., 2016).

For being applied, MBPTA imposes a number of requirements on the conditions under which measurements are taken, and also on the observed data. It requires, for instance, (A) measured execution times to present independence and identical distribution characteristics, (B) measurements being taken under either realistic or pessimistic conditions with respect to the system's real operation environment, and (C) the observed execution times' maxima adhering to the extreme value distributions used by EVT (LIMA; DIAS; BARROS, 2016).

Time-randomized processors are built replacing, by design, internal information that influence execution times – which are typically deterministic or speculative in traditional processor designs – with (pseudo-)random numbers. For instance, such processors may use (1) cache memories with random placement and/or replacement policies (KOSMIDIS et al., 2013a), (2) bus arbiters with randomized client scheduling (LAHIRI; RAGHUNATHAN; LAKSHMINARAYANA, 2001; JALLE et al., 2014), and/or (3) NoCs whose routing employs random decisions (SLIJEPCEVIC et al., 2016). Time-randomized processors were recently shown to better fit the needs of probabilistic timing analysis techniques, since they produce execution times whose maxima often proves modellable through the extreme value distributions used by EVT (CAZORLA et al., 2013b; KOSMIDIS et al., 2016).

The research whose outcomes are presented in this thesis produced contributions in the context of MBPTA on two distinct fronts.

Firstly, we proposed methods for evaluating the reliability of probabilistic WCET estimates (pWCETs) yielded by MBPTA. The proposed methods are based on collecting large execution time samples (i.e. of size 10^8) and then comparing (1) pWCET estimates against the largest observed execution times, and (2) pWCET exceedance densities against their expected values. Reliability evaluations led us to conclude that EVT probabilistic models intended to yield more precise bounds (i.e. GEV and GP) may often lead to underestimations. For this reason, recommendations were drawn that precise models should be used mainly for diagnosing execution times' maxima tails' shapes, and tail density upper-bounding models (i.e. Gumbel and Exponential) should then be used for effectively deriving reliable pWCETs.

Secondly, we evaluated whether randomized thread scheduling approaches are capable of benefiting the application of MBPTA to analyse multiple tasks which are simultaneously executed on multithread pipelines. For that we evaluated (1) a baseline scheduler that employs a purely random policy, and (2) an interference-regulated scheduler that employs a credit-based eligibility regulation mechanism coupled to an inter-thread interference detection logic. Evaluations led us to conclude that (A) thread scheduling randomization potentially enables multithread pipelines producing EVT-suitable execution times, but that (B) purely random scheduling does not balance interference-related delays, leading to unbalanced processing provision, while (C) credit-based regulation is potentially a suitable approach for limiting inter-thread interference on time-randomized multithread pipelines without compromising analysability through MBPTA.

This thesis provides valuable contributions in the context of MBPTA, both by evaluating and improving the reliability of its outcomes and by assessing the introduction of randomization at hardware level for facilitating its application. Particularly, our reliability evaluation methods showed that MBPTA requires probabilistic models being used such that pWCET estimates are likely to upper-bound (and not only estimate) the maximum observable execution times. Moreover, the herein presented work explored original research paths involving multithreading, a technique that is often avoided in RTSs' design since it either potentially (1) leads to extreme pessimism in WCET estimation through static approaches, or (2) requires timing guarantees being provided only for a single thread whose execution is generally prioritized. The presented work can therefore be considered to provide relevant contributions, which mainly targeted increasing the reliable applicability of MBPTA and promoting its usability for the timing analysis of RTSs executed on multithread pipelines.

7.1 GENERAL CONSIDERATIONS

Despite static timing analysis techniques being capable of solving numerous WCET-related issues even for relatively complex processors (WILHELM et al., 2008), they lack composability characteristics needed for being applied in modern (highly complex) hardware platforms. Since temporal variability is vast in such processors due to numerous hardware- and software-related aspects, timing analysis through statistical tools is potentially adequate and promising. Moreover, it is a relatively common practice in other fields determining safety margins through statistical techniques such as EVT (COLES, 2001), which supports the potential adequacy of the approach. However, intrinsic differences between computing systems and other phenomena in which EVT has been traditionally applied must be handled carefully.

The main challenges of MBPTA arise from the fact that typical execution environments give no clues regarding execution times' behaviour under extreme conditions. This obligates computing systems being induced into scenarios that can be deemed either representative or pessimistic with respect to the worst-case environment during execution times' sampling – which usually proves quite hard. In this regard, there are a number of controllable factors – mainly associated with tasks' input data – that largely influence execution times and that have no direct counterpart in other fields where EVT is used. These factors must therefore be carefully manipulated for obtaining measurements that, when analysed through EVT, in fact lead to safety margins that are globally reliable in relation to tasks' WCETs. On the other hand, there is also a significant benefit MBPTA has in relation to other areas in which EVT is used. In other fields, the collection of arbitrarily large samples easily proves too hard and costly, or can even be infeasible. This prevents practitioners from performing experiments that enable evidencing and/or quantifying the reliability of safety margins produced using EVT. On the other hand, large execution time samples can be collected with relatively small cost, enabling evaluations such as those proposed in this thesis to be performed.

The empirical experiments performed during the herein presented research using time-randomized processors suggest that their use in fact make it easier applying MBPTA. The timing variability that arises using randomization techniques produces smoother execution time distributions in comparison with traditional designs, which ultimately leads to maxima more easily modellable through EVT. Moreover, in this work time-randomization proved successful in enabling MBPTA being applied on a processor whose design is hardly analysable through traditional approaches (i.e. using a multithread pipeline). In such designs, static timing analysis often needs assuming extremely unrealistic and pessimistic local worst-cases, leading to extreme pessimism on the produced WCET upper-bounds. Randomization allowed us using a regulation mechanism whose static analysis would be challenging, but which enabled pWCET estimates with acceptable slack in relation to ideal values being transparently produced. This suggests that randomization techniques are indeed promising in enabling hardware designs of increasing complexity being handled within MBPTA, for totally abstracting their constructive details. A clear drawback of time-randomization regards performance reductions that may arise, e.g., as work-conserving resource arbitration policies are randomized. We should consider, however, that (A) reasonable performance reductions are an acceptable price for enabling the production of reliable pWCET bounds, and (B) penalties also exist when static methods are used in timing analysis, which show up mainly in the form of sub-utilization.

7.2 PUBLICATIONS

The following papers were published based on the research work whose outcomes are presented in this thesis:

SILVA, K. P.; ARCARO, L. F.; OLIVEIRA, R. S. de. On Using GEV or Gumbel Models when Applying EVT for Probabilistic WCET Estimation. In: **Real-Time Systems Symposium 2017 (RTSS'17)**. IEEE, 2017. p. 220–230.

ARCARO, L. F.; SILVA, K. P.; OLIVEIRA, R. S. de. On the Reliability and Tightness of GP and Exponential Models for Probabilistic WCET Estimation. **ACM Transactions on Design Automation of Electronic Systems (TODAES)**, ACM, v. 23, p. 39:1–39:27, 2018.

ARCARO, L. F.; SILVA, K. P.; OLIVEIRA, R. S. de. A Reliability Evaluation Method for Probabilistic WCET Estimates based on the Comparison of Empirical Exceedance Densities. In: **Brazilian Symposium on Computing Systems Engineering 2018 (SBESC'18) – Work-in-Progress**. IEEE, 2018.

The following papers were submitted for publication based on the research work presented in this thesis:

ARCARO, L. F.; SILVA, K. P.; OLIVEIRA, R. S. de. On Using Randomized Scheduling on Multithread Pipelines for Benefiting Probabilistic Timing Analysis. **ACM Transactions on Embedded Computing Systems (TECS)**, ACM. Submitted on January 14, 2019.

7.3 FUTURE WORK

There are many directions in which the herein presented research can be extended in future works, especially concerning the introduction of randomization at hardware level for benefiting MBPTA. For the IRS element proposed in Chapter 6 we consider, for instance, performing (A) further empirical evaluations using processors with e.g. more than two threads and/or multiple cores, (B) optimized calibrations and sensitivity analyses of its parameters, (C) assessments regarding the use of unbalanced credits between threads e.g. for providing increased performance and/or reduced worst-case slowdown for particular threads, and (D) the derivation of formal/theoretical explanations for properties shown through empirical data in this thesis. We also consider (E) designing a randomized thread scheduler in which the probabilities of threads being chosen are smoothly decreased as they induce interference on others, instead of using binary suspensions as performed by the proposed scheduler, (F) scaling time-randomized processors equipped with multithread pipelines into multi-/many-core configurations, and evaluating the limitations such designs are subject with respect to execution time increases observed in the maximum interference scenario, (G) coupling peripherals to time-randomized processors, and evaluating whether timing analysability or performance limitations may arise from their introduction, and (H) evaluating the design of a time-randomized multi-/many-core processor equipped with a stand-alone task scheduler, which could prove capable of executing multiple tasks in parallel while enabling probabilistic timing guarantees being provided for all of them.

7.4 ACKNOWLEDGEMENT

This research was partially funded by CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) and CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico).

REFERENCES

ABELLA, J. et al. On the Comparison of Deterministic and Probabilistic WCET Estimation Techniques. In: **Euromicro Conference on Real-Time Systems 2014 (ECRTS'14)**. [S.l.]: IEEE, 2014. p. 266–275.

ABELLA, J. et al. WCET analysis methods: Pitfalls and challenges on their trustworthiness. In: **International Symposium on Industrial Embedded Systems 2015 (SIES'15)**. [S.l.]: IEEE, 2015. p. 1–10.

ABELLA, J. et al. Measurement-Based Worst-Case Execution Time Estimation Using the Coefficient of Variation. **ACM Transactions on Design Automation of Electronic Systems (TODAES)**, ACM, v. 22, p. 72:1–72:29, 2017.

ABELLA, J. et al. **Understanding MBPTA and its requirements on program instructions**. [S.l.], 2013. 11 p.

ABELLA, J. et al. Heart of Gold: Making the Improbable Happen to Increase Confidence in MBPTA. In: **Euromicro Conference on Real-Time Systems 2014 (ECRTS'14)**. [S.l.]: IEEE, 2014. p. 255–265.

AGIRRE, I. et al. Fitting Software Execution-Time Exceedance into a Residual Random Fault in ISO-26262. **IEEE Transactions on Reliability**, IEEE, p. 14, 2018.

AKESSON, B. et al. Composability and Predictability for Independent Application Development, Verification, and Execution. In: HÜBNER, M.; BECKER, J. (Ed.). **Multiprocessor System-on-Chip**. [S.l.]: Springer, 2011. p. 25–56.

AKESSON, B.; STEFFENS, L.; GOOSSENS, K. Efficient Service Allocation in Hardware Using Credit-Controlled Static-Priority Arbitration. In: **International Conference on Embedded and Real-Time Computing Systems and Applications 2009 (RTCSA'09)**. [S.l.]: IEEE, 2009. p. 59–68.

ALFKE, P. **Efficient Shift Registers, LFSR Counters, and Long Pseudo-Random Sequence Generators**. [S.l.], 1996. 6 p.

ALTMEYER, S. et al. Evaluation of Cache Partitioning for Hard Real-Time Systems. In: **Euromicro Conference on Real-Time Systems 2014 (ECRTS'14)**. [S.l.]: IEEE, 2014. p. 15–26.

ANWAR, H. **A Probabilistically Analyzable Cache to Estimate Timing Bounds**. 85 p. Dissertação (Mestrado), 2016.

ARCARO, L. F.; SILVA, K. P.; OLIVEIRA, R. S. de. A Reliability Evaluation Method for Probabilistic WCET Estimates based on the Comparison of Empirical Exceedance Densities. In: **Brazilian Symposium on Computing Systems Engineering 2018 (SBESC'18)**. [S.l.]: IEEE, 2018. p. 6.

ARCARO, L. F.; SILVA, K. P.; OLIVEIRA, R. S. de. On the Reliability and Tightness of GP and Exponential Models for Probabilistic WCET Estimation. **ACM Transactions on Design Automation of Electronic Systems (TODAES)**, ACM, v. 23, p. 39:1–39:27, 2018.

BAETONIU, C. **High Speed True Random Number Generators in Xilinx FPGAs**. 2004. Disponível em: <http://forums.xilinx.com/xlnx/attachments/xlnx/Virtex/24484/1/High Speed True Random Number Generators in Xilinx FPGAs.pdf>.

BALKEMA, A. A.; HAAN, L. de. Residual Life Time at Great Age. **The Annals of Probability**, Institute of Mathematical Statistics, v. 2, p. 792–804, 1974.

BALL, T.; LARUS, J. R. Branch Prediction for Free. In: **Conference on Programming Language Design and Implementation 1993 (PLDI'93)**. [S.l.]: ACM, 1993. p. 300–313.

BEIRLANT, J. et al. **Statistics of Extremes: Theory and Applications**. [S.l.]: Wiley, 2004. 504 p. (Wiley Series in Probability and Statistics).

BENEDICTE, P. et al. RPR: A Random Replacement Policy with Limited Pathological Replacements. In: **Symposium on Applied Computing 2018 (SAC'18)**. [S.l.]: ACM, 2018. p. 593–600.

BENEDICTE, P. et al. A confidence assessment of WCET estimates for software time randomized caches. In: **International Conference on Industrial Informatics 2016 (INDIN'16)**. [S.l.]: IEEE, 2016. p. 90–97.

BENEDICTE, P. et al. Modelling the confidence of timing analysis for time randomised caches. In: **International Symposium on Industrial Embedded Systems 2016 (SIES'16)**. [S.l.]: IEEE, 2016. p. 1–8.

BUI, B. D.; CACCAMO, M.; PELLIZZONI, R. A Slot-Based Real-Time Scheduling Algorithm for Concurrent Transactions in NoC. In: **International Conference on Embedded and Real-Time Computing Systems and Applications 2011 (RTCSA'11)**. [S.l.]: IEEE, 2011. v. 1, p. 329–338.

BUI, D. et al. Temporal isolation on multiprocessing architectures. In: **Design Automation Conference 2011 (DAC'11)**. [S.l.]: ACM, 2011. p. 274–279.

BURNS, A.; EDGAR, S. Predicting computation time for advanced processor architectures. In: **Euromicro Conference on Real-Time Systems 2000 (ECRTS'00)**. [S.l.]: IEEE, 2000. p. 89–96.

ČAKAREVIĆ, V. et al. Characterizing the resource-sharing levels in the UltraSPARC T2 processor. In: **International Symposium on Microarchitecture 2009 (MICRO-42)**. [S.l.]: ACM, 2009. p. 481–492.

CAZORLA, F. J. et al. PROXIMA: Improving Measurement-Based Timing Analysis through Randomisation and Probabilistic Analysis. In: **Euromicro Conference on Digital System Design 2016 (DSD'16)**. [S.l.]: IEEE, 2016. p. 276–285.

CAZORLA, F. J. et al. Reconciling Time Predictability and Performance in Future Computing Systems. **IEEE Design & Test**, IEEE, PP, p. 1, 2017.

CAZORLA, F. J. et al. PROARTIS: Probabilistically Analyzable Real-Time Systems. **ACM Transactions on Embedded Computing Systems (TECS)**, ACM, v. 12, p. 94:1–94:26, 2013.

CAZORLA, F. J. et al. Upper-bounding Program Execution Time with Extreme Value Theory. In: **International Workshop on Worst-Case Execution Time Analysis 2013 (WCET'13)**. [S.l.]: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2013. v. 30, p. 64–76.

COLES, S. G. **An Introduction to Statistical Modeling of Extreme Values**. 1st. ed. [S.l.]: Springer, 2001. 219 p. (Springer Series in Statistics).

COLES, S. G.; DIXON, M. J. Likelihood-Based Inference for Extreme Value Models. **Extremes**, Springer, v. 2, p. 5–23, 1999.

CONMY, P. R. et al. Measurement-Based Probabilistic Timing Analysis - From Academia to Space Industry. In: **Data Systems In Aerospace 2015 (DASIA'15)**. [S.l.: s.n.], 2015. v. 732, p. 61.

CROS, F. et al. Dynamic software randomisation: Lessons learned from an aerospace case study. In: **Design, Automation and Test in Europe Conference and Exhibition 2017 (DATE'17)**. [S.l.]: IEEE, 2017. p. 103–108.

CUCU-GROSJEAN, L. Independence - A misunderstood property of and for probabilistic real-time systems. In: **Real-Time Systems: the past, the present and the future**. [S.l.]: CreateSpace, 2013. p. 29–37.

CUCU-GROSJEAN, L. et al. Measurement-Based Probabilistic Timing Analysis for Multi-path Programs. In: **Euromicro Conference on Real-Time Systems 2012 (ECRTS'12)**. [S.l.]: IEEE, 2012. p. 91–101.

CULLMANN, C. et al. Predictability Considerations in the Design of Multi-Core Embedded Systems. In: **Embedded Real Time Software and Systems Conference 2010 (ERTS'10)**. [S.l.: s.n.], 2010. v. 807, p. 36–42.

DASARI, D. et al. Identifying the sources of unpredictability in COTS-based multicore systems. In: **International Symposium on Industrial Embedded Systems 2013 (SIES'13)**. [S.l.]: IEEE, 2013. p. 39–48.

DAVIS, R. I.; BURNS, A.; GRIFFIN, D. On the Meaning of pWCET Distributions and their use in Schedulability Analysis. In: **International Real-Time Scheduling Open Problems Seminar 2017 (RTSOPS'17)**. [S.l.: s.n.], 2017. p. 4.

DAVIS, R. I. et al. PROXIMA: A Probabilistic Approach to the Timing Behaviour of Mixed-Criticality Systems. **Ada User Journal (AUJ)**, Ada-Europe, p. 118–122, 2014.

EDGAR, S. **Estimation of Worst-Case Execution Time Using Statistical Analysis**. 193 p. Tese (Doutorado) — University of York, 2002.

EDGAR, S.; BURNS, A. Statistical analysis of WCET for scheduling. In: **Real-Time Systems Symposium 2001 (RTSS'01)**. [S.l.]: IEEE, 2001. p. 215–224.

EMBRECHTS, P.; KLÜPPELBERG, C.; MIKOSCH, T. **Modelling Extremal Events: for Insurance and Finance**. 1. ed. [S.l.]: Springer, 2013. 648 p. (Stochastic Modelling and Applied Probability, v. 33).

FARANDA, D. et al. Numerical Convergence of the Block-Maxima Approach to the Generalized Extreme Value Distribution. **Journal of Statistical Physics**, Springer, v. 145, p. 1156–1180, 2011.

FELLER, W. **An Introduction to Probability Theory and its Applications**. 3. ed. [S.l.]: Wiley, 1968. 527 p.

FERNANDEZ, G.; CAZORLA, F. J.; ABELLA, J. Consumer Electronics Processors for Critical Real-Time Systems: a (Failed) Practical Experience. In: **European Congress on Embedded Real Time Software and Systems 2018 (ERTS'18)**. [S.l.]: HAL, 2018. p. 5.

FERNANDEZ, M. et al. Probabilistic timing analysis on time-randomized platforms for the space domain. In: **Design, Automation and Test in Europe Conference and Exhibition 2017 (DATE'17)**. [S.l.]: IEEE, 2017. p. 738–739.

FISHER, R. A.; TIPPETT, L. H. C. Limiting forms of the frequency distribution of the largest or smallest member of a sample. **Mathematical Proceedings of the Cambridge Philosophical Society**, Cambridge University Press, v. 24, p. 180–190, 1928.

GALTON, F. Dice for Statistical Experiments. **Nature**, Nature Publishing Group, p. 2, 1890.

GEBHARD, G. Timing Anomalies Reloaded. In: **International Workshop on Worst-Case Execution Time Analysis 2010 (WCET'10)**. [S.l.]: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010. v. 15, p. 1–10.

GIL, S. J. et al. Open Challenges for Probabilistic Measurement-Based Worst-Case Execution Time. **IEEE Embedded Systems Letters (ESL)**, IEEE, PP, p. 4, 2017.

GILLELAND, E.; KATZ, R. W. extRemes 2.0: An Extreme Value Analysis Package in R. **Journal of Statistical Software**, v. 72, p. 39, 2016.

GILLELAND, E.; RIBATET, M.; STEPHENSON, A. G. A software review for extreme value analysis. **Extremes**, Springer, v. 16, p. 103–119, 2013.

GONZALEZ, A.; LATORRE, F.; MAGKLIS, G. **Processor Microarchitecture: An Implementation Perspective**. [S.l.]: Morgan & Claypool, 2010. 116 p. (Synthesis Lectures on Computer Architecture).

GRIFFIN, D.; BURNS, A. Realism in Statistical Analysis of Worst Case Execution Times. In: **International Workshop on Worst-Case Execution Time Analysis 2010 (WCET'10)**. [S.l.]: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010. v. 15, p. 44–53.

GUET, F.; SANTINELLI, L.; MORIO, J. On the Reliability of the Probabilistic Worst-Case Execution Time Estimates. In: **European Congress on Embedded Real Time Software and Systems 2016 (ERTS'16)**. [S.l.: s.n.], 2016. p. 10.

GUET, F.; SANTINELLI, L.; MORIO, J. On the Representativity of Execution Time Measurements: Studying Dependence and Multi-Mode Tasks. In: **International Workshop on Worst-Case Execution Time Analysis 2017 (WCET'17)**. [S.l.]: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. v. 57, p. 1–13.

GUMBEL, E. J. **Statistics of Extremes**. [S.l.]: Dover Publications, 2012. 396 p. (Dover Books on Mathematics).

GUSTAFSSON, J. et al. The Mälardalen WCET Benchmarks: Past, Present And Future. In: **International Workshop on Worst-Case Execution Time Analysis 2010 (WCET'10)**. [S.l.]: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010. v. 15, p. 136–146.

HAAN, L. de; FERREIRA, A. **Extreme Value Theory - An Introduction**. [S.l.]: Springer, 2006. 421 p. (Springer Series in Operations Research and Financial Engineering).

HANSEN, J.; HISSAM, S.; MORENO, G. A. Statistical-Based WCET Estimation and Validation. In: **International Workshop on Worst-Case Execution Time Analysis 2009 (WCET'09)**. [S.l.]: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2009. v. 10, p. 1–11.

HENNESSY, J. L.; PATTERSON, D. A. **Computer Architecture: A Quantitative Approach**. 5. ed. [S.l.]: Morgan Kaufmann, 2012. 1357 p.

HERNANDEZ, C. et al. Towards Making a LEON3 Multicore Compatible with Probabilistic Timing Analysis. In: **Data Systems In Aerospace 2015 (DASIA'15)**. [S.l.: s.n.], 2015. v. 732, p. 60.

HERNANDEZ, C. et al. Random Modulo: A New Processor Cache Design for Real-time Critical Systems. In: **Design Automation Conference 2016 (DAC'16)**. [S.l.]: ACM, 2016. p. 29:1–29:6.

HOSKING, J. R. M. L-Moments: Analysis and Estimation of Distributions Using Linear Combinations of Order Statistics. **Journal of the Royal Statistical Society. Series B (Methodological)**, [Royal Statistical Society, Wiley], v. 52, p. 105–124, 1990.

INTEL. **Intel® Digital Random Number Generator (DRNG) - Software Implementation Guide**. [S.l.], 2012. 35 p.

INTEL. **Intel® 64 and IA-32 Architectures Software Developer's Manual - Combined Volumes: 1, 2A, 2B, 2C, 2D, 3A, 3B, 3C and 3D**. [S.l.], 2016. 4670 p.

JACOB, B.; NG, S. W.; WANG, D. T. **Memory systems: Cache, DRAM, Disk**. [S.l.]: Morgan Kaufmann, 2008. 998 p.

JALLE, J. et al. Bus designs for time-probabilistic multicore processors. In: **Design, Automation and Test in Europe Conference and Exhibition 2014 (DATE'14)**. [S.l.]: IEEE, 2014. p. 1–6.

KIM, N. et al. Attacking the One-Out-Of-m Multicore Problem by Combining Hardware Management with Mixed-Criticality Provisioning. In: **Real-Time and Embedded Technology and Applications Symposium 2016 (RTAS'16)**. [S.l.]: IEEE, 2016. p. 15.

KINNAN, L. M. Use of multicore processors in avionics systems and its potential impact on implementation and certification. In: **Digital Avionics Systems Conference 2009 (DASC'09)**. [S.l.]: IEEE, 2009. p. 1.E.4–1–1.E.4–6.

KNUTH, D. E. **The Art of Computer Programming**. 3rd. ed. [S.l.]: Addison-Wesley, 1997. 782 p.

KOETER, J. **What's an LFSR?** [S.l.], 1996. 12 p.

KOSMIDIS, L. et al. A Cache Design for Probabilistically Analysable Real-time Systems. In: **Design, Automation and Test in Europe Conference and Exhibition 2013 (DATE'13)**. [S.l.]: IEEE, 2013. p. 513–518.

KOSMIDIS, L. et al. Multi-level Unified Caches for Probabilistically Time Analysable Real-Time Systems. In: **Real-Time Systems Symposium 2013 (RTSS'13)**. [S.l.]: IEEE, 2013. p. 360–371.

KOSMIDIS, L. et al. Efficient Cache Designs for Probabilistically Analysable Real-Time Systems. **IEEE Transactions on Computers (TC)**, IEEE, v. 63, p. 2998–3011, 2014.

KOSMIDIS, L. et al. PUB: Path Upper-Bounding for Measurement-Based Probabilistic Timing Analysis. In: **Euromicro Conference on Real-Time Systems 2014 (ECRTS'14)**. [S.l.]: IEEE, 2014. p. 276–287.

KOSMIDIS, L. et al. Probabilistic Timing Analysis on Conventional Cache Designs. In: **Design, Automation and Test in Europe Conference and Exhibition 2013 (DATE'13)**. [S.l.]: EDA Consortium, 2013. p. 603–606.

KOSMIDIS, L. et al. Achieving timing composability with measurement-based probabilistic timing analysis. In: **International Symposium on Object / Component / Service-Oriented Real-Time Distributed Computing 2013 (ISORC'13)**. [S.l.]: IEEE, 2013. p. 1–8.

KOSMIDIS, L. et al. Measurement-Based Probabilistic Timing Analysis and Its Impact on Processor Architecture. In: **Euromicro Conference on Digital System Design 2014 (DSD'14)**. [S.l.]: IEEE, 2014. p. 401–410.

KOSMIDIS, L. et al. Fitting processor architectures for measurement-based probabilistic timing analysis. **Microprocessors and Microsystems (MICPRO)**, Elsevier, v. 47B, p. 287–302, 2016.

KOSMIDIS, L. et al. Applying Measurement-Based Probabilistic Timing Analysis to Buffer Resources. In: **International Workshop**

on Worst-Case Execution Time Analysis 2013 (WCET'13). [S.l.]: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2013. v. 30, p. 97–108.

KOTABA, O. et al. Multicore In Real-Time Systems - Temporal Isolation Challenges Due To Shared Resources. In: **Workshop on Industry-Driven Approaches for Cost-Effective Certification of Safety-Critical Mixed-Criticality Systems 2013 (WICERT 2013)**. [S.l.: s.n.], 2013. p. 6.

LAHIRI, K.; RAGHUNATHAN, A.; LAKSHMINARAYANA, G. LOTTERYBUS: A new high-performance communication architecture for system-on-chip designs. In: **Design Automation Conference 2001 (DAC'01)**. [S.l.]: IEEE, 2001. p. 15–20.

LAIO, F. Cramer-von Mises and Anderson-Darling goodness of fit tests for extreme value distributions with unknown parameters. **Water Resources Research**, John Wiley & Sons, v. 40, p. 10, 2004.

LAW, S.; BATE, I. Achieving Appropriate Test Coverage for Reliable Measurement-Based Timing Analysis. In: **Euromicro Conference on Real-Time Systems 2016 (ECRTS'16)**. [S.l.]: IEEE, 2016. p. 189–199.

LEADBETTER, M. R.; LINDGREN, G.; ROOTZÉN, H. **Extremes and Related Properties of Random Sequences and Processes**. 1. ed. [S.l.]: Springer, 1983. 336 p. (Springer Series in Statistics).

LESAGE, B. et al. A Framework for the Evaluation of Measurement-based Timing Analyses. In: **International Conference on Real-Time and Network Systems 2015 (RTNS'15)**. [S.l.]: ACM, 2015. p. 35–44.

LESAGE, B.; LAW, S.; BATE, I. TACO: An industrial case study of Test Automation for COverage. In: **International Conference on Real-Time Networks and Systems 2018 (RTNS'18)**. [S.l.]: ACM, 2018. p. 114–124.

LIMA, G.; DIAS, D.; BARROS, E. Extreme Value Theory for Estimating Task Execution Time Bounds: A Careful Look. In: **Euromicro Conference on Real-Time Systems 2016 (ECRTS'16)**. [S.l.]: IEEE, 2016. p. 200–211.

LIU, I.; REINEKE, J.; LEE, E. A. A PRET architecture supporting concurrent programs with composable timing properties. In: **Asilomar**

Conference on Signals, Systems and Computers 2010 (ASILOMAR'10). [S.l.]: IEEE, 2010. p. 2111–2115.

LIU, J. W.-S. **Real-Time systems**. 1. ed. [S.l.]: Prentice Hall, 2000. 409 p.

LIU, M.; BEHNAM, M.; NOLTE, T. Applying the peak over thresholds method on worst-case response time analysis of complex real-time systems. In: **International Conference on Embedded and Real-Time Computing Systems and Applications 2013 (RTCSA'13)**. [S.l.]: IEEE, 2013. p. 22–31.

LJUNG, G. M.; BOX, G. E. P. On a Measure of Lack of Fit in Time Series Models. **Biometrika**, Biometrika Trust, v. 65, p. 297–303, 1978.

LU, Y. et al. A Trace-Based Statistical Worst-Case Execution Time Analysis of Component-Based Real-Time Embedded Systems. In: **International Conference on Emerging Technologies and Factory Automation 2011 (ETFA'11)**. [S.l.]: IEEE, 2011. p. 1–4.

LU, Y. et al. A Statistical Response-Time Analysis of Real-Time Embedded Systems. In: **Real-Time Systems Symposium 2012 (RTSS'12)**. [S.l.]: IEEE, 2012. p. 351–362.

LUNDQVIST, T.; STENSTRÖM, P. Timing anomalies in dynamically scheduled microprocessors. In: **Real-Time Systems Symposium 1999 (RTSS'99)**. [S.l.]: IEEE, 1999. p. 12–21.

MA, S. et al. **Networks-on-Chip: From Implementations to Programming Paradigms**. 1. ed. [S.l.]: Morgan Kaufmann, 2014. 361 p.

MARKOVIC, N. **Hardware Thread Scheduling Algorithms for Single-ISA Asymmetric CMPs**. 124 p. Tese (Doutorado) — Universitat Politècnica de Catalunya, 2015.

MARSAGLIA, G.; TSANG, W. W. Some Difficult-to-pass Tests of Randomness. **Journal of Statistical Software**, Foundation for Open Access Statistics, v. 7, p. 1–9, 2002.

MARTINS, E. S.; STEDINGER, J. R. Generalized maximum-likelihood generalized extreme-value quantile estimators for hydrologic data. **Water Resources Research**, Wiley, v. 36, p. 737–744, 2000.

MAXIM, C. et al. Reproducibility and representativity - mandatory properties for the compositionality of measurement-based WCET estimation approaches. In: **International Workshop on Compositional Theory and Technology for Real-Time Embedded System (CRTS'16)**. [S.l.: s.n.], 2016. p. 17–24.

MAYS, L. W. **Water Resources Engineering**. 2nd. ed. [S.l.]: John Wiley & Sons, 2010. 920 p.

MELANI, A.; NOULARD, E.; SANTINELLI, L. Learning from probabilities: Dependences within real-time systems. In: **International Conference on Emerging Technologies and Factory Automation 2013 (ETFA'13)**. [S.l.]: IEEE, 2013. p. 1–8.

MEZZETTI, E. et al. EPC Enacted: Integration in an Industrial Toolbox and Use against a Railway Application. In: **Real-Time and Embedded Technology and Applications Symposium 2017 (RTAS'17)**. [S.l.]: IEEE, 2017. p. 163–174.

MEZZETTI, E. et al. Randomized Caches Can Be Pretty Useful to Hard Real-Time Systems. **Leibniz Transactions on Embedded Systems (LITES)**, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, v. 2, p. 01:1–01:10, 2015.

MILUTINOVIC, S. et al. Software Time Reliability in the Presence of Cache Memories. In: **International Conference on Reliable Software Technologies 2017 (Ada-Europe 2017)**. [S.l.]: Springer, 2017. p. 233–249.

MILUTINOVIC, S.; ABELLA, J.; CAZORLA, F. J. Modelling Probabilistic Cache Representativeness in the Presence of Arbitrary Access Patterns. In: **International Symposium on Real-Time Distributed Computing 2016 (ISORC'16)**. [S.l.]: IEEE, 2016. p. 142–149.

MILUTINOVIC, S.; ABELLA, J.; CAZORLA, F. J. On the assessment of probabilistic WCET estimates reliability for arbitrary programs. **EURASIP Journal on Embedded Systems**, Springer, v. 2017, p. 28, 2017.

MILUTINOVIC, S. et al. Measurement-based Cache Representativeness on Multipath Programs. In: **Design Automation Conference 2018 (DAC'18)**. [S.l.]: ACM, 2018. p. 123:1–123:6.

MILUTINOVIC, S. et al. On uses of extreme value theory fit for industrial-quality WCET analysis. In: **International Symposium on Industrial Embedded Systems 2017 (SIES'17)**. [S.l.]: IEEE, 2017. p. 1–6.

MISES, R. V. La distribution de la plus grande de n valeurs. **Rev. math. Union interbalcanique**, v. 1, p. 141–160, 1936.

MURALI, S. **Designing Reliable and Efficient Networks on Chips**. [S.l.]: Springer, 2009. 200 p. (Lecture Notes in Electrical Engineering, v. 34).

MURDOCCA, M.; HEURING, V. P. **Principles of Computer Architecture**. [S.l.]: Prentice Hall, 1999. 654 p.

NÉLIS, V.; YOMSI, P. M.; PINHO, L. M. The variability of application execution times on a multi-core platform. In: **International Workshop on Worst-Case Execution Time Analysis 2016 (WCET'16)**. [S.l.]: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. p. 12.

NÉLIS, V. et al. The Challenge of Time-Predictability in Modern Many-Core Architectures. In: **International Workshop on Worst-Case Execution Time Analysis 2014 (WCET'14)**. [S.l.]: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014. v. 39, p. 72.

NOWOTSCH, J.; PAULITSCH, M. Leveraging Multi-core Computing Architectures in Avionics. In: **European Dependable Computing Conference 2012 (EDCC'12)**. [S.l.]: IEEE, 2012. p. 132–143.

NULL, L.; LOBUR, J. **The Essentials of Computer Organization and Architecture**. 1. ed. [S.l.]: Jones & Bartlett Learning, 2003. 703 p.

PANIĆ, M. et al. Enabling TDMA Arbitration in the Context of MBPTA. In: **Euromicro Conference on Digital System Design 2015 (DSD'15)**. [S.l.]: IEEE, 2015. p. 462–469.

PANIĆ, M. et al. Adapting TDMA Arbitration for Measurement-Based Probabilistic Timing Analysis. **Microprocessors and Microsystems (MICPRO)**, Elsevier, v. 52, p. 188–201, 2017.

PATTERSON, D. A.; HENNESSY, J. L. **Computer Organization and Design, Fourth Edition: The Hardware/Software Interface**. 4. ed. [S.l.]: Morgan Kaufmann, 2011. 917 p. (The Morgan Kaufmann Series in Computer Architecture and Design).

PELLIZZONI, R.; CACCAMO, M. Impact of Peripheral-Processor Interference on WCET Analysis of Real-Time Embedded Systems. **IEEE Transactions on Computers (TC)**, IEEE, v. 59, p. 400–415, 2010.

PETTERS, S. M. **Worst Case Execution Time Estimation for Advanced Processor Architectures**. 158 p. Tese (Doutorado) — Technische Universität München, 2002.

PICKANDS, J. Statistical Inference Using Extreme Order Statistics. **The Annals of Statistics**, Institute of Mathematical Statistics, v. 3, p. 119–131, 1975.

PRICE, C. **MIPS IV Instruction Set: Revision 3.2**. [S.l.]: MIPS Technologies, 1995. 334 p.

QUIÑONES, E. et al. Using Randomized Caches in Probabilistic Real-Time Systems. In: **Euromicro Conference on Real-Time Systems 2009 (ECRTS'09)**. [S.l.]: IEEE, 2009. p. 129–138.

R. **R: A Language and Environment for Statistical Computing**. 2017. Disponível em: <http://www.r-project.org/>.

REINEKE, J. Randomized Caches Considered Harmful in Hard Real-Time Systems. **Leibniz Transactions on Embedded Systems (LITES)**, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, v. 1, p. 03:1–03:13, 2014.

REINEKE, J. et al. A definition and classification of timing anomalies. In: **International Workshop on Worst-Case Execution Time Analysis 2006 (WCET'06)**. [S.l.]: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2006. v. 4, p. 1–6.

REINEKE, J.; WILHELM, R. Impact of resource sharing on performance and performance prediction. In: **Design, Automation and Test in Europe Conference and Exhibition 2014 (DATE'14)**. [S.l.]: IEEE, 2014. p. 1–2.

SAIDI, S. et al. The shift to multicores in real-time and safety-critical systems. In: **International Conference on Hardware/Software Codesign and System Synthesis 2015 (CODES+ISSS'15)**. [S.l.]: IEEE, 2015. p. 220–229.

SANTINELLI, L.; GUET, F.; MORIO, J. Revising Measurement-Based Probabilistic Timing Analysis. In: **Real-Time**

and Embedded Technology and Applications Symposium 2017 (RTAS'17). [S.l.]: IEEE, 2017. p. 199–208.

SANTINELLI, L. et al. On the Sustainability of the Extreme Value Theory for WCET Estimation. In: **International Workshop on Worst-Case Execution Time Analysis 2014 (WCET'14)**. [S.l.]: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014. v. 39, p. 21–30.

SARMA, M. S. On the convergence of the Baba and Dorea random optimization methods. **Journal of Optimization Theory and Applications**, Springer, p. 337–343, 1990.

SCARROTT, C.; MACDONALD, A. A review of extreme value threshold estimation and uncertainty quantification. **REVSTAT - Statistical Journal**, Instituto Nacional de Estatística, v. 10, p. 33–60, 2012.

SCHLANSKER, M.; SHAW, R.; SIVARAMAKRISHNAN, S. **Randomization and associativity in the design of placement-insensitive caches**. [S.l.], 1993. 21 p.

SCHLIECKER, S.; ERNST, R. Real-time performance analysis of multiprocessor systems with shared memory. **ACM Transactions on Embedded Computing Systems (TECS)**, ACM, v. 10, p. 22:1–22:27, 2010.

SCHOLZ, F. W.; STEPHENS, M. A. K-Sample Anderson-Darling Tests. **Journal of the American Statistical Association**, Taylor & Francis, v. 82, p. 918–924, 1987.

SCHÖNBERG, S. Impact of PCI-bus load on applications in a PC architecture. In: **Real-Time Systems Symposium 2003 (RTSS'03)**. [S.l.]: IEEE, 2003. p. 430–439.

SHAH, H.; HUANG, K.; KNOLL, A. Timing anomalies in multi-core architectures due to the interference on the shared resources. In: **Asia and South Pacific Design Automation Conference 2014 (ASP-DAC'14)**. [S.l.]: IEEE, 2014. p. 708–713.

SHEN, J. P.; LIPASTI, M. H. **Modern Processor Design: Funds of Superscalar Processors**. [S.l.]: Waveland, 2013. 642 p.

SILVA, K. P. et al. An Empirical Study on the Adequacy of MBPTA for Tasks Executed on a Complex Computer Architecture with Linux.

In: **International Conference on Emerging Technologies and Factory Automation 2018 (ETFA'18)**. [S.l.]: IEEE, 2018. p. 321–328.

SILVA, K. P.; ARCARO, L. F.; OLIVEIRA, R. S. de. On Using GEV or Gumbel Models when Applying EVT for Probabilistic WCET Estimation. In: **Real-Time Systems Symposium 2017 (RTSS'17)**. [S.l.]: IEEE, 2017. p. 220–230.

SLIJEPCEVIC, M. **Probabilistically time-analyzable complex processor designs**. 168 p. Tese (Doutorado), 2017.

SLIJEPCEVIC, M. et al. pTNoC: Probabilistically Time-Analyzable Tree-Based NoC for Mixed-Criticality Systems. In: **Euromicro Conference on Digital System Design 2016 (DSD'16)**. [S.l.]: IEEE, 2016. p. 404–412.

SLIJEPCEVIC, M. et al. Boosting Guaranteed Performance in Wormhole NoCs with Probabilistic Timing Analysis. In: **Euromicro Conference on Digital System Design 2017 (DSD'17)**. [S.l.]: IEEE, 2017. p. 440–444.

SLIJEPCEVIC, M. et al. Design and implementation of a fair credit-based bandwidth sharing scheme for buses. In: **Design, Automation and Test in Europe Conference and Exhibition 2017 (DATE'17)**. [S.l.]: IEEE, 2017. p. 926–929.

SLIJEPCEVIC, M. et al. Time-Analysable Non-Partitioned Shared Caches for Real-Time Multicore Systems. In: **Design Automation Conference 2014 (DAC'14)**. [S.l.]: ACM, 2014. p. 198:1–198:6.

SMITH, R. L. Maximum Likelihood Estimation in a Class of Nonregular Cases. **Biometrika**, [Oxford University Press, Biometrika Trust], v. 72, p. 67–90, 1985.

STEPHENS, L. J. **Schaum's Outline of Beginning Statistics**. 2. ed. [S.l.]: McGraw-Hill, 2009. 413 p. (Schaum's Outline).

TANENBAUM, A. S.; AUSTIN, T. **Structured Computer Organization**. 6th. ed. [S.l.]: Prentice Hall, 2012. 801 p.

TKACIK, T. E. A Hardware Random Number Generator. In: KALISKI, B. S.; KOÇ, Ç. K.; PAAR, C. (Ed.). **Cryptographic Hardware and Embedded Systems 2002 (CHES'02)**. [S.l.]: Springer, 2003. p. 450–453.

TRILLA, D. et al. Resilient random modulo cache memories for probabilistically-analyzable real-time systems. In: **International Symposium on On-Line Testing and Robust System Design 2016 (IOLTS'16)**. [S.l.]: IEEE, 2016. p. 27–32.

TRILLA, D. et al. Aging Assessment and Design Enhancement of Randomized Cache Memories. **IEEE Transactions on Device and Materials Reliability**, IEEE, v. 17, p. 32–41, 2017.

TRILLA, D. et al. On the suitability of time-randomized processors for secure and reliable high-performance computing. In: **BSC Severo Ochoa Doctoral Symposium 2017**. [S.l.]: Barcelona Supercomputing Center (BSC), 2017. p. 110–113.

TRILLA, D. et al. Cache side-channel attacks and time-predictability in high-performance critical real-time systems. In: **Design Automation Conference 2018 (DAC'18)**. [S.l.]: ACM, 2018. p. 6.

TSAI, W.-C. et al. Networks on chips: structure and design methodologies. **Journal of Electrical and Computer Engineering**, Hindawi, v. 2012, p. 2, 2012.

UNGERER, T. et al. MERASA: Multicore Execution of Hard Real-Time Applications Supporting Analyzability. **IEEE Micro**, IEEE, v. 30, p. 66–75, 2010.

WARTEL, F. et al. Timing analysis of an avionics case study on complex hardware/software platforms. In: **Design, Automation and Test in Europe Conference and Exhibition 2015 (DATE'15)**. [S.l.]: IEEE, 2015. p. 397–402.

WARTEL, F. et al. Measurement-Based Probabilistic Timing Analysis: Lessons from an Integrated-Modular Avionics Case Study. In: **International Symposium on Industrial Embedded Systems 2013 (SIES'13)**. [S.l.]: IEEE, 2013. p. 241–248.

WASSERSTEIN, R. L.; LAZAR, N. A. The ASA's Statement on p-Values: Context, Process, and Purpose. **The American Statistician**, Taylor & Francis, v. 70, p. 129–133, 2016.

WILHELM, R. et al. Designing Predictable Multicore Architectures for Avionics and Automotive Systems. In: **Workshop on Reconciling Performance with Predictability 2009 (RePP'09)**. [S.l.: s.n.], 2009. p. 6.

WILHELM, R. et al. The Worst-Case Execution-Time Problem - Overview of Methods and Survey of Tools. **ACM Transactions on Embedded Computing Systems (TECS)**, ACM, v. 7, p. 36:1–36:53, 2008.

WILLMOTT, C. J.; MATSUURA, K. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. **Climate Research**, Inter-Research, v. 30, p. 79–82, 2005.

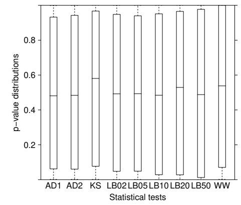
ZICCARDI, M. et al. EPC: Extended Path Coverage for Measurement-Based Probabilistic Timing Analysis. In: **Real-Time Systems Symposium 2015 (RTSS'15)**. [S.l.]: IEEE, 2015. p. 338–349.

ZIMMER, M. et al. FlexPRET: A Processor Platform for Mixed-Criticality Systems. In: **Real-Time and Embedded Technology and Applications Symposium 2014 (RTAS'14)**. [S.l.]: IEEE, 2014. p. 101–110.

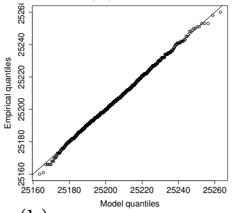
APPENDIX A – pWCET Reliability Evaluation

A.1 APPLICABILITY EVIDENCE

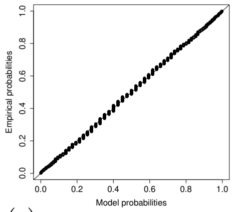
Figure 49: Applicability evidence for *bsort* on *DPCpArrr*



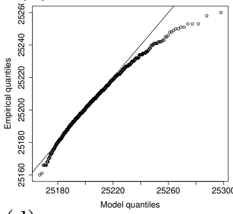
(a) Applicability statistical tests' p-values



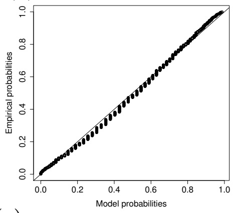
(b) GEV quantile plot



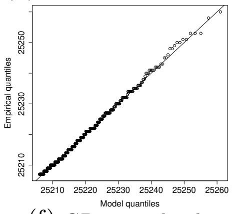
(c) GEV probability plot



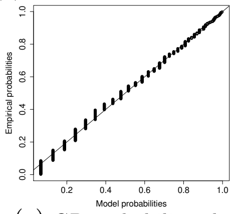
(d) Gumbel quantile plot



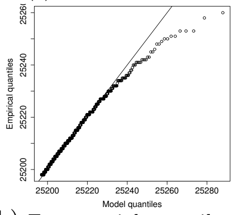
(e) Gumbel probability plot



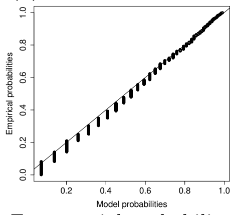
(f) GP quantile plot



(g) GP probability plot

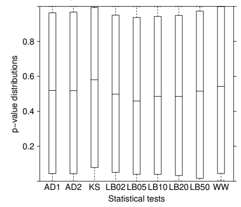


(h) Exponential quantile plot

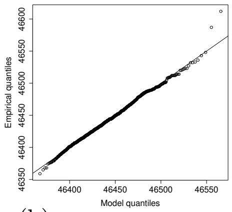


(i) Exponential probability plot

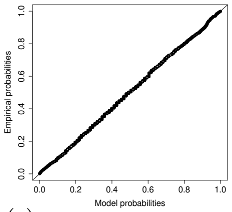
Figure 50: Applicability evidence for *bsort* on *DPArptdm*



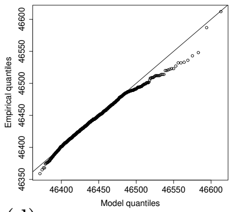
(a) Applicability statistical tests' p-values



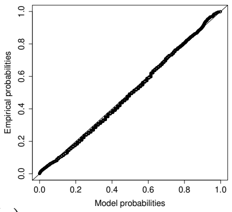
(b) GEV quantile plot



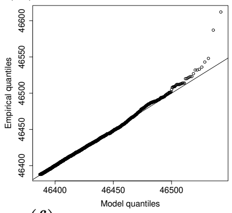
(c) GEV probability plot



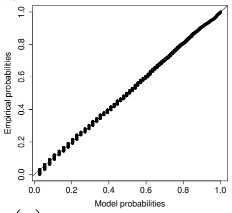
(d) Gumbel quantile plot



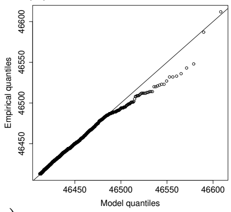
(e) Gumbel probability plot



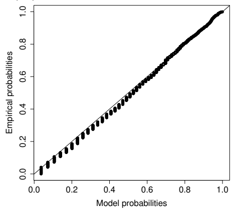
(f) GP quantile plot



(g) GP probability plot

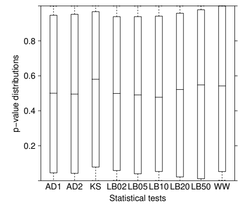


(h) Exponential quantile plot

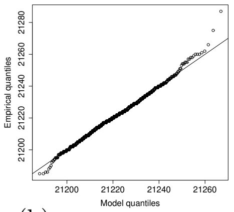


(i) Exponential probability plot

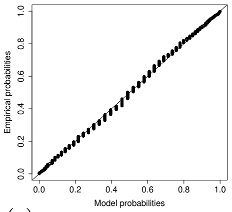
Figure 51: Applicability evidence for *insertsort* on *DPCpArrr*



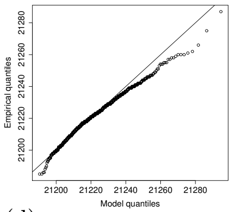
(a) Applicability statistical tests' p-values



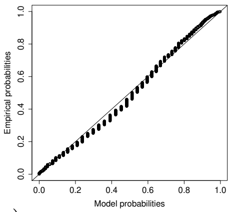
(b) GEV quantile plot



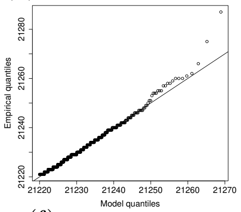
(c) GEV probability plot



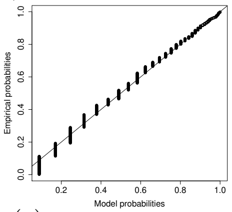
(d) Gumbel quantile plot



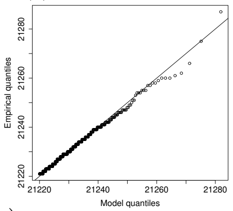
(e) Gumbel probability plot



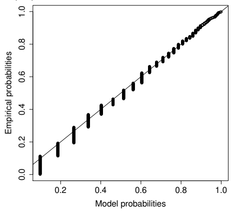
(f) GP quantile plot



(g) GP probability plot

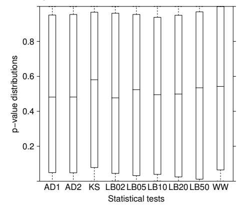


(h) Exponential quantile plot

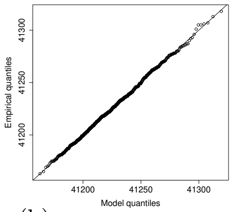


(i) Exponential probability plot

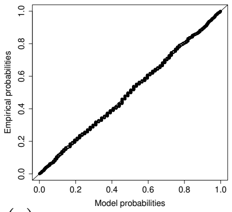
Figure 52: Applicability evidence for *insertsort* on *DPArptdm*



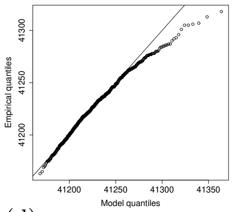
(a) Applicability statistical tests' p-values



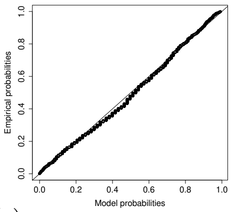
(b) GEV quantile plot



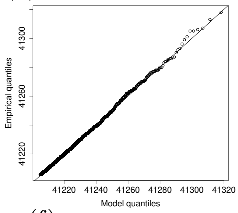
(c) GEV probability plot



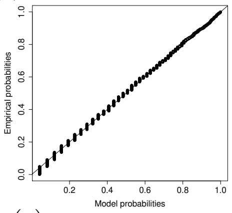
(d) Gumbel quantile plot



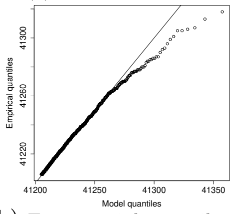
(e) Gumbel probability plot



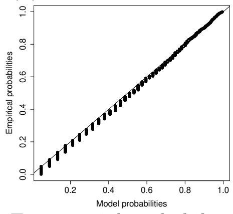
(f) GP quantile plot



(g) GP probability plot

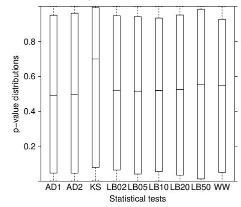


(h) Exponential quantile plot

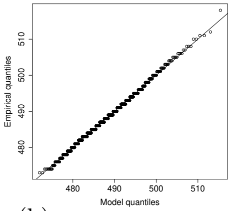


(i) Exponential probability plot

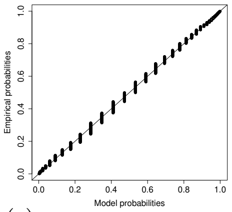
Figure 53: Applicability evidence for *bs* on *DPCpArrr*



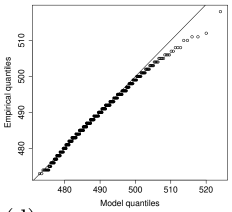
(a) Applicability statistical tests' p-values



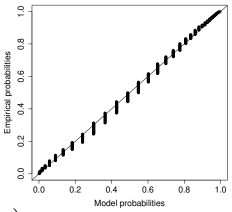
(b) GEV quantile plot



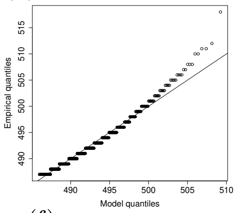
(c) GEV probability plot



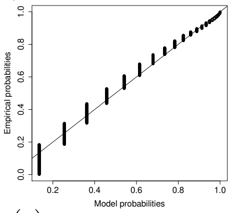
(d) Gumbel quantile plot



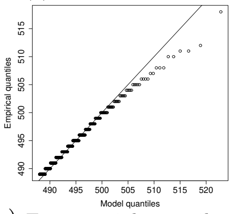
(e) Gumbel probability plot



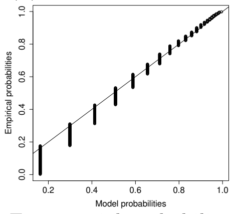
(f) GP quantile plot



(g) GP probability plot

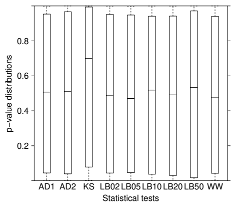


(h) Exponential quantile plot

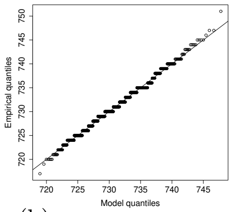


(i) Exponential probability plot

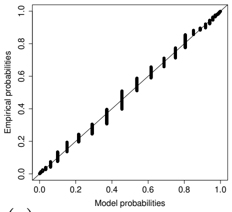
Figure 54: Applicability evidence for *bs* on *DPArptdm*



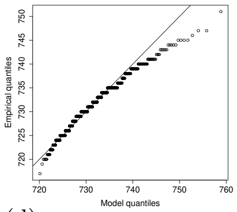
(a) Applicability statistical tests' p-values



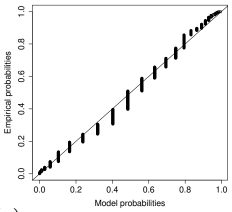
(b) GEV quantile plot



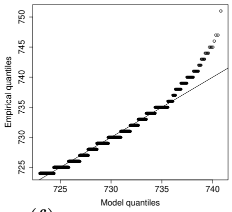
(c) GEV probability plot



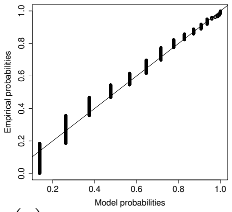
(d) Gumbel quantile plot



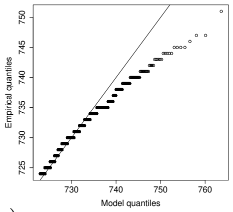
(e) Gumbel probability plot



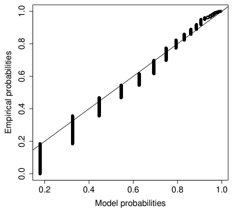
(f) GP quantile plot



(g) GP probability plot

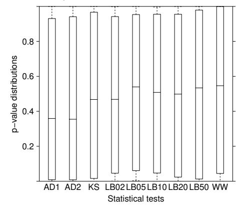


(h) Exponential quantile plot

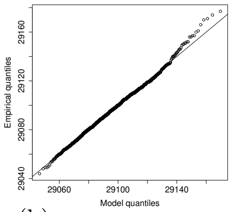


(i) Exponential probability plot

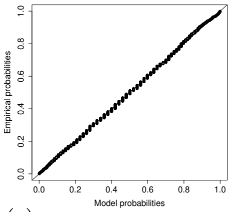
Figure 55: Applicability evidence for *expint* on *DPCpArrr*



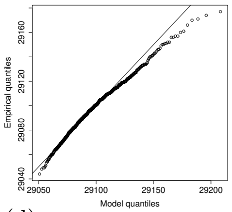
(a) Applicability statistical tests' p-values



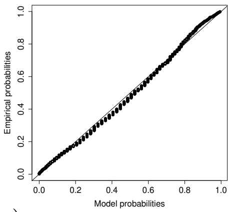
(b) GEV quantile plot



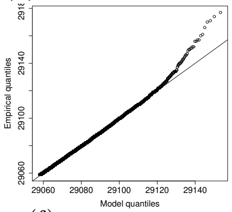
(c) GEV probability plot



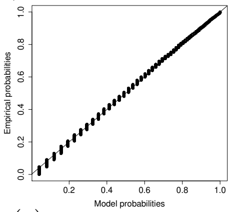
(d) Gumbel quantile plot



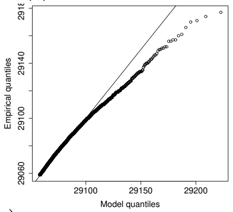
(e) Gumbel probability plot



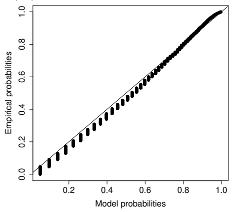
(f) GP quantile plot



(g) GP probability plot

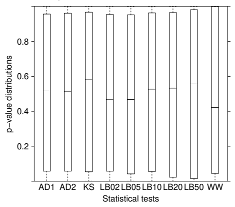


(h) Exponential quantile plot

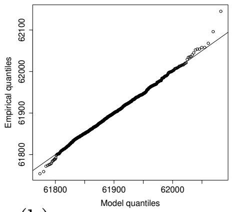


(i) Exponential probability plot

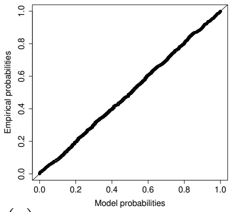
Figure 56: Applicability evidence for *expint* on *DPArptdm*



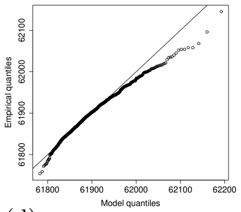
(a) Applicability statistical tests' p-values



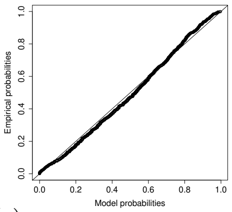
(b) GEV quantile plot



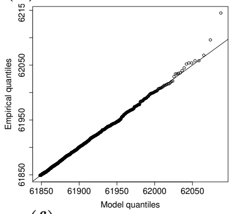
(c) GEV probability plot



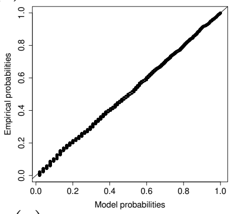
(d) Gumbel quantile plot



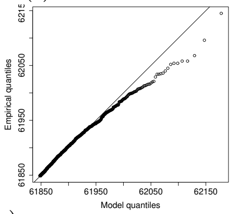
(e) Gumbel probability plot



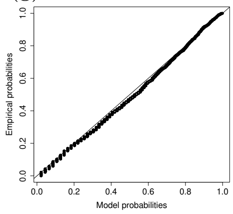
(f) GP quantile plot



(g) GP probability plot

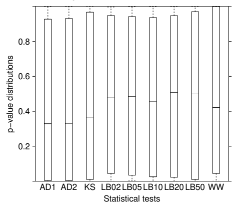


(h) Exponential quantile plot

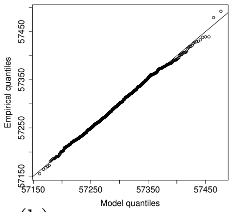


(i) Exponential probability plot

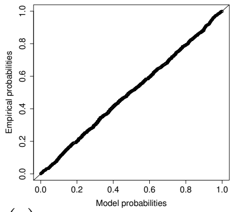
Figure 57: Applicability evidence for *fdct* on *DPCpArrr*



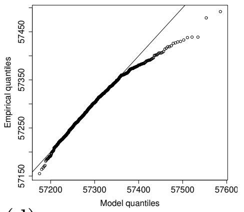
(a) Applicability statistical tests' p-values



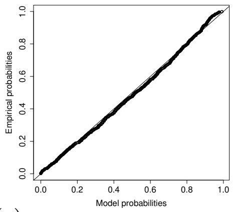
(b) GEV quantile plot



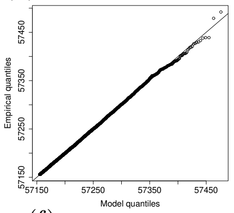
(c) GEV probability plot



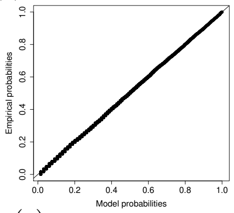
(d) Gumbel quantile plot



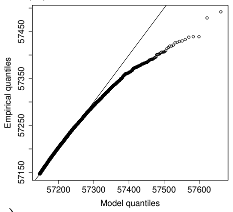
(e) Gumbel probability plot



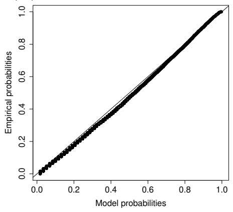
(f) GP quantile plot



(g) GP probability plot

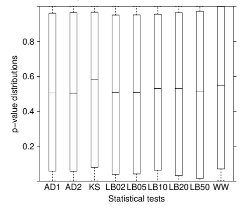


(h) Exponential quantile plot

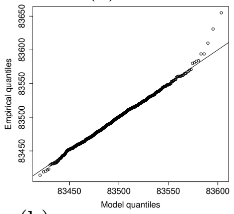


(i) Exponential probability plot

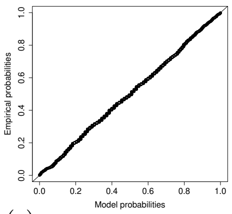
Figure 58: Applicability evidence for *fdct* on *DPArptdm*



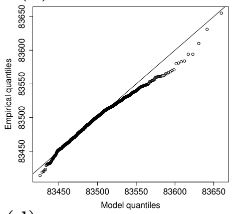
(a) Applicability statistical tests' p-values



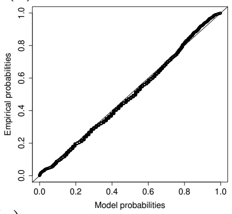
(b) GEV quantile plot



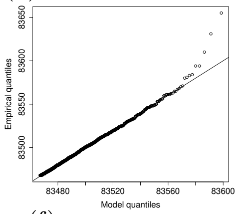
(c) GEV probability plot



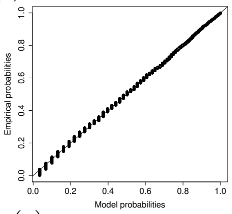
(d) Gumbel quantile plot



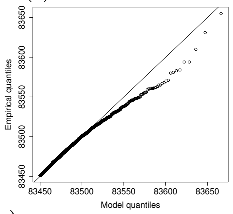
(e) Gumbel probability plot



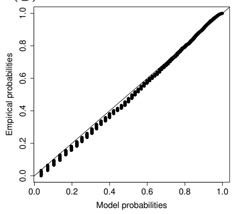
(f) GP quantile plot



(g) GP probability plot

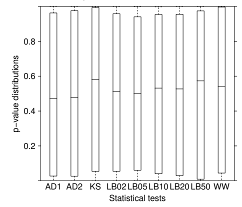


(h) Exponential quantile plot

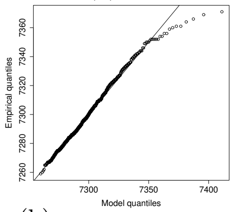


(i) Exponential probability plot

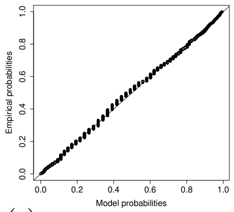
Figure 59: Applicability evidence for *crc* on *DPCpArrr*



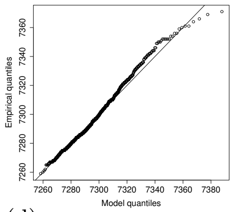
(a) Applicability statistical tests' p-values



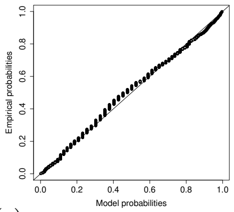
(b) GEV quantile plot



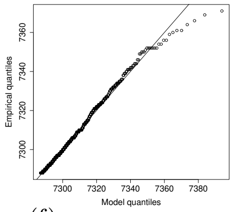
(c) GEV probability plot



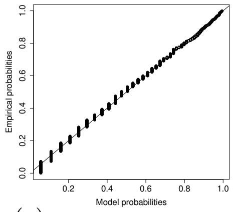
(d) Gumbel quantile plot



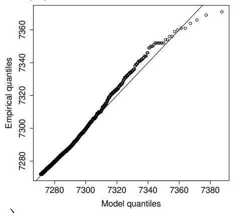
(e) Gumbel probability plot



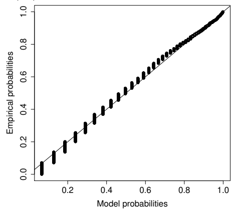
(f) GP quantile plot



(g) GP probability plot

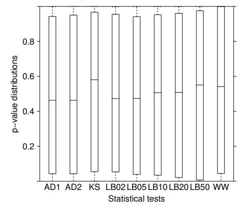


(h) Exponential quantile plot

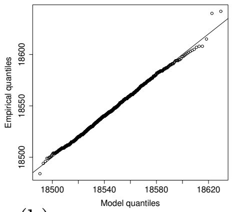


(i) Exponential probability plot

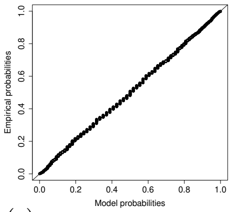
Figure 60: Applicability evidence for *crc* on *DPArptdm*



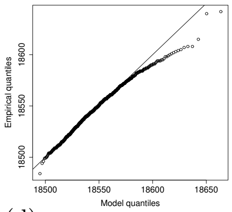
(a) Applicability statistical tests' p-values



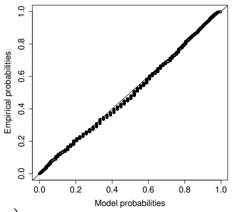
(b) GEV quantile plot



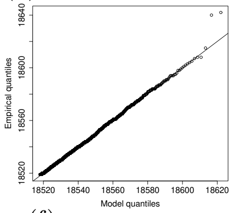
(c) GEV probability plot



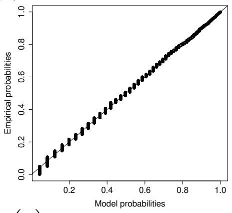
(d) Gumbel quantile plot



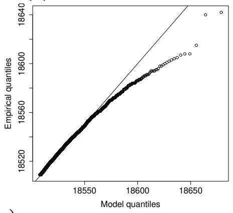
(e) Gumbel probability plot



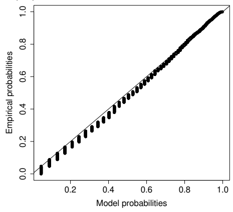
(f) GP quantile plot



(g) GP probability plot

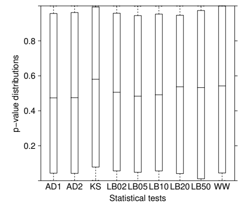


(h) Exponential quantile plot

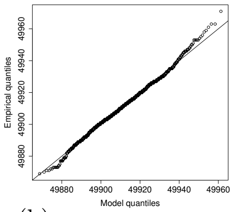


(i) Exponential probability plot

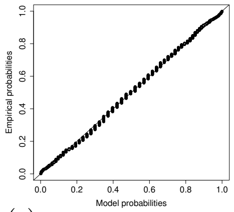
Figure 61: Applicability evidence for *matmult* on *DPCpArrr*



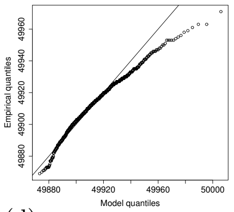
(a) Applicability statistical tests' p-values



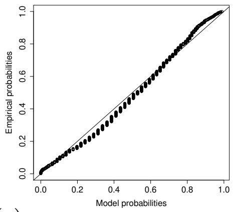
(b) GEV quantile plot



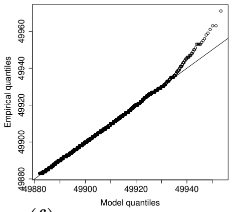
(c) GEV probability plot



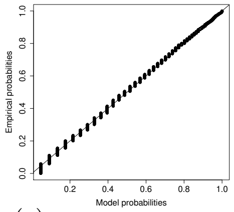
(d) Gumbel quantile plot



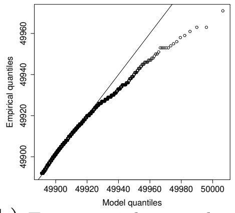
(e) Gumbel probability plot



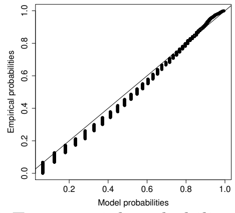
(f) GP quantile plot



(g) GP probability plot

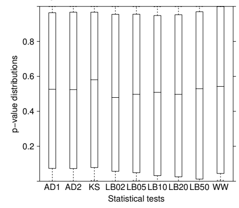


(h) Exponential quantile plot

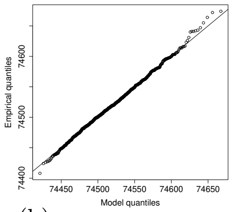


(i) Exponential probability plot

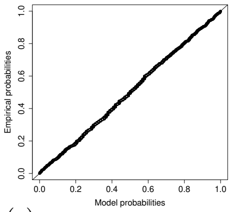
Figure 62: Applicability evidence for *matmult* on *DPArptdm*



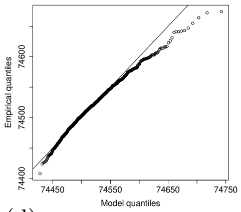
(a) Applicability statistical tests' p-values



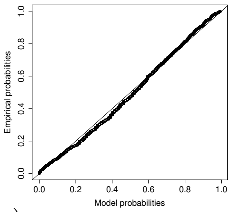
(b) GEV quantile plot



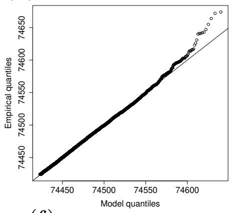
(c) GEV probability plot



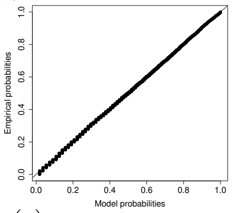
(d) Gumbel quantile plot



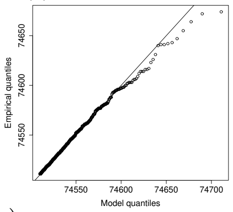
(e) Gumbel probability plot



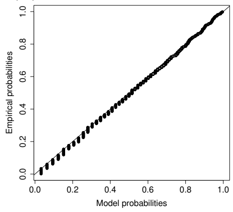
(f) GP quantile plot



(g) GP probability plot

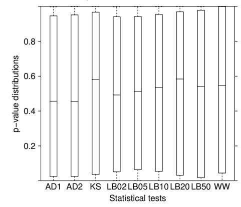


(h) Exponential quantile plot

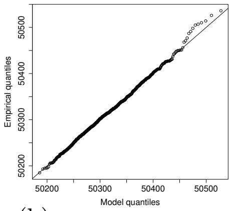


(i) Exponential probability plot

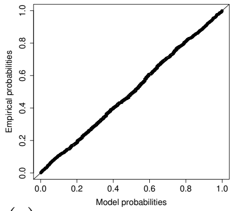
Figure 63: Applicability evidence for *fir* on *DPCpArrr*



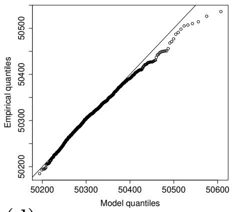
(a) Applicability statistical tests' p-values



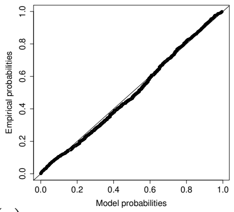
(b) GEV quantile plot



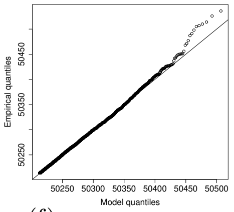
(c) GEV probability plot



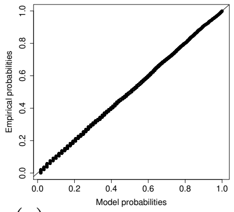
(d) Gumbel quantile plot



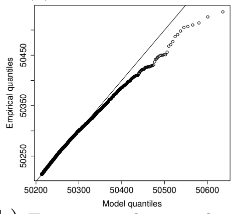
(e) Gumbel probability plot



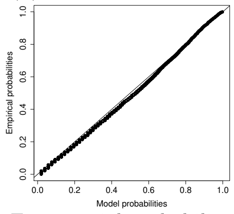
(f) GP quantile plot



(g) GP probability plot

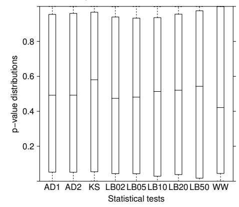


(h) Exponential quantile plot

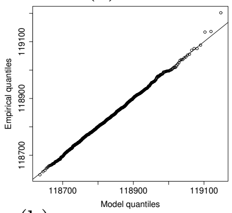


(i) Exponential probability plot

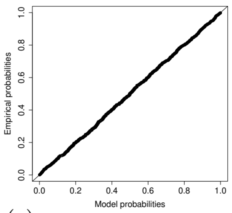
Figure 64: Applicability evidence for *fir* on *DPArptdm*



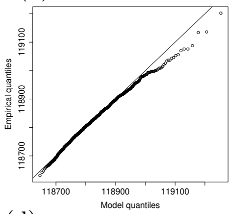
(a) Applicability statistical tests' p-values



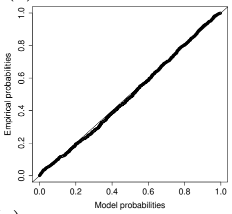
(b) GEV quantile plot



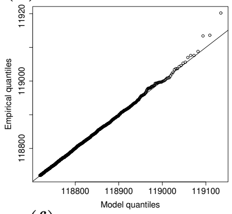
(c) GEV probability plot



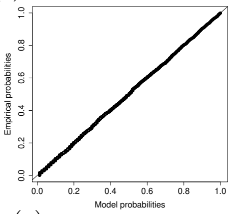
(d) Gumbel quantile plot



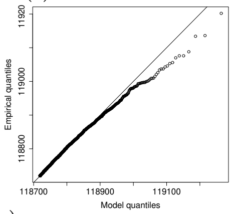
(e) Gumbel probability plot



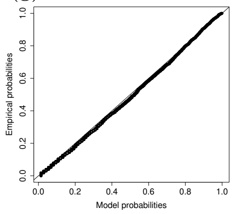
(f) GP quantile plot



(g) GP probability plot

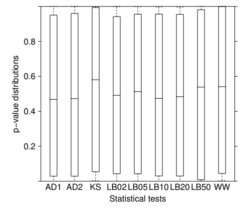


(h) Exponential quantile plot

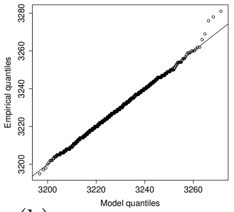


(i) Exponential probability plot

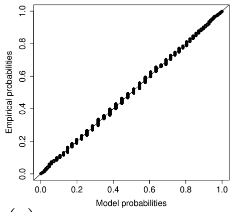
Figure 65: Applicability evidence for *fibcall* on *DPCpArrr*



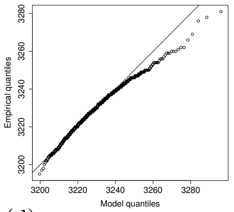
(a) Applicability statistical tests' p-values



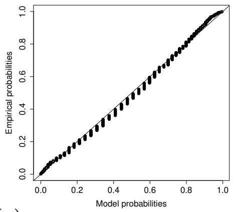
(b) GEV quantile plot



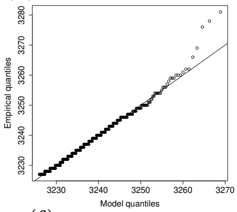
(c) GEV probability plot



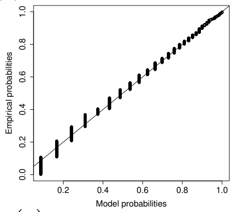
(d) Gumbel quantile plot



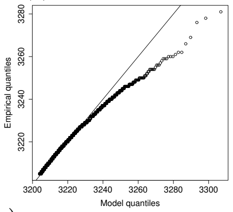
(e) Gumbel probability plot



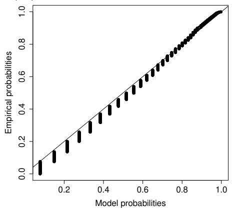
(f) GP quantile plot



(g) GP probability plot

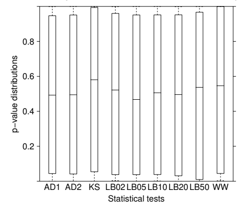


(h) Exponential quantile plot

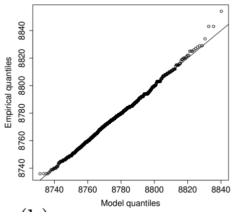


(i) Exponential probability plot

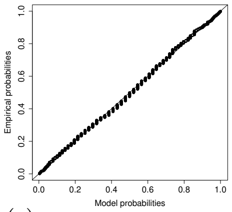
Figure 66: Applicability evidence for *fibcall* on *DPArptdm*



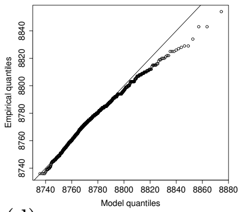
(a) Applicability statistical tests' p-values



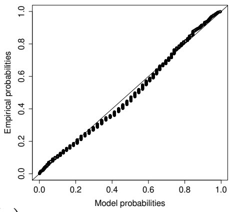
(b) GEV quantile plot



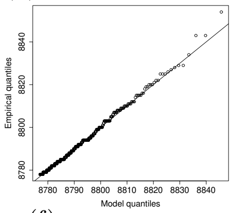
(c) GEV probability plot



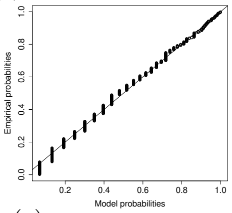
(d) Gumbel quantile plot



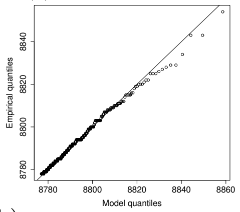
(e) Gumbel probability plot



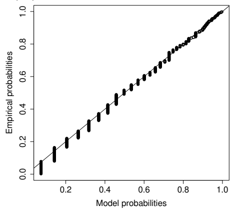
(f) GP quantile plot



(g) GP probability plot

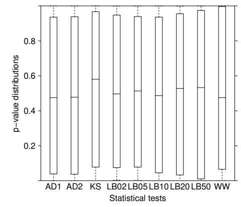


(h) Exponential quantile plot

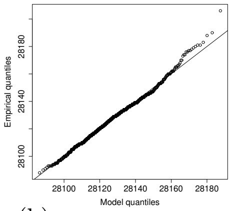


(i) Exponential probability plot

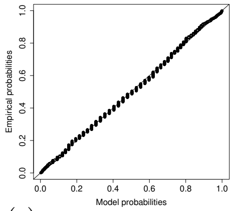
Figure 67: Applicability evidence for *cnt* on *DPCpArrr*



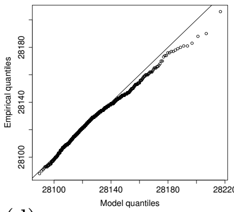
(a) Applicability statistical tests' p-values



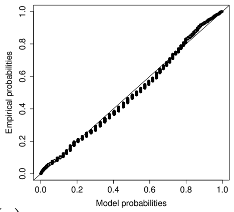
(b) GEV quantile plot



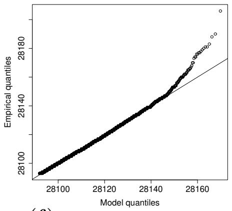
(c) GEV probability plot



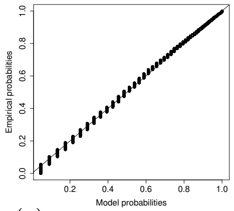
(d) Gumbel quantile plot



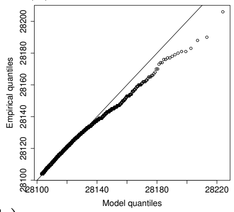
(e) Gumbel probability plot



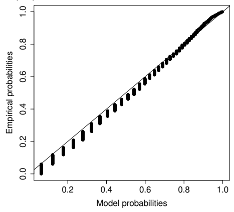
(f) GP quantile plot



(g) GP probability plot

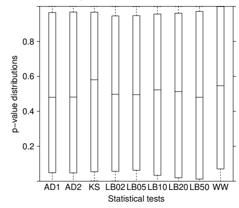


(h) Exponential quantile plot

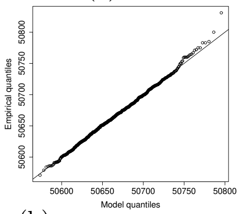


(i) Exponential probability plot

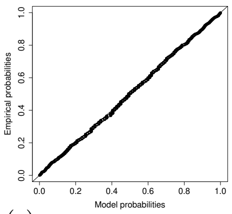
Figure 68: Applicability evidence for *cnt* on *DPArptdm*



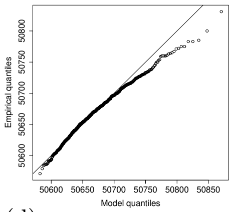
(a) Applicability statistical tests' p-values



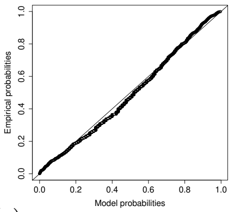
(b) GEV quantile plot



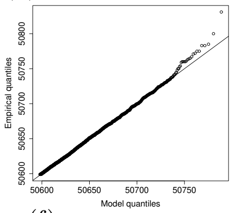
(c) GEV probability plot



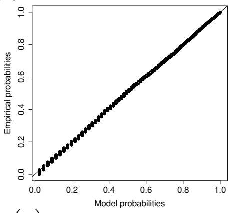
(d) Gumbel quantile plot



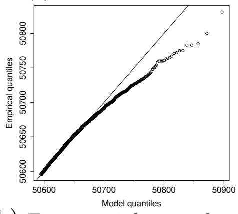
(e) Gumbel probability plot



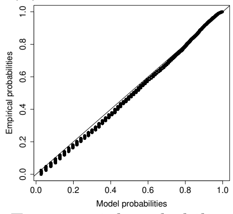
(f) GP quantile plot



(g) GP probability plot



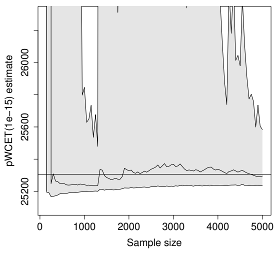
(h) Exponential quantile plot



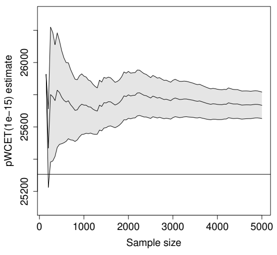
(i) Exponential probability plot

A.2 PWCET HWM RELIABILITY

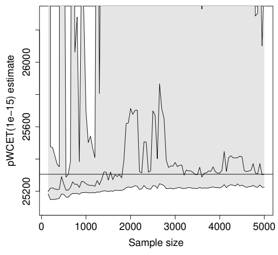
Figure 69: pWCET HWM reliability for *bsort* on *DPCpArrr*



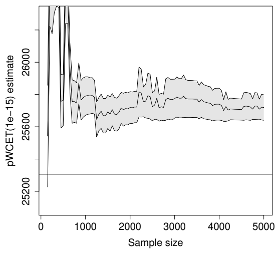
(a) GEV



(b) Gumbel

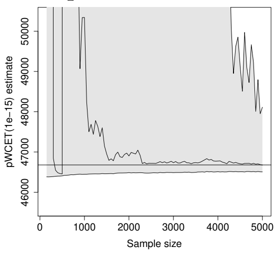


(c) GP

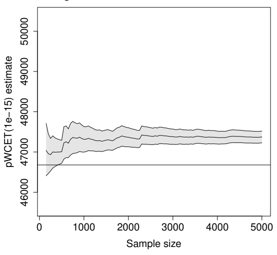


(d) Exponential

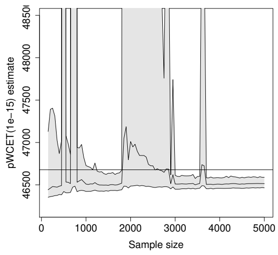
Figure 70: pWCET HWM reliability for *bsort* on *DPArptdm*



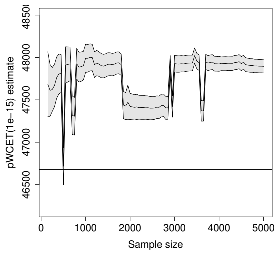
(a) GEV



(b) Gumbel

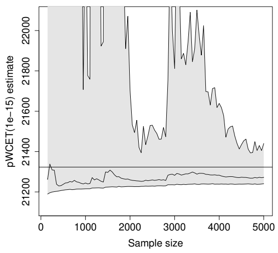


(c) GP

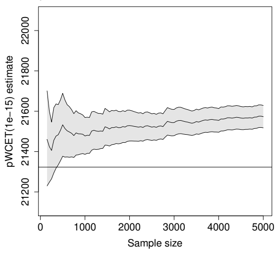


(d) Exponential

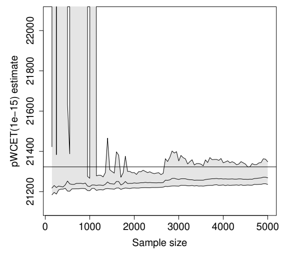
Figure 71: pWCET HWM reliability for *insertsort* on *DPCpArrr*



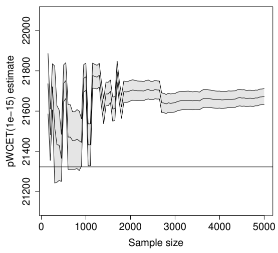
(a) GEV



(b) Gumbel

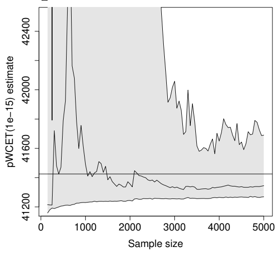


(c) GP

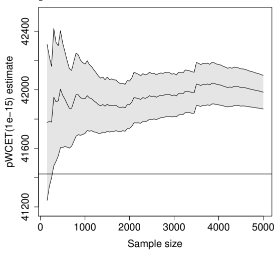


(d) Exponential

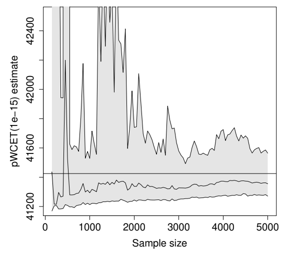
Figure 72: pWCET HWM reliability for *insertsort* on *DPArptdm*



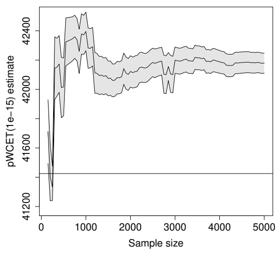
(a) GEV



(b) Gumbel

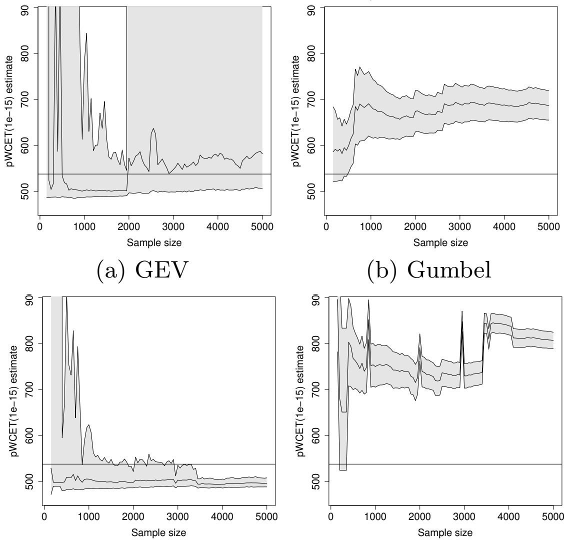


(c) GP



(d) Exponential

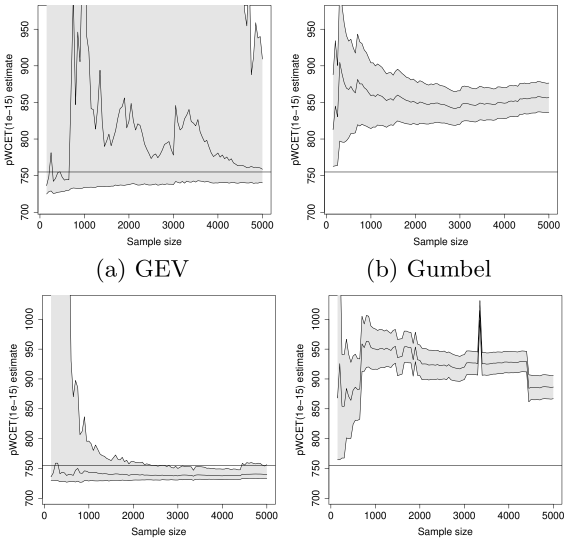
Figure 73: pWCET HWM reliability for *bs* on *DPCpArrr*



(a) GEV (b) Gumbel

(c) GP (d) Exponential

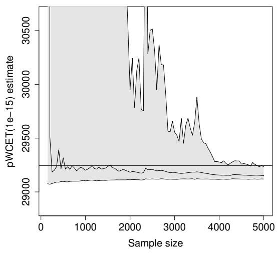
Figure 74: pWCET HWM reliability for *bs* on *DPArptdm*



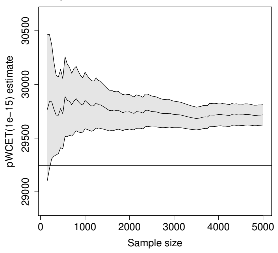
(a) GEV (b) Gumbel

(c) GP (d) Exponential

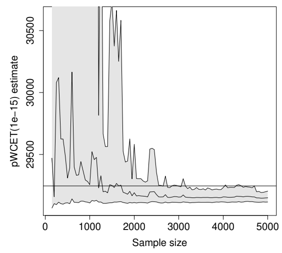
Figure 75: pWCET HWM reliability for *expint* on *DPCpArrr*



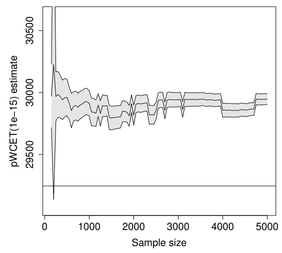
(a) GEV



(b) Gumbel

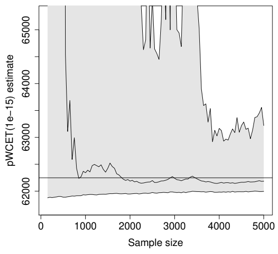


(c) GP

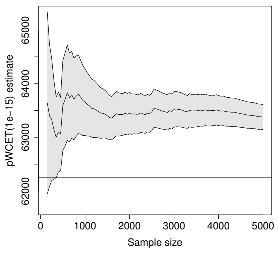


(d) Exponential

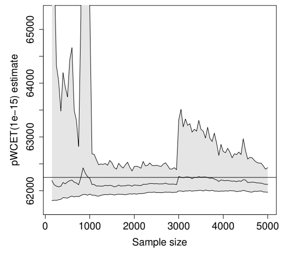
Figure 76: pWCET HWM reliability for *expint* on *DPArptdm*



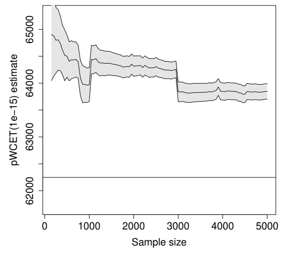
(a) GEV



(b) Gumbel

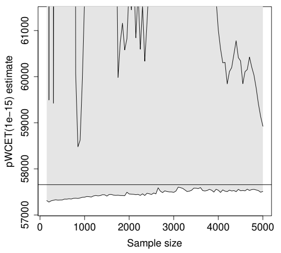


(c) GP

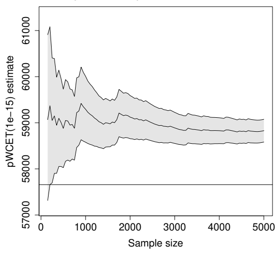


(d) Exponential

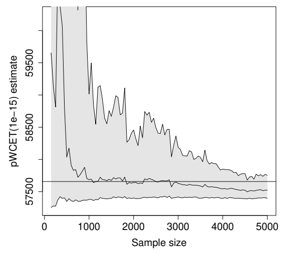
Figure 77: pWCET HWM reliability for *fdct* on *DPCpArrr*



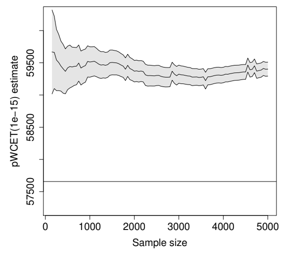
(a) GEV



(b) Gumbel

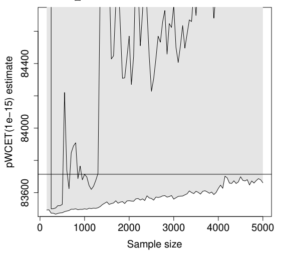


(c) GP

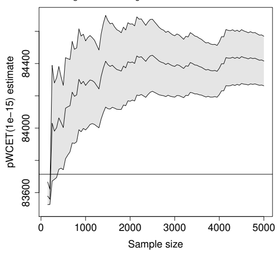


(d) Exponential

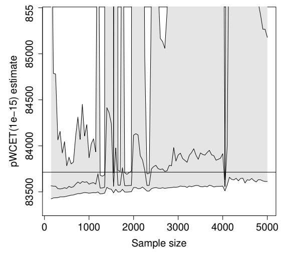
Figure 78: pWCET HWM reliability for *fdct* on *DPArptdm*



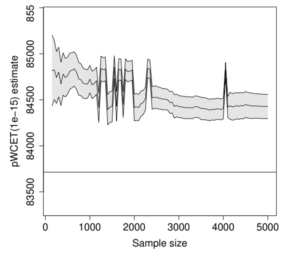
(a) GEV



(b) Gumbel

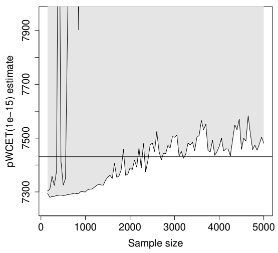


(c) GP

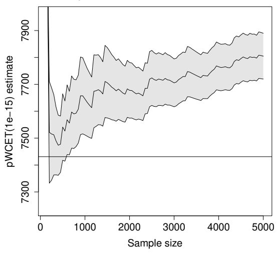


(d) Exponential

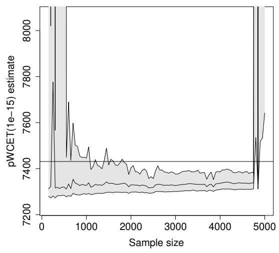
Figure 79: pWCET HWM reliability for *crc* on *DPCpArrr*



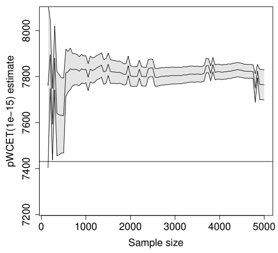
(a) GEV



(b) Gumbel

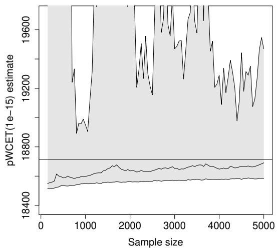


(c) GP

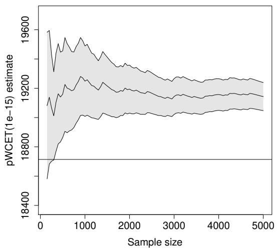


(d) Exponential

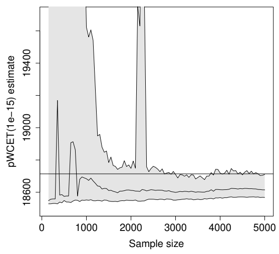
Figure 80: pWCET HWM reliability for *crc* on *DPArptdm*



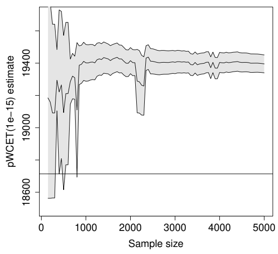
(a) GEV



(b) Gumbel

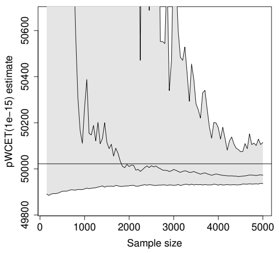


(c) GP

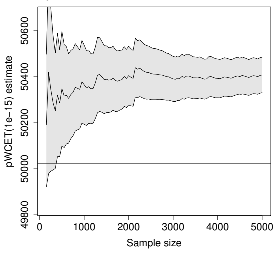


(d) Exponential

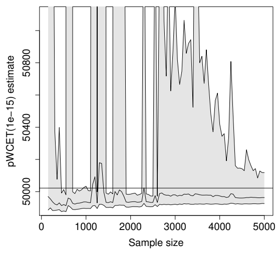
Figure 81: pWCET HWM reliability for *matmult* on *DPCpArrr*



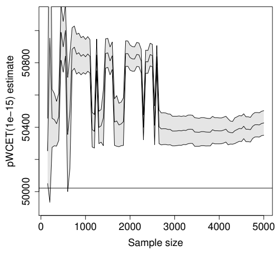
(a) GEV



(b) Gumbel

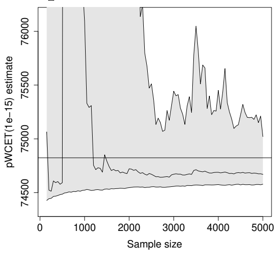


(c) GP

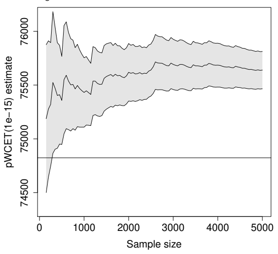


(d) Exponential

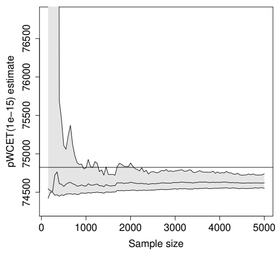
Figure 82: pWCET HWM reliability for *matmult* on *DPArptdm*



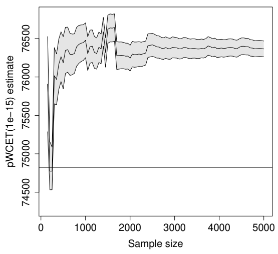
(a) GEV



(b) Gumbel

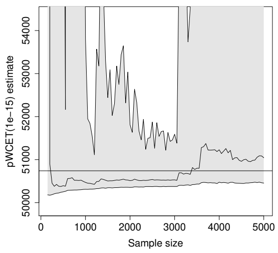


(c) GP

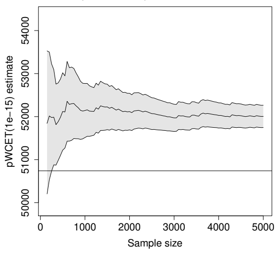


(d) Exponential

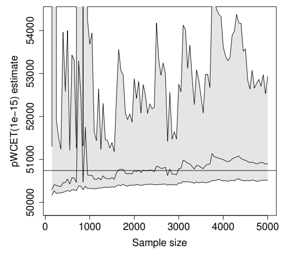
Figure 83: pWCET HWM reliability for *fir* on *DPCpArrr*



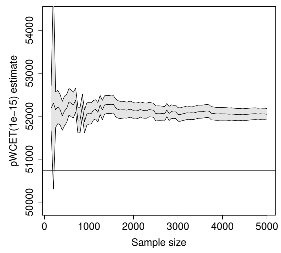
(a) GEV



(b) Gumbel

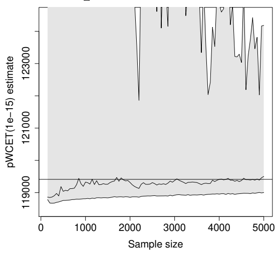


(c) GP

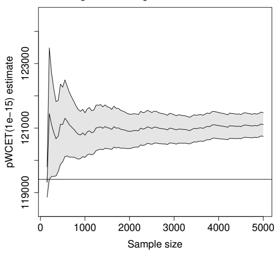


(d) Exponential

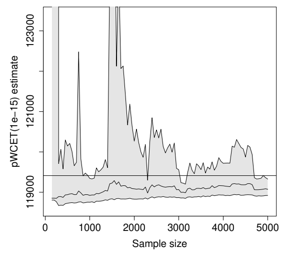
Figure 84: pWCET HWM reliability for *fir* on *DPArptdm*



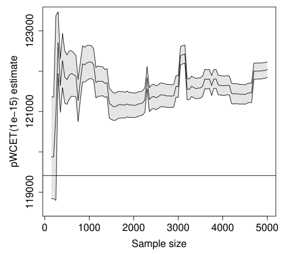
(a) GEV



(b) Gumbel

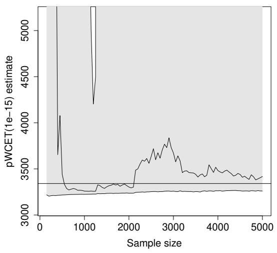


(c) GP

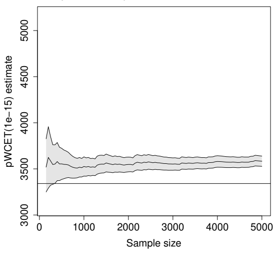


(d) Exponential

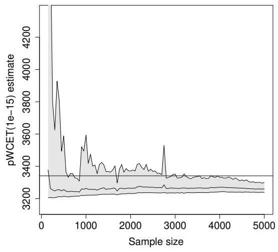
Figure 85: pWCET HWM reliability for *fibcall* on *DPCpArrr*



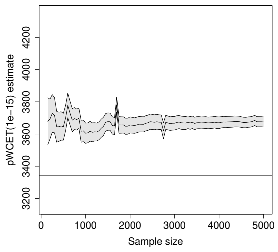
(a) GEV



(b) Gumbel

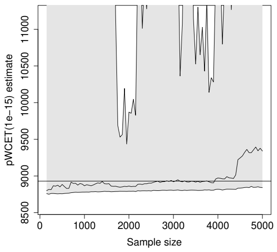


(c) GP

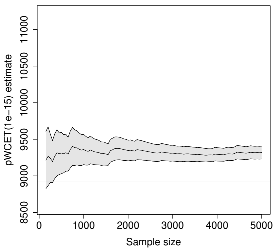


(d) Exponential

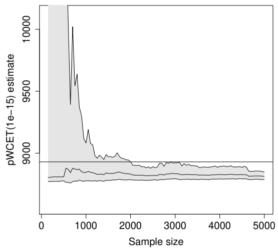
Figure 86: pWCET HWM reliability for *fibcall* on *DPArptdm*



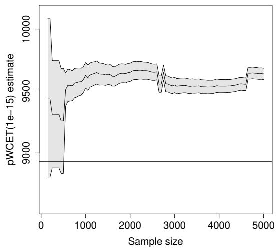
(a) GEV



(b) Gumbel

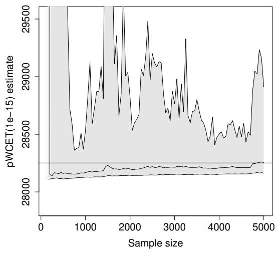


(c) GP

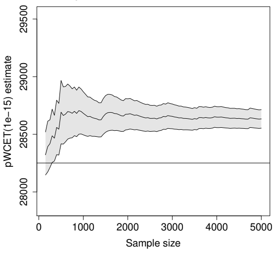


(d) Exponential

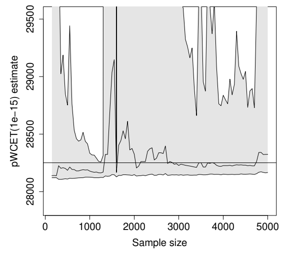
Figure 87: pWCET HWM reliability for *cnt* on *DPCpArrr*



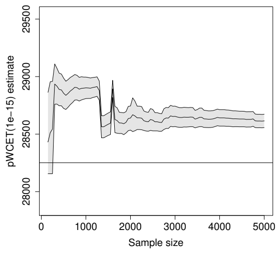
(a) GEV



(b) Gumbel

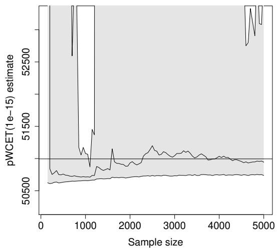


(c) GP

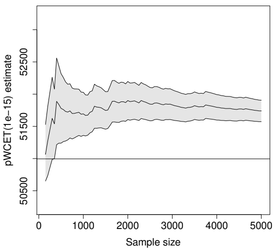


(d) Exponential

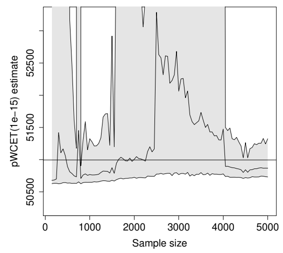
Figure 88: pWCET HWM reliability for *cnt* on *DPArptdm*



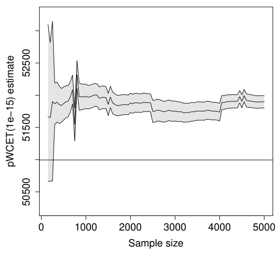
(a) GEV



(b) Gumbel



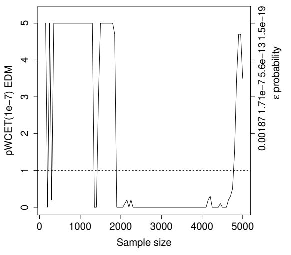
(c) GP



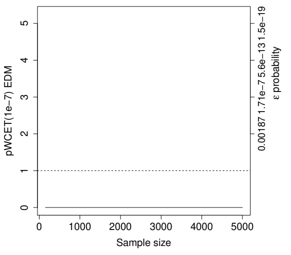
(d) Exponential

A.3 PWCET DENSITY RELIABILITY

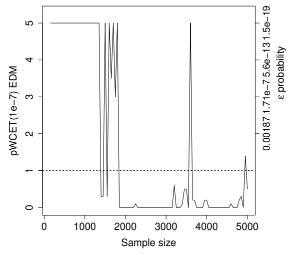
Figure 89: pWCET density reliability for *bsort* on *DPCpArrr*



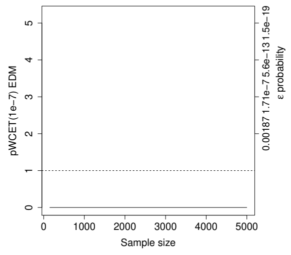
(a) GEV



(b) Gumbel

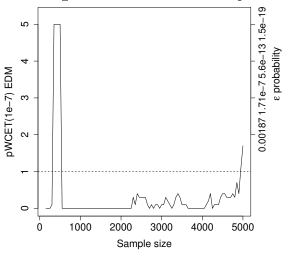


(c) GP

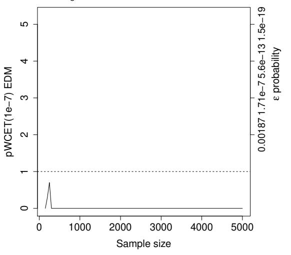


(d) Exponential

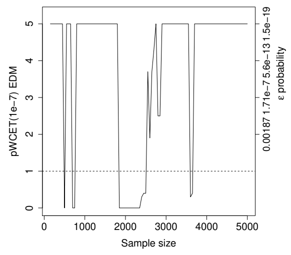
Figure 90: pWCET density reliability for *bsort* on *DPArptdm*



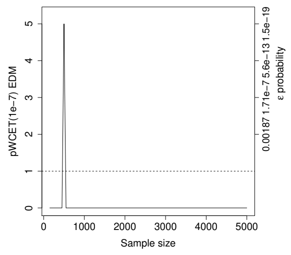
(a) GEV



(b) Gumbel

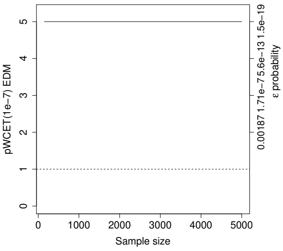


(c) GP

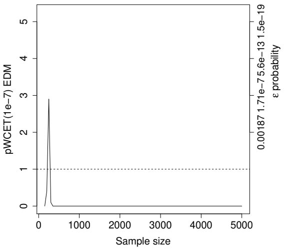


(d) Exponential

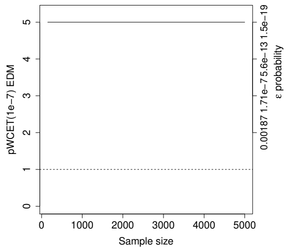
Figure 91: pWCET density reliability for *insertsort* on *DPCpArrr*



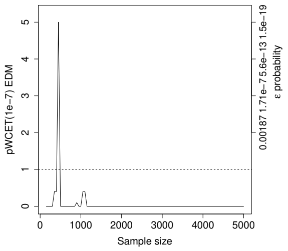
(a) GEV



(b) Gumbel

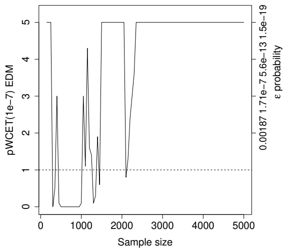


(c) GP

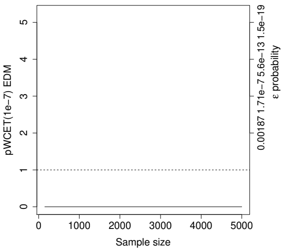


(d) Exponential

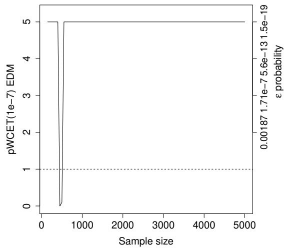
Figure 92: pWCET density reliability for *insertsort* on *DPArptdm*



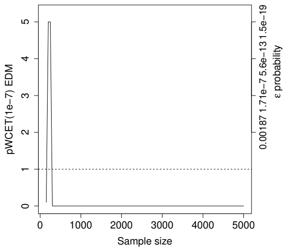
(a) GEV



(b) Gumbel

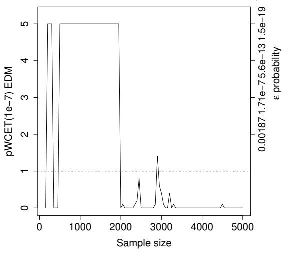


(c) GP

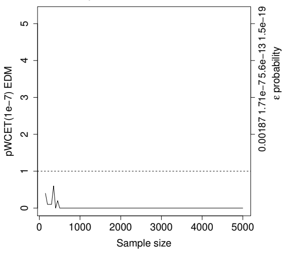


(d) Exponential

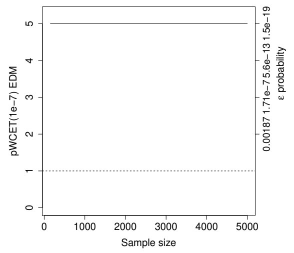
Figure 93: pWCET density reliability for *bs* on *DPCpArrr*



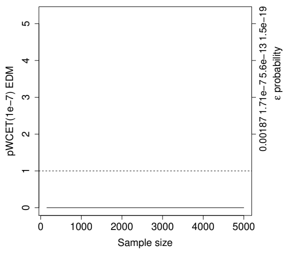
(a) GEV



(b) Gumbel

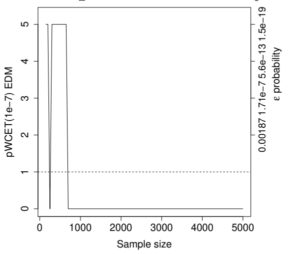


(c) GP

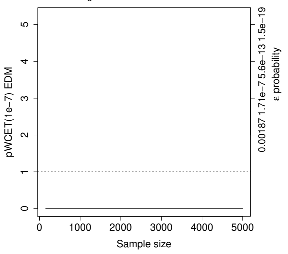


(d) Exponential

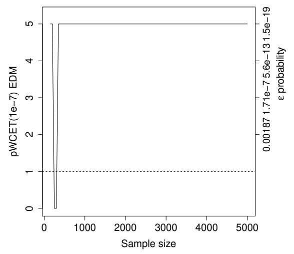
Figure 94: pWCET density reliability for *bs* on *DPArptdm*



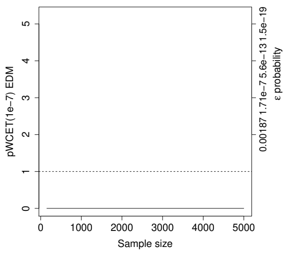
(a) GEV



(b) Gumbel

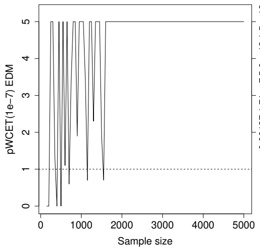


(c) GP

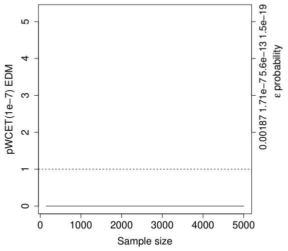


(d) Exponential

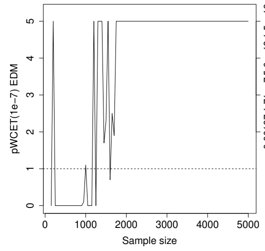
Figure 95: pWCET density reliability for *expint* on *DPCpArrr*



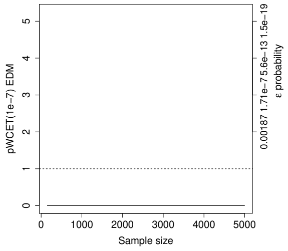
(a) GEV



(b) Gumbel

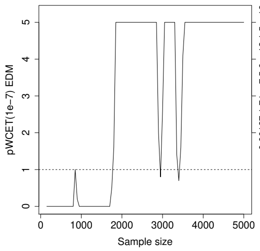


(c) GP

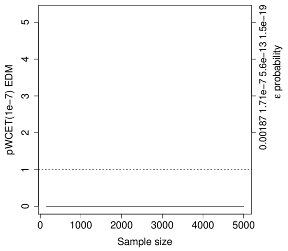


(d) Exponential

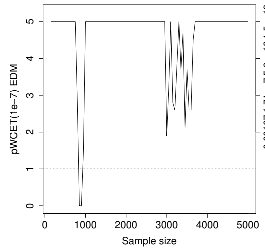
Figure 96: pWCET density reliability for *expint* on *DPArptdm*



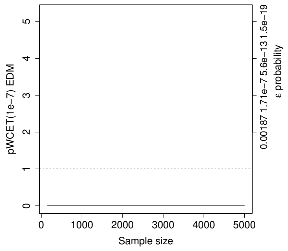
(a) GEV



(b) Gumbel

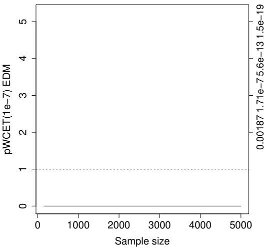


(c) GP

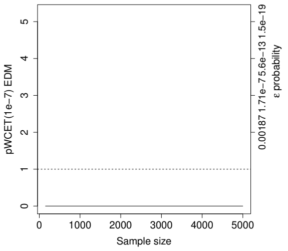


(d) Exponential

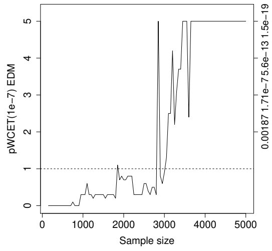
Figure 97: pWCET density reliability for *fdct* on *DPCpArrr*



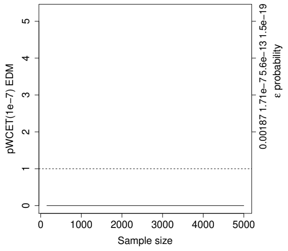
(a) GEV



(b) Gumbel

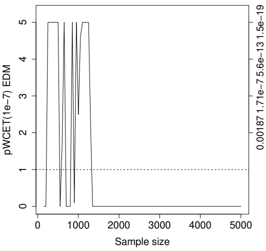


(c) GP

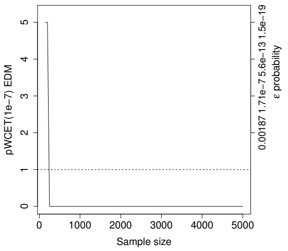


(d) Exponential

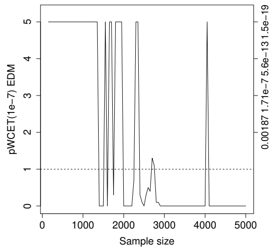
Figure 98: pWCET density reliability for *fdct* on *DPArptdm*



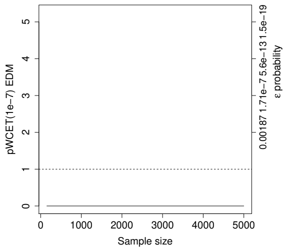
(a) GEV



(b) Gumbel

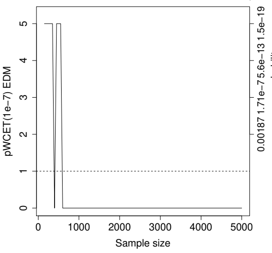


(c) GP

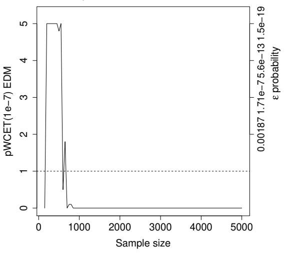


(d) Exponential

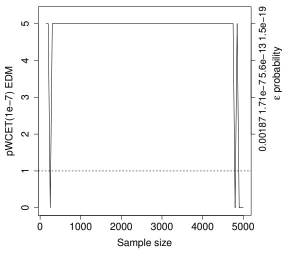
Figure 99: pWCET density reliability for *crc* on *DPCpArrr*



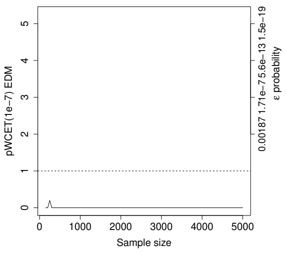
(a) GEV



(b) Gumbel

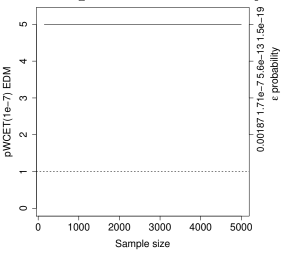


(c) GP

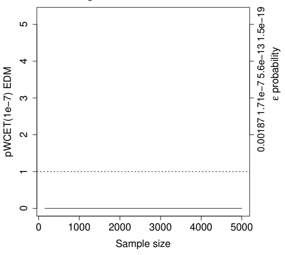


(d) Exponential

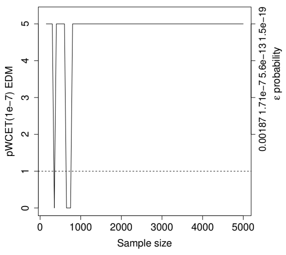
Figure 100: pWCET density reliability for *crc* on *DPArptdm*



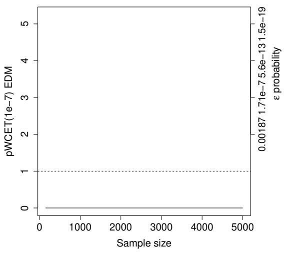
(a) GEV



(b) Gumbel

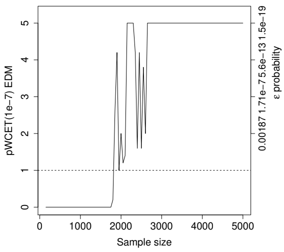


(c) GP

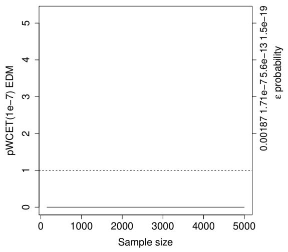


(d) Exponential

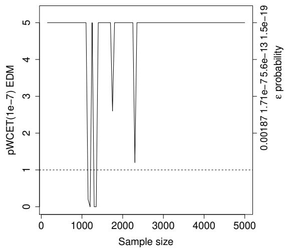
Figure 101: pWCET density reliability for *matmult* on *DPCpArrr*



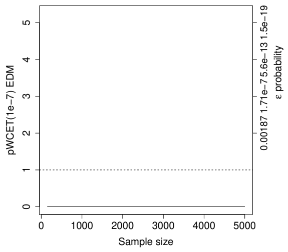
(a) GEV



(b) Gumbel

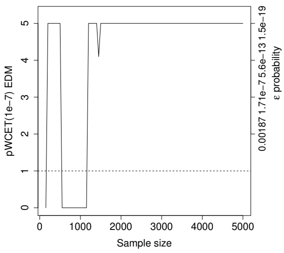


(c) GP

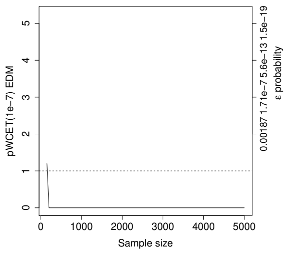


(d) Exponential

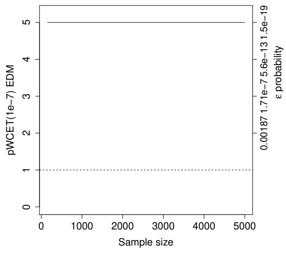
Figure 102: pWCET density reliability for *matmult* on *DPArptdm*



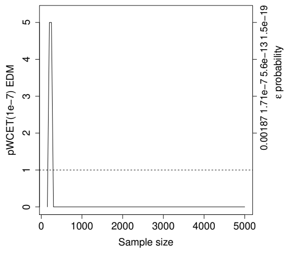
(a) GEV



(b) Gumbel

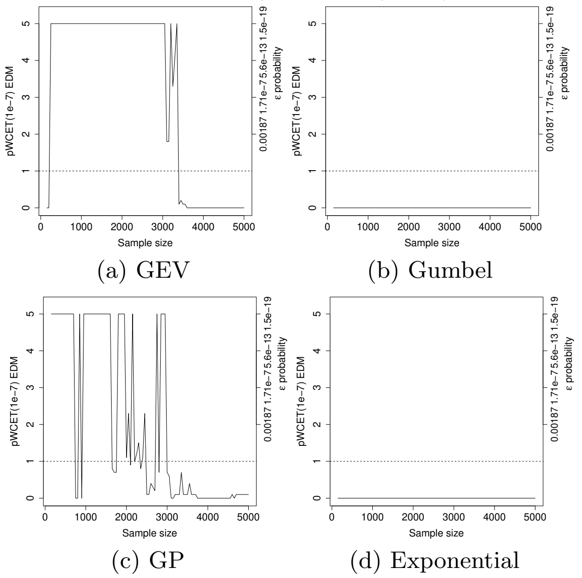


(c) GP



(d) Exponential

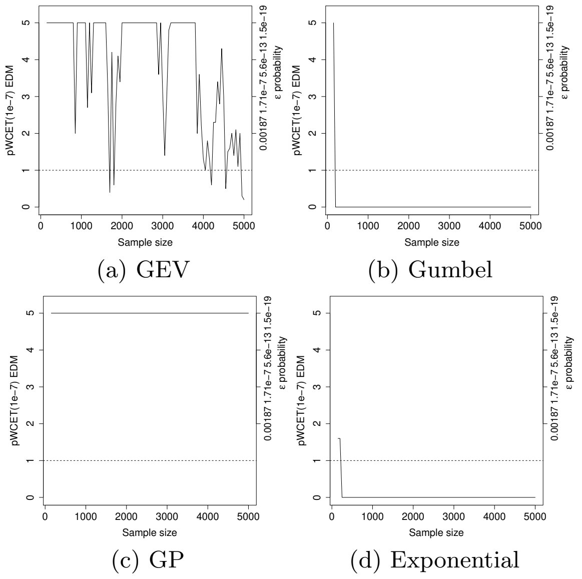
Figure 103: pWCET density reliability for *fir* on *DPCpArrr*



(a) GEV (b) Gumbel

(c) GP (d) Exponential

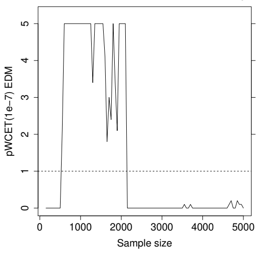
Figure 104: pWCET density reliability for *fir* on *DPArptdm*



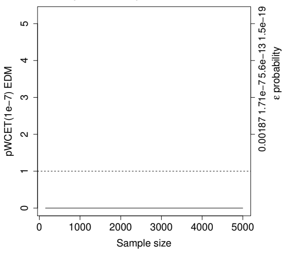
(a) GEV (b) Gumbel

(c) GP (d) Exponential

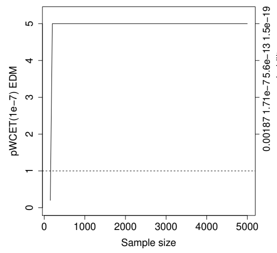
Figure 105: pWCET density reliability for *fibcall* on *DPCpArrr*



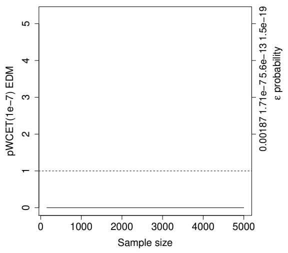
(a) GEV



(b) Gumbel

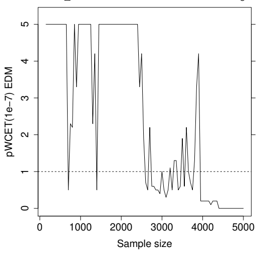


(c) GP

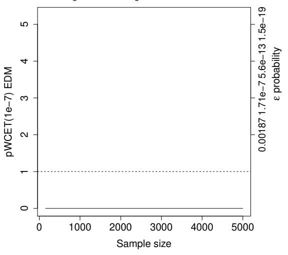


(d) Exponential

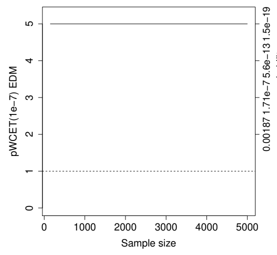
Figure 106: pWCET density reliability for *fibcall* on *DPArptdm*



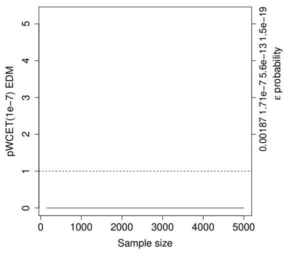
(a) GEV



(b) Gumbel

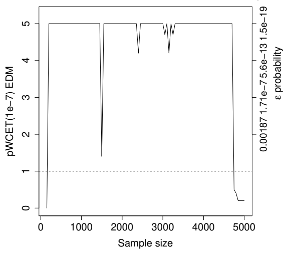


(c) GP

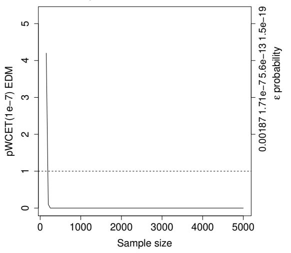


(d) Exponential

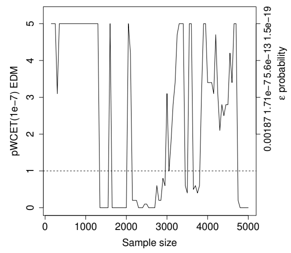
Figure 107: pWCET density reliability for *cnt* on *DPCpArrr*



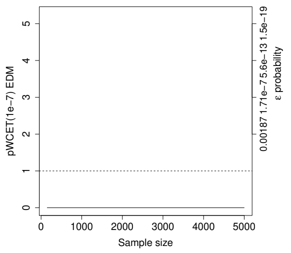
(a) GEV



(b) Gumbel

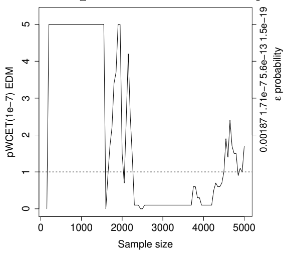


(c) GP

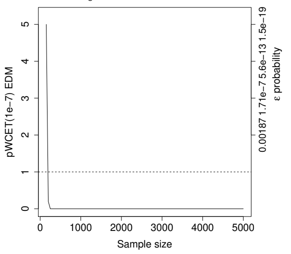


(d) Exponential

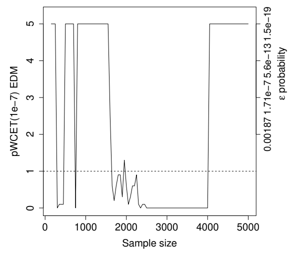
Figure 108: pWCET density reliability for *cnt* on *DPArptdm*



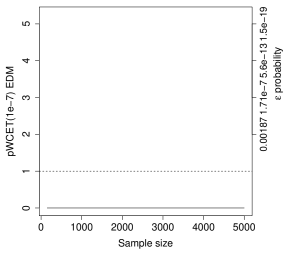
(a) GEV



(b) Gumbel



(c) GP



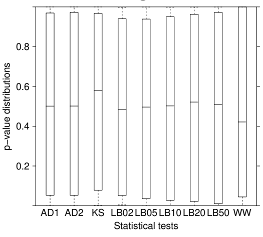
(d) Exponential

APPENDIX B – Evaluating Randomized Scheduling on Multithread Pipelines to Benefit MBPTA

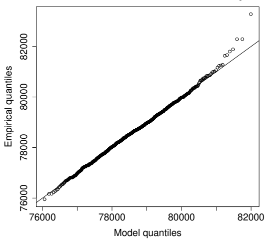
B.1 PRS EVALUATION

B.1.1 Maxima Analysability

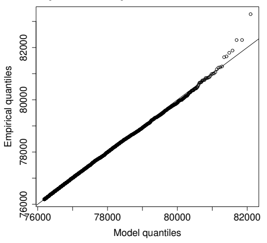
Figure 109: *crc* PRS maxima analysability analysis



(a) I.i.d. statistical tests

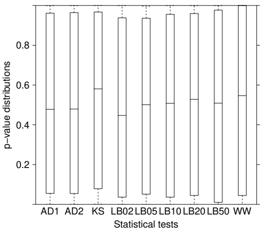


(b) GEV quantile plot

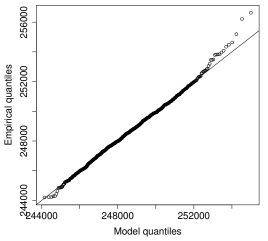


(c) GP quantile plot

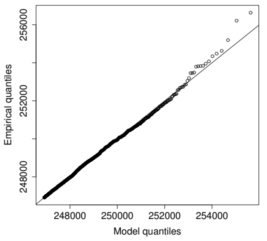
Figure 110: *expint* PRS maxima analysability analysis



(a) I.i.d. statistical tests

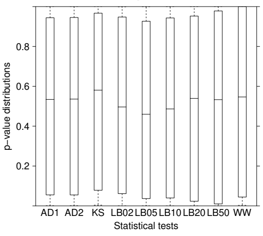


(b) GEV quantile plot

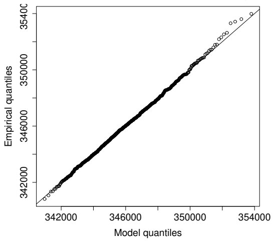


(c) GP quantile plot

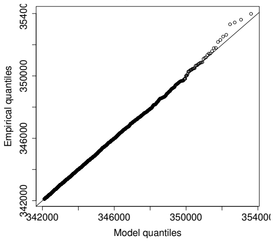
Figure 111: *fdct* PRS maxima analysability analysis



(a) I.i.d. statistical tests

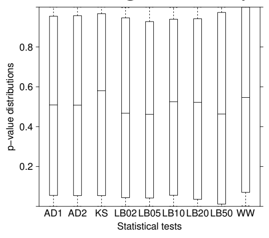


(b) GEV quantile plot

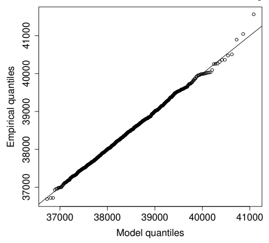


(c) GP quantile plot

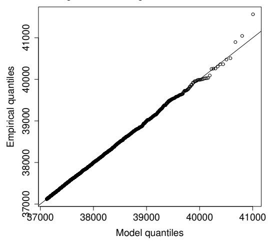
Figure 112: *fibcall* PRS maxima analysability analysis



(a) I.i.d. statistical tests

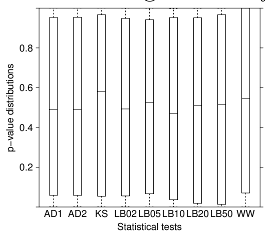


(b) GEV quantile plot

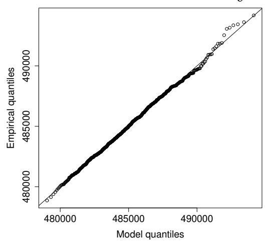


(c) GP quantile plot

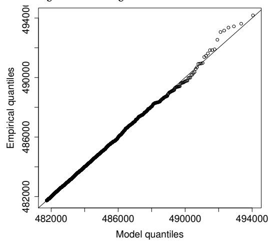
Figure 113: *fir* PRS maxima analysability analysis



(a) I.i.d. statistical tests

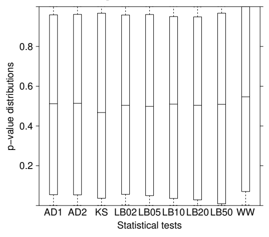


(b) GEV quantile plot

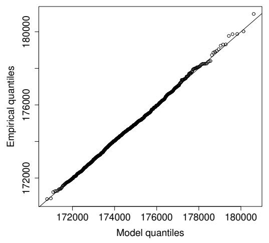


(c) GP quantile plot

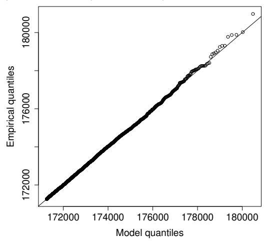
Figure 114: *insertsort* PRS maxima analysability analysis



(a) I.i.d. statistical tests

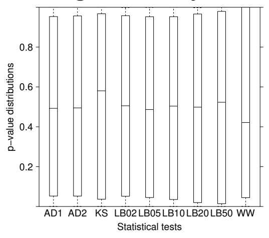


(b) GEV quantile plot

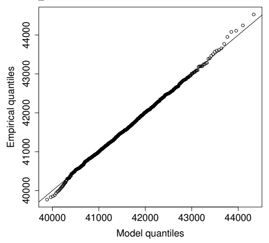


(c) GP quantile plot

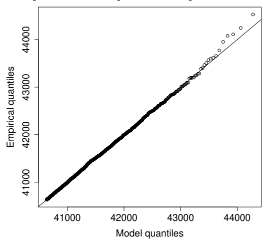
Figure 115: *janne_complex* PRS maxima analysability analysis



(a) I.i.d. statistical tests

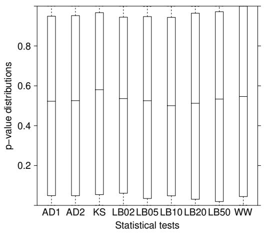


(b) GEV quantile plot

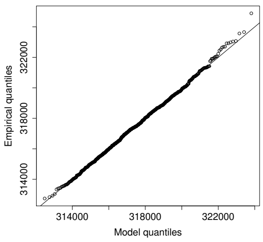


(c) GP quantile plot

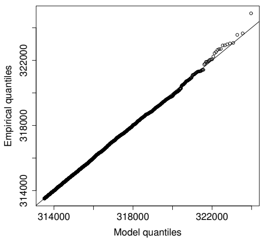
Figure 116: *matmult* PRS maxima analysability analysis



(a) I.i.d. statistical tests

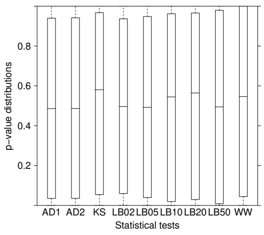


(b) GEV quantile plot

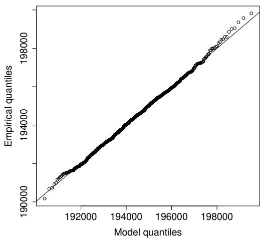


(c) GP quantile plot

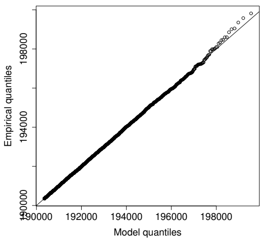
Figure 117: *ns* PRS maxima analysability analysis



(a) I.i.d. statistical tests

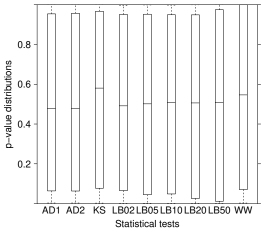


(b) GEV quantile plot

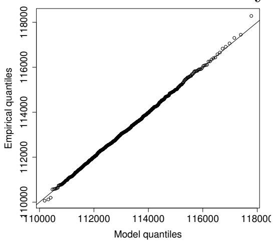


(c) GP quantile plot

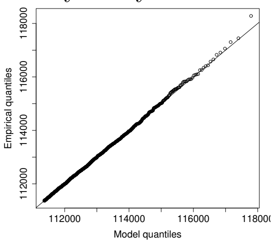
Figure 118: *prime* PRS maxima analysability analysis



(a) I.i.d. statistical tests



(b) GEV quantile plot



(c) GP quantile plot

B.1.2 Class Timing Dominance

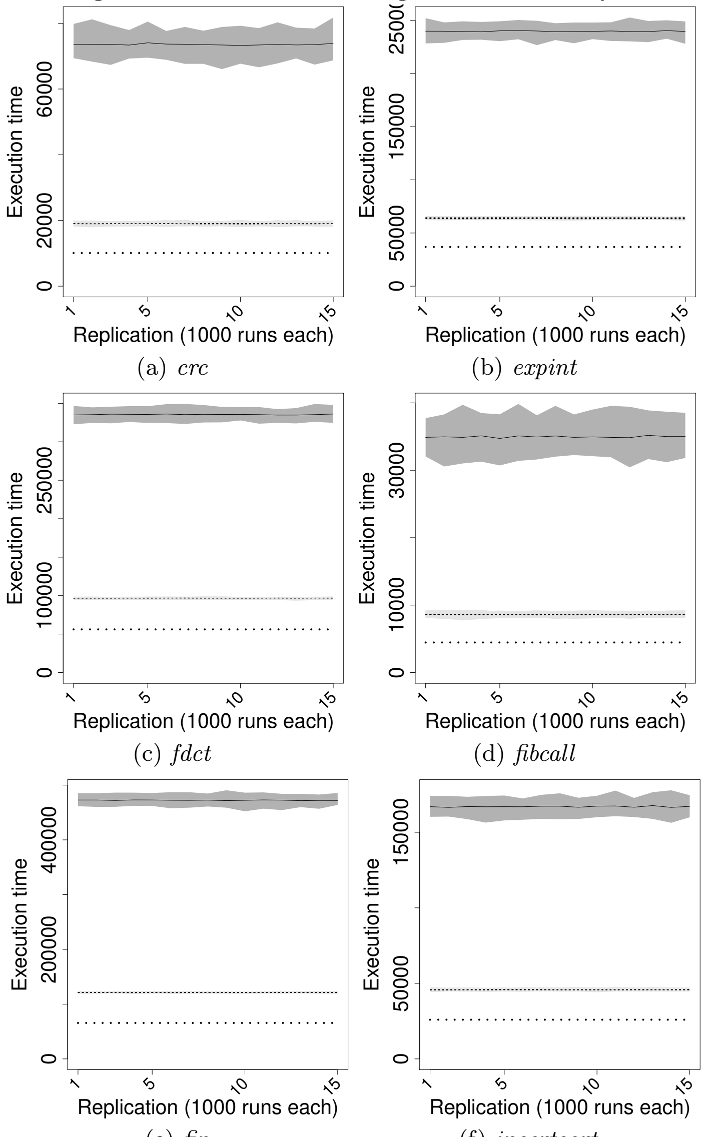


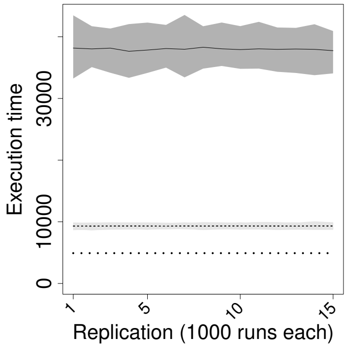
Figure 119: PRS class timing dominance analysis

(a) *crc* (b) *expint*

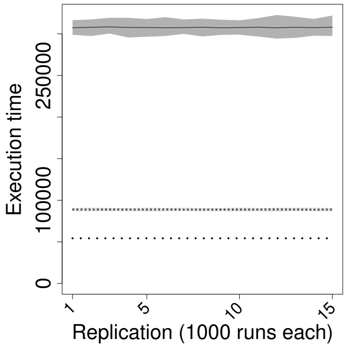
(c) *fdct* (d) *fibcall*

(e) *fir* (f) *insertsort*

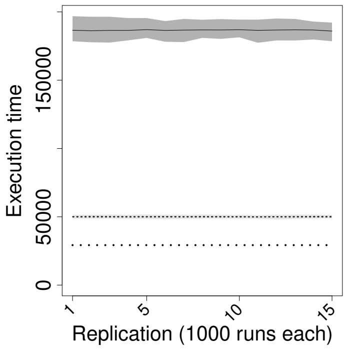
Figure 120: PRS class timing dominance analysis



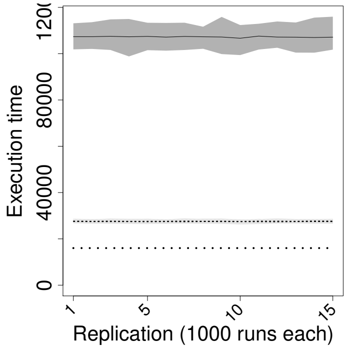
(a) *janne_complex*



(b) *matmult*



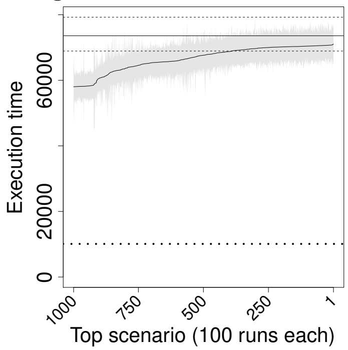
(c) *ns*



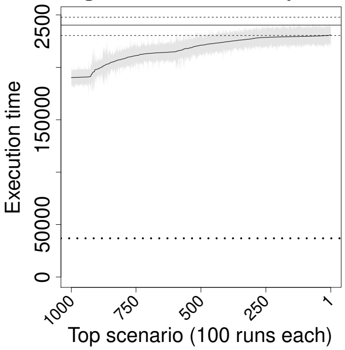
(d) *prime*

B.1.3 Behavioural Timing Dominance

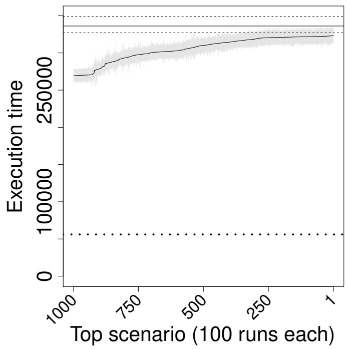
Figure 121: PRS behavioural timing dominance analysis



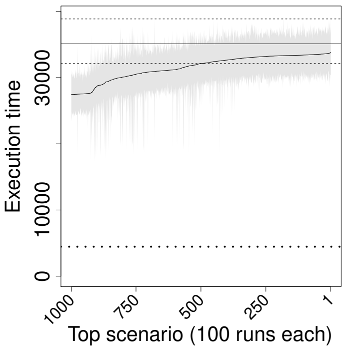
(a) *crc*



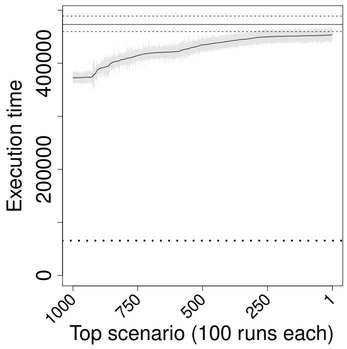
(b) *expint*



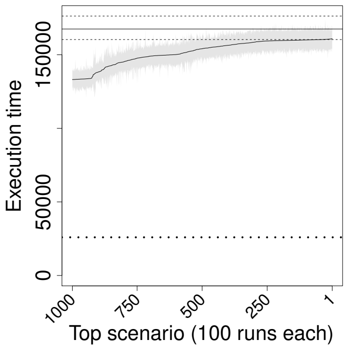
(c) *fdct*



(d) *fibcall*

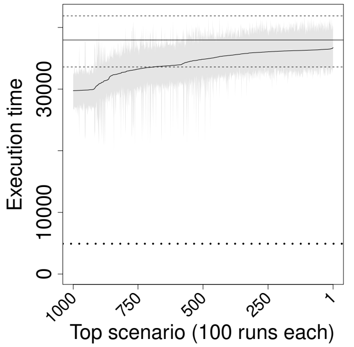


(e) *fir*

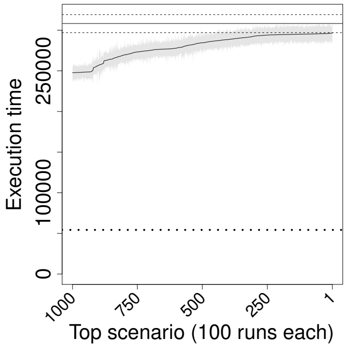


(f) *insertsort*

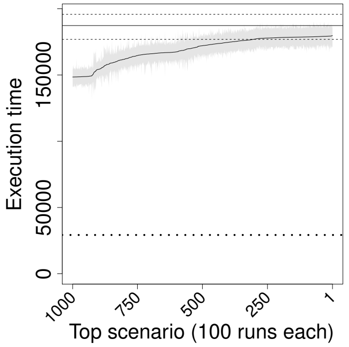
Figure 122: PRS behavioural timing dominance analysis



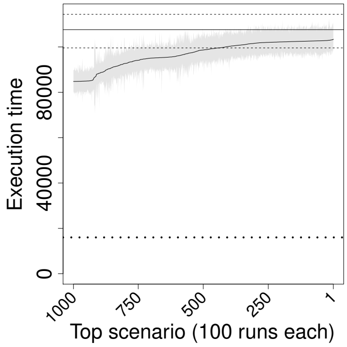
(a) *janne_complex*



(b) *matmult*



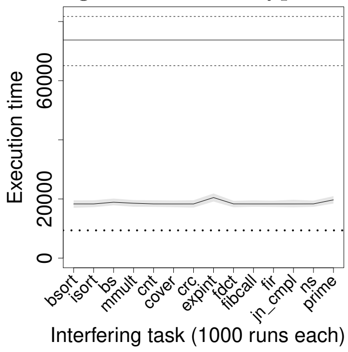
(c) *ns*



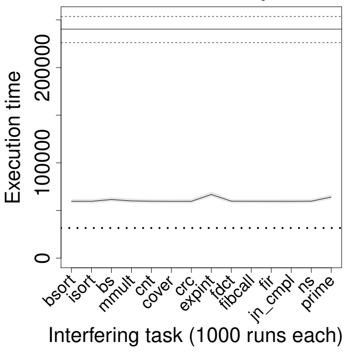
(d) *prime*

B.1.4 Typical Scenario Slowdown

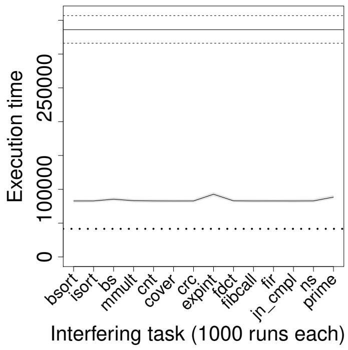
Figure 123: PRS typical scenario slowdown analysis



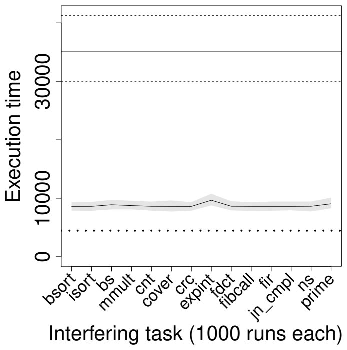
(a) *crc*



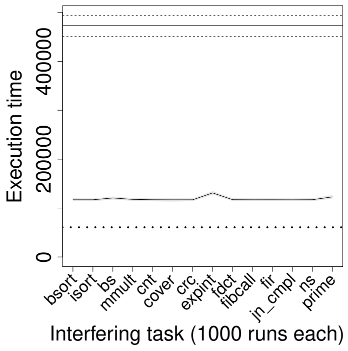
(b) *expint*



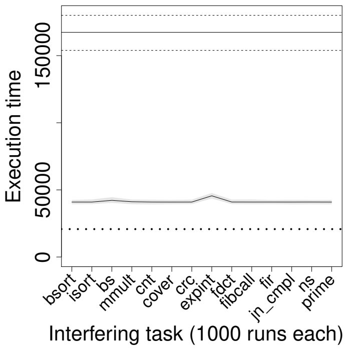
(c) *fdct*



(d) *fibcall*

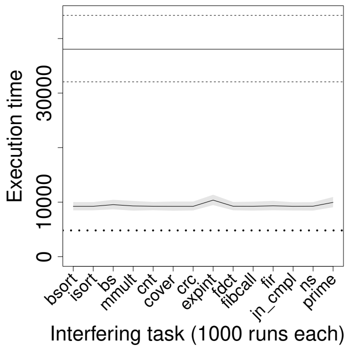


(e) *fir*

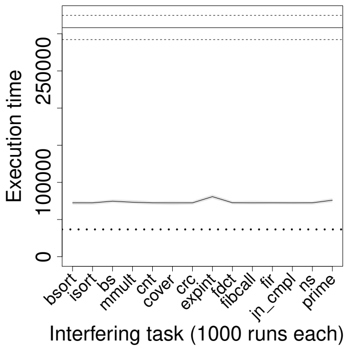


(f) *insertsort*

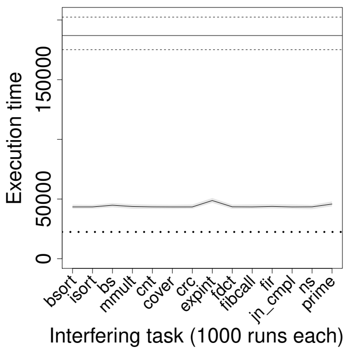
Figure 124: PRS typical scenario slowdown analysis



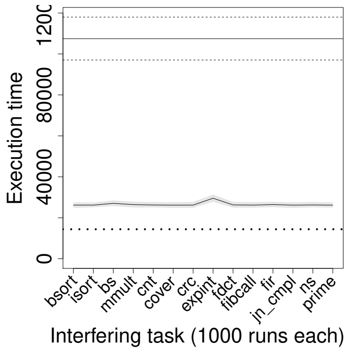
(a) *janne_complex*



(b) *matmult*



(c) *ns*

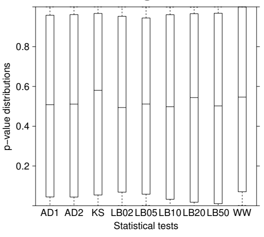


(d) *prime*

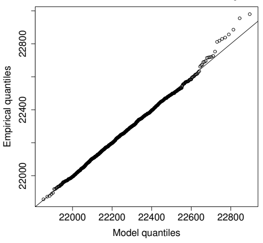
B.2 IRS EVALUATION

B.2.1 Maxima Analysability

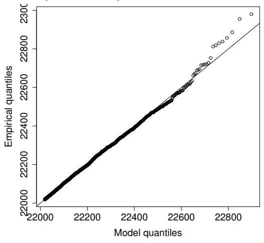
Figure 125: *crc* IRS maxima analysability analysis



(a) I.i.d. statistical tests

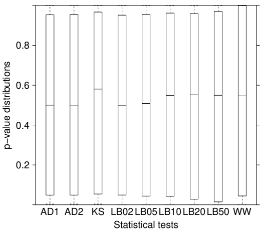


(b) GEV quantile plot

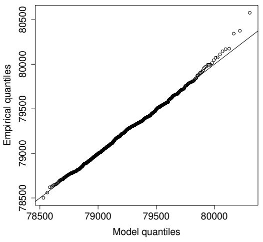


(c) GP quantile plot

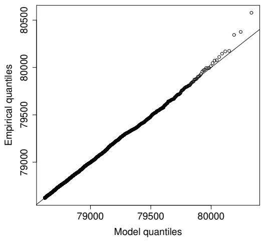
Figure 126: *expint* IRS maxima analysability analysis



(a) I.i.d. statistical tests

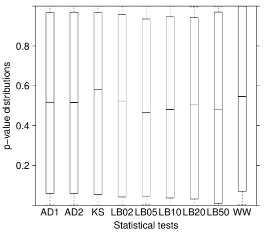


(b) GEV quantile plot

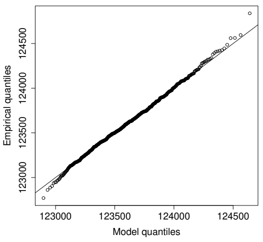


(c) GP quantile plot

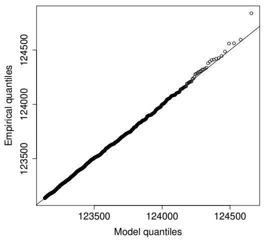
Figure 127: *fdct* IRS maxima analysability analysis



(a) I.i.d. statistical tests

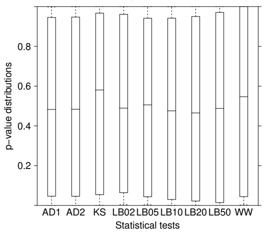


(b) GEV quantile plot

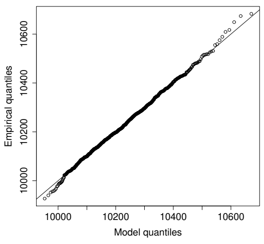


(c) GP quantile plot

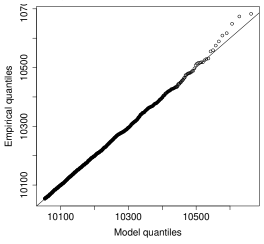
Figure 128: *fibcall* IRS maxima analysability analysis



(a) I.i.d. statistical tests

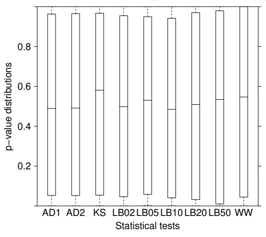


(b) GEV quantile plot

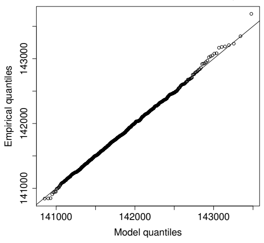


(c) GP quantile plot

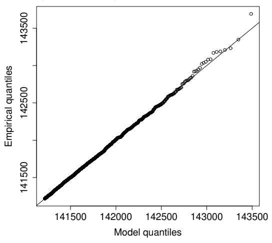
Figure 129: *fir* IRS maxima analysability analysis



(a) I.i.d. statistical tests

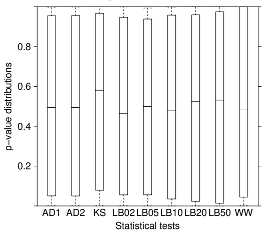


(b) GEV quantile plot

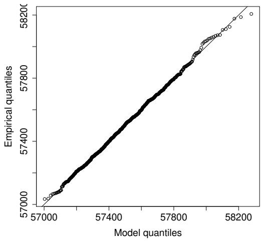


(c) GP quantile plot

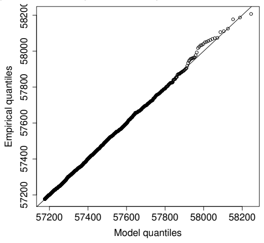
Figure 130: *insertsort* IRS maxima analysability analysis



(a) I.i.d. statistical tests

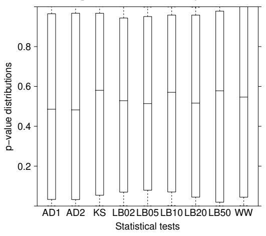


(b) GEV quantile plot

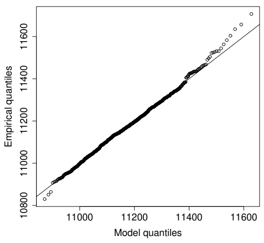


(c) GP quantile plot

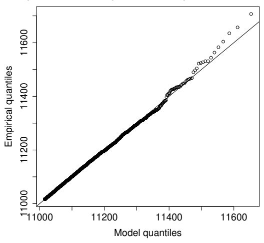
Figure 131: *janne_complex* IRS maxima analysability analysis



(a) I.i.d. statistical tests

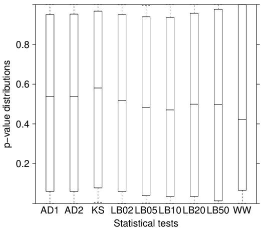


(b) GEV quantile plot

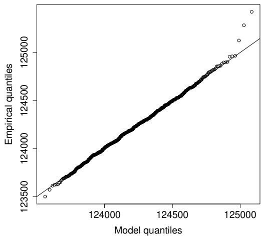


(c) GP quantile plot

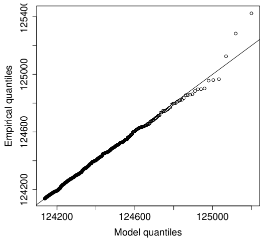
Figure 132: *matmult* IRS maxima analysability analysis



(a) I.i.d. statistical tests

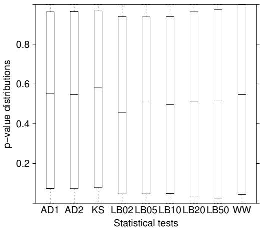


(b) GEV quantile plot

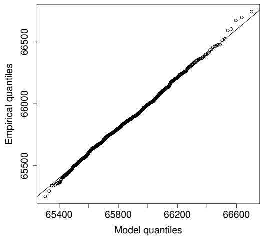


(c) GP quantile plot

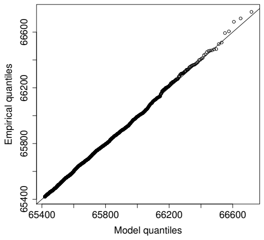
Figure 133: *ns* IRS maxima analysability analysis



(a) I.i.d. statistical tests

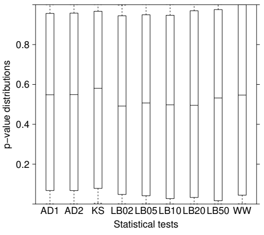


(b) GEV quantile plot

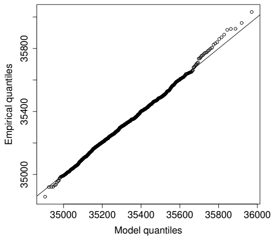


(c) GP quantile plot

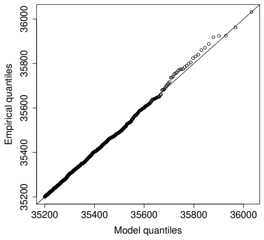
Figure 134: *prime* IRS maxima analysability analysis



(a) I.i.d. statistical tests



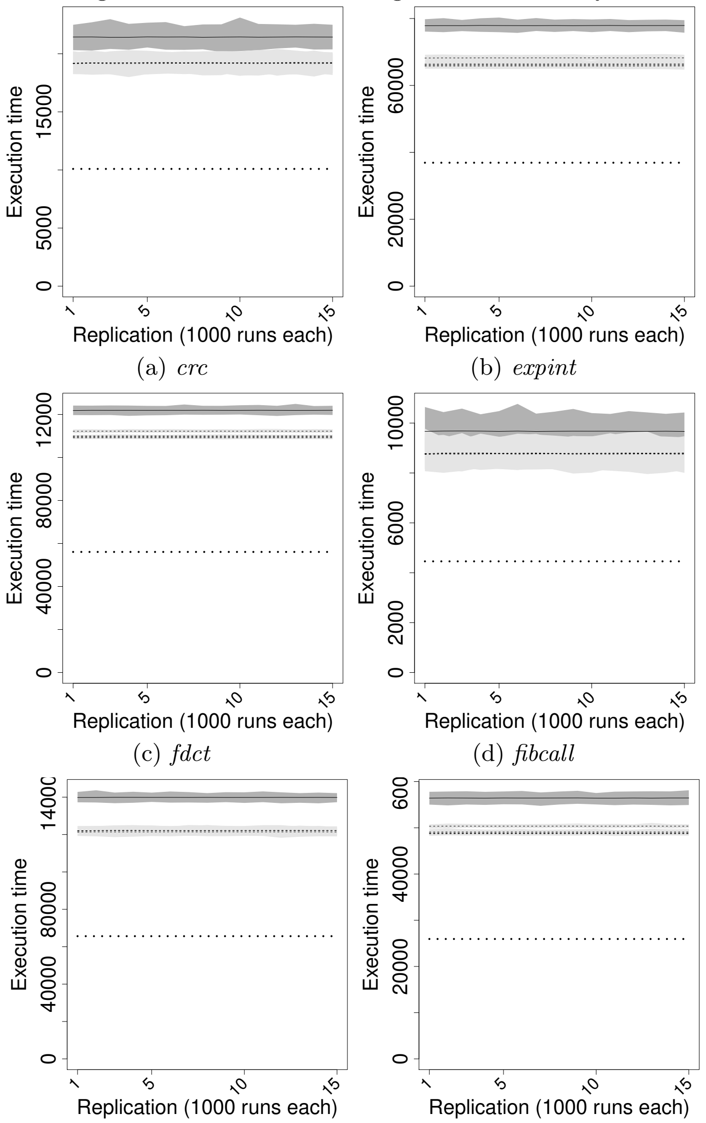
(b) GEV quantile plot



(c) GP quantile plot

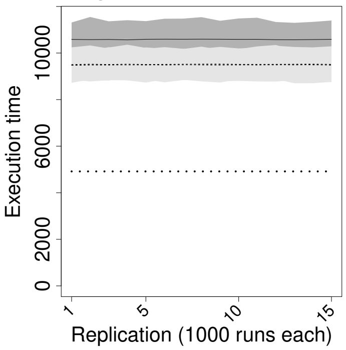
B.2.2 Class Timing Dominance

Figure 135: IRS class timing dominance analysis

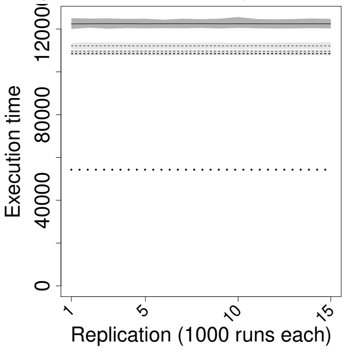


(a) *crc* (b) *expint* (c) *fdct* (d) *fibcall* (e) *fir* (f) *insertsort*

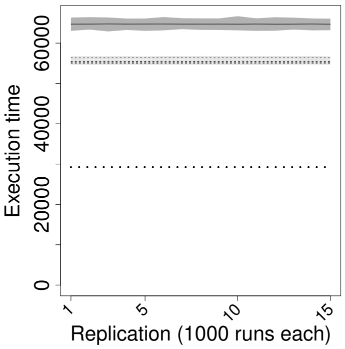
Figure 136: IRS class timing dominance analysis



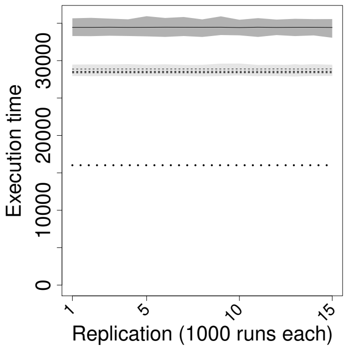
(a) *janne_complex*



(b) *matmult*



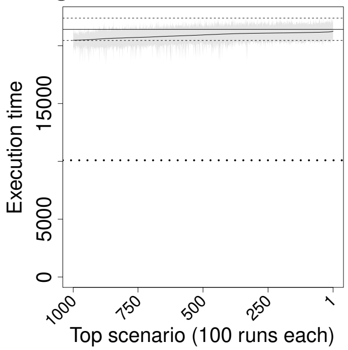
(c) *ns*



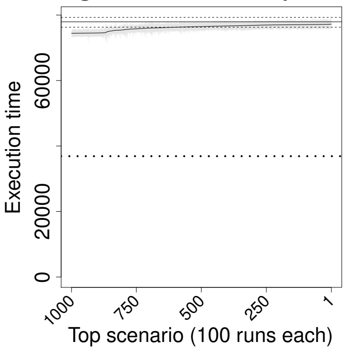
(d) *prime*

B.2.3 Behavioural Timing Dominance

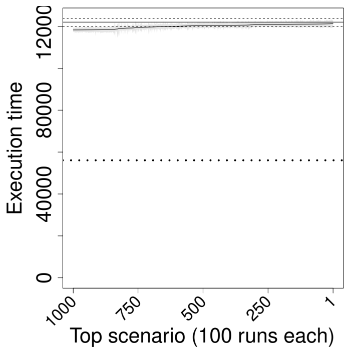
Figure 137: IRS behavioural timing dominance analysis



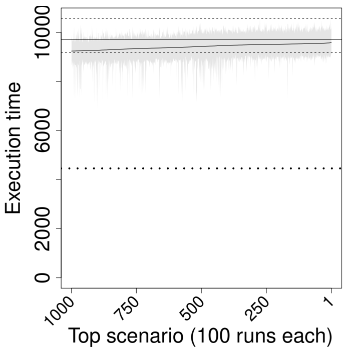
(a) *crc*



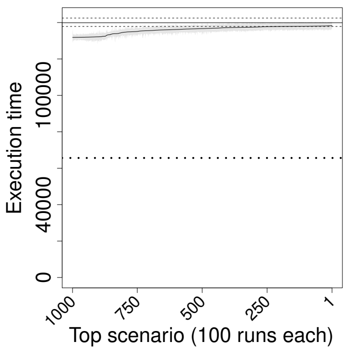
(b) *expint*



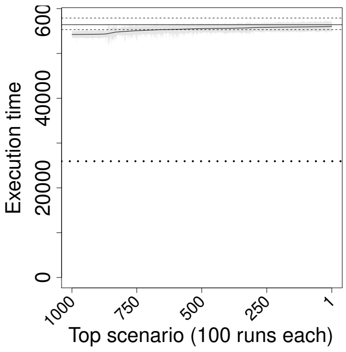
(c) *fdct*



(d) *fibcall*

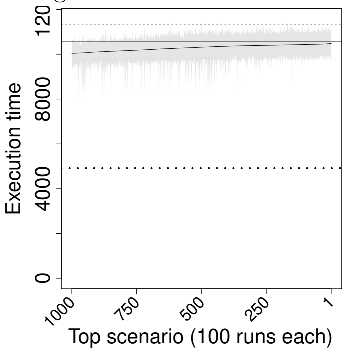


(e) *fir*

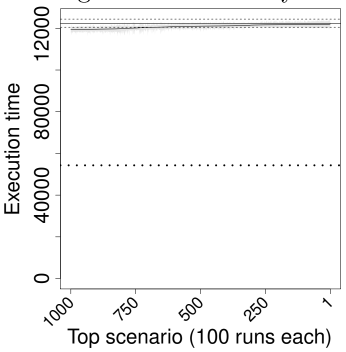


(f) *insertsort*

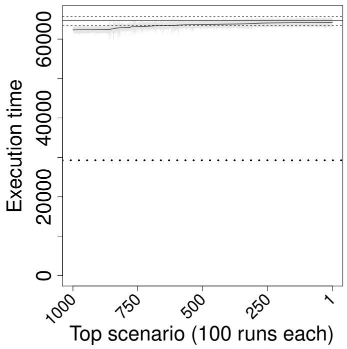
Figure 138: IRS behavioural timing dominance analysis



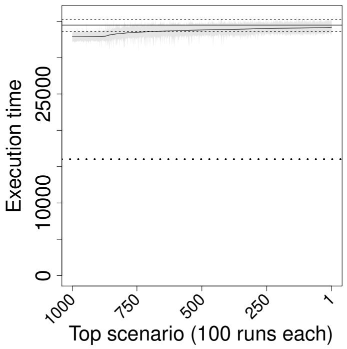
(a) *janne_complex*



(b) *matmult*



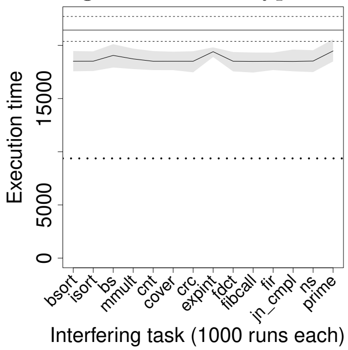
(c) *ns*



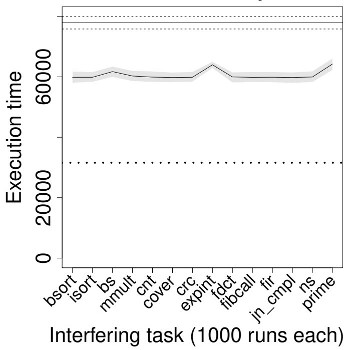
(d) *prime*

B.2.4 Typical Scenario Slowdown

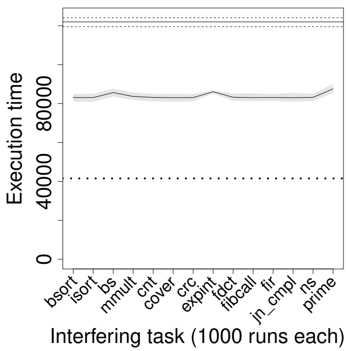
Figure 139: IRS typical scenario slowdown analysis



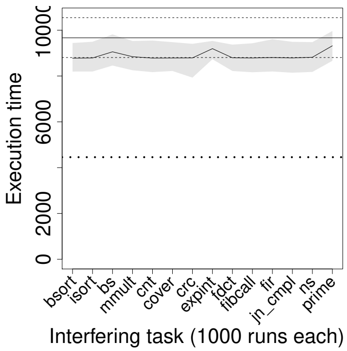
(a) *crc*



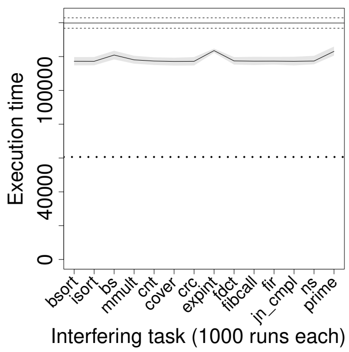
(b) *expint*



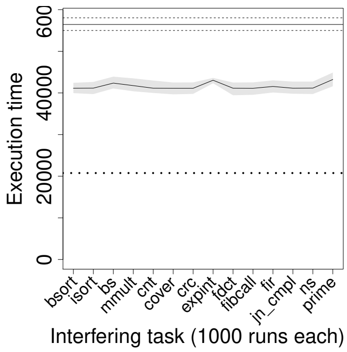
(c) *fdct*



(d) *fibcall*

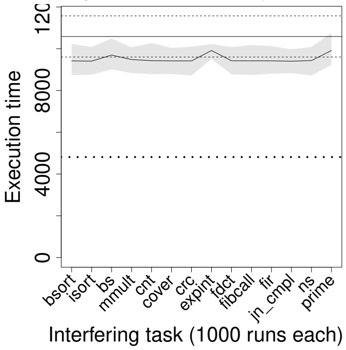


(e) *fir*

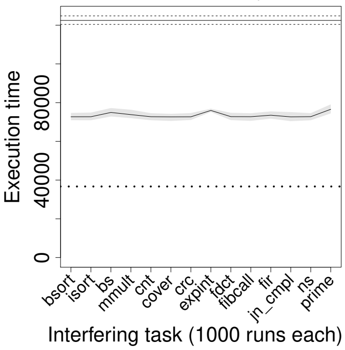


(f) *insertsort*

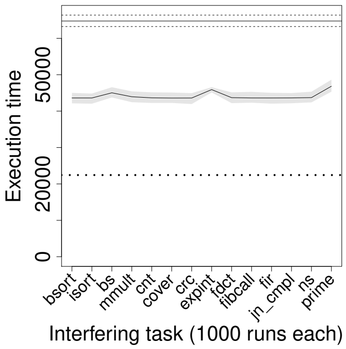
Figure 140: IRS typical scenario slowdown analysis



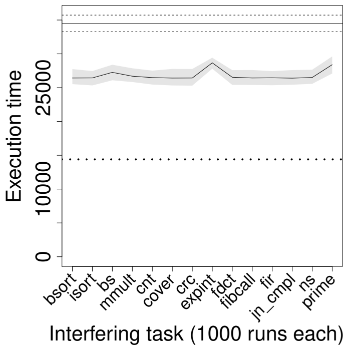
(a) *janne_complex*



(b) *matmult*



(c) *ns*



(d) *prime*

APPENDIX C – Experimentation Platforms

In this chapter we present technical details (1) of the hardware elements developed during the research whose outcomes are presented in this thesis, and (2) of the software tools that were necessary for building and using the hardware platforms that were employed in this work.

C.1 HARDWARE ELEMENTS

The following hardware elements were developed during this work, using the SystemVerilog hardware description language, for building the time-randomized processors we employed in our research.

- ALU: Performs operations such as integer negation, addition, subtraction, logical AND, logical OR, exclusive OR, negated OR, shift left and shift right in a single clock cycle. Multiplications are executed with data-dependent latency of 1 to 32 cycles, and divisions are performed using a non-restoring algorithm that always takes 32 clock cycles to complete. Supports a constant-time mode in which data-dependent operations always take the maximum possible latency to be acknowledged as finished, even if results are available sooner. Its validation was performed through a large set of corner-case integer operations, mainly by comparing its results with those produced by a traditional processor.
- Memory: A true dual-port synchronous RAM memory, with byte-enable write support and single-cycle latency for read/write.
- Memory controller: A single-port memory controller capable of performing multi-word read (e.g. for using cache memories) and single-word write with byte-enable support. At least one extra clock cycle is consumed by the controller for each access, and multi-word reads consume one more cycle for each extra word.
- PRNG: A pseudo-random number generator with single clock cycle latency that combines through a XOR operation the results of two independent and different generators, namely a LFSR and a CASR, which can be clocked from independent and non-synchronized ring oscillators (TKACIK, 2003).
- TRNG: A hardware-level generator of true random numbers that consists of a ring of ring oscillators (BAETONIU, 2004), used as entropy source for seeding pseudo-random number generators.
- Simple (sequential) core: A reference sequential processing core, i.e. which fetches, decodes and executes instructions in a sequential manner such that one instruction's execution only starts when

the last one was finished. The core implements the MIPS I ISA, and for being sequential the latencies of each processing stage are cumulative to the instructions' overall execution times. A halting instruction was added to the ISA to ease execution time measurements. The core's functional validation was performed through a set of test cases, ranging from simple expressions to complete algorithms, for which the memory and the registers' values were validated against their known correct values.

- Pipelined core: A core that implements exactly the same ISA as the sequential one, but which employs a five-stage pipeline (i.e. fetch, decode, execute, memory access, and write back) for exploiting instruction-level parallelism. Instruction execution latencies vary depending on the internal pipeline behaviour, which itself depends on other elements' timing (e.g. cache memories and memory controllers). The pipelined core was subject to the same functional validation as the sequential one.
- Time-randomized cache memory: A set-associative cache memory whose size, block size, and way count can be defined through module parameters. It currently implements only a randomized replacement policy, i.e. it defines the cache way in which lines are to be evicted and replaced based on pseudo-random numbers.
- Random Time Division Multiplexing (TDM) bus arbiter: Randomly chooses the next client to be served, reserving a time slice regardless of the resource being effectively accessed, hence providing temporal isolation but allowing starvation.
- Random Round-Robin (RR) bus arbiter: Randomly chooses the next client to be served, allocating the resource only if a request is pending, not providing temporal isolation and allowing starvation.
- Random Permutation TDM bus arbiter: Uses a non-work-conserving time schedule that is randomly permuted after each round, providing temporal isolation and avoiding starvation.

C.2 DATA COLLECTION TOOLS

The developed hardware elements were synthesized for two similar educational boards, namely DE2-115 and DE2i-150 from Terasic, both equipped with Cyclone IV Altera FPGAs. The synthesis proved necessary both for validating the hardware implementation and for accelerating execution time measurements' collection, since simulation proved too slow for obtaining samples of reasonable size. For enabling data collection applications to control the hardware platforms executed on the FPGA, we also developed a special hardware element referred to as “manager”. This element is capable of communicating with an external computer through the IEEE 802.3 (Ethernet) physical layer transceiver provided by the employed development boards, and provides data collection applications with the following functionalities:

- Inquiring configuration: Allows checking the configuration of the processor, such as the number of cores and bus widths.
- Memory management: Enables reading, writing and internally copying the contents of instruction and data RAM memories.
- Clearing caches: For clearing individual cache memories’ contents.
- Setting PRNGs’ seeds: Allows the seeds (initial random values) of the pseudo-random number generators being set.
- Configuring cores: Supports the assignment of configuration words, used to determine specific aspects of the developed hardware platforms’ behaviour, without requiring them to be re-synthesized.
- Thread scheduling: Performs task scheduling at the thread level, supporting periodic, aperiodic and sporadic tasks’ dispatching, by monitoring and controlling their reset and halt signals.
- Execution time measurement: Measurements are taken in a cycle-precise and non-intrusive manner by manipulating the cores’ reset signals and monitoring their halt signals. More specifically, only clock cycles in which the reset and the halt signals of the cores are both de-asserted are considered in measurements.

C.3 OTHER SOFTWARE TOOLS

The following software tools were also developed during this work for supporting and automating the platforms’ generation and synthesis, and also for accelerating data collection and analysis processes:

- Hardware platform generator: Based on a platform configuration description, generates the hardware platform’s main module, by instantiating and interconnecting e.g. memories and cores
- Data collection applications: Capable of invoking external tools for synthesizing and programming generated processors on the development boards, and also of communicating with the manager to configure, execute and measure programs’ execution times.