UNIVERSIDADE FEDERAL DE PELOTAS
Centro de Desenvolvimento Tecnológico
Programa de Pós-Graduação em Computação

Dissertação

Têmpera Simulada aplicada no Mapeamento Tecnológico de FPGAs baseadas em LUTs

Matheus Garcia Nachtigall

Pelotas, 2015

Matheus Garcia Nachtigall

Têmpera Simulada aplicada no Mapeamento Tecnológico de FPGAs baseadas em LUTs

Dissertação apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Mestre em Ciência da Computação

Orientador: Prof. Dr. Paulo Roberto Ferreira Jr.
Coorientador: Prof. Dr. Felipe de Souza Marques

Pelotas, 2015

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação na Publicação

N124t Nachtigall, Matheus Garcia

Têmpera simulada aplicada no mapeamento tecnológico de FPGAs baseadas em LUTs / Matheus Garcia Nachtigall ; Paulo Roberto Ferreira Júnior, orientador ; Felipe de Souza Marques, coorientador. — Pelotas, 2015.

72 f. : il.

Dissertação (Mestrado) — Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2015.

1. Otimização de circuitos integrados. 2. Mapeamento tecnológico. 3. Têmpera simulada. 4. FPGA. I. Ferreira Júnior, Paulo Roberto, orient. II. Marques, Felipe de Souza, coorient. III. Título.

CDD : 005

Elaborada por Aline Herbstrith Batista CRB: 10/1737

RESUMO

NACHTIGALL, Matheus Garcia. **Têmpera Simulada aplicada no Mapeamento Tecnológico de FPGAs baseadas em LUTs**. 2015. 72 f. Dissertação (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2015.

Existem várias técnicas para a otimização de atributos de circuitos integrados. O foco atual dessas técnicas é a minimização da área do circuito em questão. Porém, as técnicas existentes possuem diversas etapas que precisam ser melhoradas, entre elas a etapa de Mapeamento Tecnológico (MT). O Mapeamento Tecnológico é uma etapa crucial no processo de síntese lógica, pois ele define qual conjunto de elementos lógicos serão utilizados para implementar o circuito na tecnologia alvo. Na literatura existem várias abordagens diferentes para otimização da etapa de mapeamento e atualmente as metodologias iterativas estão se popularizado.

Esta dissertação propõe uma nova abordagem para o Mapeamento Tecnológico de *Field Programmable Gate Arrays* (FPGAs), baseada em técnicas de otimização de Inteligência Artificial (IA), mais especificamente a técnica de Têmpera Simulada. A utilização de uma técnica de IA no Mapeamento Tecnológico é uma abordagem promissora pois se diferencia fortemente das técnicas já existentes, devido aos fatores de aleatoriedade em técnicas de otimização baseados em IA.

A abordagem elaborada age em uma etapa do mapeamento chamada de cobertura, criando uma solução para o circuito baseada no número de *cortes-K* necessários para uma cobertura total do mesmo. Cada *corte-K* pode ser diretamente relacionado a uma Look-Up Table(LUT) da tecnologia FPGA, permitindo assim a geração de um circuito com a lógica equivalente a requisitada. Essa abordagem foi implementada na ferramenta FlexMap, a qual é um *framework* para o desenvolvimento de métodos para o MT. Foram realizados testes em 85 benchmarks dos pacotes ISCAS85 e MCNC91, amplamente conhecidos na área e frequentemente utilizados para testes de desempenho de novas abordagens. Os testes realizados apresentaram resultados promissores, mostrando que a abordagem desenvolvida consegue encontrar soluções comparáveis em vários casos à ferramenta ABC, considerada estado-da-arte no processo de MT. Os resultados obtidos pela abordagem proposta obtiveram melhoras em aproximadamente 19% dos casos avaliados com *K=4* e 26% dos casos com *K=5* sobre os resultados do ABC.

Palavras-chave: Otimização de Circuitos Integrados, Mapeamento Tecnológico, Têmpera Simulada, FPGA.

ABSTRACT

NACHTIGALL, Matheus Garcia. **Simulated Annealing applied to LUT-based FPGA Technology Mapping**. 2015. 72 f. Dissertação (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2015.

Currently there are several techniques for integrated circuit's atribute optimization. The current focus of these techniques is to minimize the area of the given circuit. These current techniques, however, have several stages that need improvement, including the Technology Mapping stage. The technology mapping is a crucial step in the logic synthesis process, because it defines which set of logic elements will be used to implement the circuit in the target technology. In the literature there are several different approaches to optimize the mapping stage and currently iterative methodologies are becoming popular.

This dissertation proposes a new approach to Technology Mapping of *Field Programmable Gate Arrays* (FPGAs), based on optimization techniques using Artificial Intelligence (AI), more specifically the Simulated Annealing technique, in order to propose an alternative solution to the problem. The utilization of an AI technique in technology mapping is promissing approach because it strongly differs from existing techniques due to the randomness factors in optimization techniques based on AI.

The developed approach acts on the mapping stage called coverage, creating a solution for the circuit based on the number of *k-cuts* needed for a complete coverage. Each *k-cut* can be directly related to a FPGA's Look-Up Table (LUT), allowing the generation of a circuit equivalent to the required logic. This approach has been implemented in the FlexMap tool, which is a framework for developing Technology Mapping methods. Tests were performed in 85 benchmarks of the ISCAS85 and MCNC91 packages, widely known in the area and commonly used for performance testing of new approaches. The tests conducted in the implemented approach had promising results, showing that the developed technique can find solutions comparable in several cases to the ABC tool, which is considered state-of-the-art in the Technology Mapping process. The results obtained by the proposed approach obtained improvements in approximately 19% of the evaluated benchmarks with *K=4* and 26% with *K=5* over ABC's results.

Keywords: Integrated Circuit Optimization, Technology Mapping, Simulated Annealing, FPGA.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

IA	Inteligência Artificial
MT	Mapeamento Tecnológico
TS	Têmpera Simulada
VLSI	*Very Large Scale Integration*
DAG	*Directed Acyclic Graph*
AIG	*And Inverter Graph*
FPGA	*Field Programmable Gate Arrays*
LUT	*Look-Up Table*
DFS	*Depth-First Search*

SUMÁRIO

1 INTRODUÇÃO

Atualmente existe uma grande demanda de desenvolvimento de técnicas para automação do processo de concepção dos circuitos integrados, com os principais objetivos sendo a redução do atraso (CONG; C.; Y., 1999; CONG; DING, 1994), potência (PANDEY; CHATTOPADHYAY, 2003) e área (MANOHARARAJAH; BROWN; VRANESIC, 2006; CHEN; CONG, 2004; KAO; LAI, 2005; TESLENKO; DUBROVA, 2004) de um circuito. O desenvolvimento de novas técnicas para o aprimoramento desse tipo de circuito, chamado de *Very Large Scale Integration (VLSI)*, tem recebido grande atenção tanto na indústria como no meio acadêmico. Embora muitos avanços na área de concepção de circuitos integrados tenham sido desenvolvidos nas últimas décadas, projetar circuitos VLSI eficientes ainda não é uma tarefa trivial. Por este motivo, é dada uma atenção especial à fase de projeto desses circuitos.

O Mapeamento Tecnológico (MT) é a etapa da síntese lógica que escolhe quais portas lógicas serão usadas na implementação de um leiaute em uma dada tecnologia (MARQUES et al., 2007). Esta etapa de mapeamento tem um grande impacto na estrutura do circuito e, consequentemente, no seu desempenho e área. Por isso a etapa de MT é considerada uma das mais importantes e é foco de várias pesquisas e trabalhos.

Existem várias ferramentas que encontram boas soluções para o MT, como por exemplo a ferramenta ABC (ABC, 2014), SIS (SENTOVICH et al., 1992) e VIRMA (MARQUES et al., 2007). Porém essas soluções ainda não alcançam garantidamente o resultado ótimo, dando margem para que novas abordagens sejam implementadas na busca de melhores resultados.

Além das abordagens já existentes na literatura, existem alguns métodos que já utilizam técnicas de Inteligência Artificial no Mapeamento Tecnológico, como por exemplo (KOMMU; POMERANZ, 1993) e (SOUZA; SILVA-FILHO, 2013), que utilizam técnicas de algoritmos genéticos para a síntese de *Field Programmable Gate Arrays* (FPGAs), com o foco em uma otimização de área baseada em *Look-Up Tables* (LUTs). Apesar das abordagens utilizando algoritmos genéticos retornarem resultados interessantes, esses resultados já são ultrapassados pelas ferramentas atuais de síntese lógica.

O Têmpera Simulada (TS) (BERTSIMAS; TSITSIKLIS, 1993; VAN LAARHOVEN; AARTS, 1987) é um algoritmo de otimização global utilizado na minimização de funções em um espaço de busca de soluções. Essa técnica é amplamente conhecida na área de Inteligência Artificial e demonstra bons resultados em vários problemas de otimização (AARTS; KORST, 1988; GOODSELL; OLSON, 1990; SVERGUN, 1999). Considerando a eficiência da técnica e a importância do MT na síntese de circuitos integrados, este trabalho tenta desenvolver uma abordagem alternativa ao problema e avaliar sua eficiência.

1.1 Objetivos

Este trabalho tem como objetivo principal o desenvolvimento de um algoritmo de otimização para o Mapeamento Tecnológico através da aplicação da técnica de Têmpera Simulada na etapa de cobertura deste problema. Além dessa implementação, propõe-se analisar o comportamento do TS quando aplicado na otimização de circuitos, verificando elementos como o ajuste de temperatura, a função de probabilidade de troca de estados, a geração do estado inicial do problema e o número de iterações necessárias para que o algoritmo consiga convergir para uma solução aceitável. Assim, é proposta uma abordagem de MT para FPGAs, onde o foco é a redução do número de LUTs necessárias para uma representação completa do circuito.

1.2 Metodologia

Este trabalho propõe uma implementação da técnica de Têmpera Simulada para a etapa de síntese lógica do Mapeamento Tecnológico (MT), buscando como objetivo principal a minimização de área do circuito. A abordagem proposta tem o foco de mapeamento para FPGAs, que são independentes de bibliotecas de células.

A abordagem implementada foi incorporada a uma ferramenta em desenvolvimento na Universidade Federal de Pelotas, chamada de *FlexMap*. Essa ferramenta implementa alguns métodos de mapeamento baseados em abordagens já existentes na literatura, como por exemplo o mapeamento utilizando bibliotecas virtuais (CORREIA, 2004) e mapeamentos utilizando *cortes-K* (MISHCHENKO et al., 2005). A vantagem dessa ferramenta é a possibilidade de configuração de diversas etapas do fluxo do Mapeamento Tecnológico, como por exemplo, a estrutura de dados, método de casamento, estratégia de cobertura ou função custo. A abordagem implementada foi nomeada *FlexMap-TS*.

A versão inicial do FlexMap-TS foi baseada no algoritmo clássico de Têmpera Simulada. Essa implementação gerou o primeiro Cenário de testes, onde foi rea-

lizado uma bateria de testes extensiva com o objetivo de realizar uma análise de comportamento do algoritmo. Os circuitos de teste utilizados foram selecionados de pacotes de benchmarks normalmente utilizados para testes de técnicas de Mapeamento Tecnológico, chamados de 'ISCAS85' (BRYAN, 1985; HANSEN; YALCIN; HAYES, 1999) e 'MCNC91' (YANG, 1991). Estes pacotes são amplamente conhecidos na área e são utilizados frequentemente para avaliação de técnicas de Mapeamento Tecnológico (CHATTERJEE; MISHCHENKO; BRAYTON, 2006; BERKELAAR; JESS, 1988; MARQUES et al., 2005; CHANG et al., 2010).

A partir dos experimentos realizados foi possível avaliar a eficiência do Cenário 1. Com essa análise, foram identificadas limitações no FlexMap-TS que levaram ao desenvolvimento de um outro Cenário, mais eficiente. O Cenário 2 tratou problemas como o ajuste da variável temperatura do TS e da condição de parada do FlexMap-TS. Após aplicadas essas mudanças, uma nova série de experimentos foi realizada, comprovando a efetividade das melhorias implementadas.

Para avaliar a eficiência do FlexMap-TS, foram comparados os experimentos realizados com os resultados obtidos através da ferramenta ABC (ABC, 2014), considerada estado-da-arte no problema abordado.

Apesar de não conseguir superar o ABC de modo geral, o FlexMap-TS conseguiu vários resultados comparáveis e até melhores que o ABC. A abordagem implementada no Cenário 2 conseguiu encontrar resultados superiores em aproximadamente 19% dos casos com *K=4* e 26% dos casos com *K=5*.

1.3 Organização do trabalho

Este trabalho divide-se em 6 capítulos de forma a explicar todos os principais conceitos envolvidos na implementação, execução e análise dos dados gerados.

No capítulo 2 são abordados os conceitos e definições necessários para entendimento do trabalho. Esse capítulo explica todas as etapas inclusas no processo de Mapeamento Tecnológico, como também os fundamentos do algoritmo de Têmpera Simulada.

O capítulo 3 apresenta a revisão bibliográfica da área de Mapeamento Tecnológico, mostrando quais são as abordagens mais difundidas da literatura. Neste capítulo também é discutida a ferramenta ABC, considerada estado-da-arte no MT. Adicionalmente, também são mostrados os trabalhos da literatura que já utilizam Inteligência Artificial no Mapeamento Tecnológico.

No capítulo 4 é apresentada a abordagem implementada. Este capítulo detalha como a técnica de Têmpera Simulada foi adaptada para o Mapeamento Tecnológico, apresentando as etapas necessárias para a preparação do circuito para sua utilização no TS. Adicionalmente, são apresentados os dois Cenários desenvolvidos durante a

execução deste trabalho, explicando as principais diferenças entre cada um.

No capítulo 5 são apresentados os testes realizados utilizando a ferramenta finalizada, fazendo análises estatísticas e comparando os resultados obtidos com a ferramenta ABC.

Por fim, no capítulo 6, são expostas as conclusões, considerações finais da dissertação e as propostas de trabalhos futuros.

2 CONCEITOS BÁSICOS

O processo de Mapeamento Tecnológico envolve várias etapas e subáreas da síntese de circuitos integrados e é considerado uma das fases mais importantes da etapa de síntese lógica. Este capítulo explica os conceitos necessários para entendimento do Mapeamento Tecnológico e da técnica de Têmpera Simulada.

2.1 Mapeamento Tecnológico

O processo de síntese dos circuitos digitais pode ser executado de várias maneiras. Entretanto, duas delas merecem destaque: *custom* e *semicustom*. A metodologia *custom* é feita manualmente e sem automatização de etapas. Esta metodologia é fortemente dependente da experiência dos projetistas desenvolvedores e pode resultar em circuitos consideravelmente otimizados, porém ela é considerada extremamente complexa quando aplicada em circuitos de grande porte.

A metodologia *semicustom* conta com a utilização de ferramentas de apoio responsáveis por aplicar algoritmos de otimização no circuito, reduzindo drasticamente a complexidade da tarefa e acelerando o processo de síntese. Nessa metodologia existem 2 abordagens consideradas as mais utilizadas: baseada em bibliotecas de células e baseada em matrizes. A abordagem baseada em bibliotecas conta com a utilização de uma biblioteca pré-programada de células, onde o objetivo da otimização é encontrar partes do circuito que possuam a lógica implementada em algum elemento da biblioteca. O objetivo dos algoritmos que utilizam essa abordagem é encontrar a melhor configuração de células para construir o circuito.

A abordagem baseada em matrizes utiliza elementos reconfiguráveis que representam a lógica que se deve implementar. Nessa abordagem, é importante citar os *Field Programable Gate Arrays* (FPGA), que é composta de LUTs (do inglês *Look-Up Table*). A LUT é o componente lógico da FPGA responsável por armazenar a tabela verdade de uma função. Uma LUT que representa a tabela verdade de uma função com *n* entradas consegue representar 2^{2^n} funções lógicas. Em geral, o objetivo dessa abordagem é a redução do número de LUTs necessárias para a implementação do

circuito.

A metodologia *semicustom* é dividida em três etapas: síntese estrutural, síntese lógica e síntese física. Esse trabalho foca nos problemas relacionados à etapa de síntese lógica da metodologia *semicustom*, responsável em transformar a descrição do circuito em uma descrição de nível lógico. O principal objetivo dessa transformação é a otimização das características do circuito, como consumo de energia, atraso, área, etc.

Existem diferentes métodos para a fatoração de funções lógicas. Alguns métodos são eficientes e simples, porém com o aumento da lógica da função necessária para o circuito necessário, esses métodos acabam não se tornando viáveis para execução. A busca de uma fatoração ideal para uma função booleana é um problema *NP-Difícil* (MINTZ; GOLUMBIC, 2005). Por isso, soluções que dependem de busca exaustiva são impraticáveis e é necessário aplicar algoritmos que utilizam soluções heurísticas para o problema, buscando soluções logicamente equivalentes que sejam eficientes ou até mesmo ótimas para o circuito requisitado. O Mapeamento Tecnológico é a última etapa da síntese lógica e tem extrema importância no processo, visto que esta etapa é responsável por definir a estrutura do circuito e determinar suas características em termos de atraso, consumo de energia e área.

O Mapeamento Tecnológico determina qual conjunto de elementos lógicos será utilizado para implementar o circuito requisitado na tecnologia alvo. O foco deste mapeamento é minimizar alguma função objetivo do circuito, sendo essa geralmente a área, consumo ou atraso. Devido a complexidade do problema, os métodos desenvolvidos buscam a otimização de apenas uma característica do circuito. O MT pode ser dividido basicamente em três estágios: *Decomposição*, *Casamento* e *Cobertura*. A seguir essas etapas serão detalhadas demonstrando sua específica importância na abordagem.

2.1.1 Decomposição

A decomposição é a primeira etapa do mapeamento e tem como objetivo transformar a descrição inicial do circuito em uma estrutura que possa ser manipulada pelas etapas seguintes.

Dada uma descrição inicial de um circuito, é necessária a preparação e adaptação do mesmo para que este circuito possa ser encaminhado para as etapas de minimização. Essa transformação do circuito é uma etapa essencial no processo, pois a estrutura de dados escolhida afeta consideravelmente as etapas futuras do mapeamento e, consequentemente, o tamanho final do circuito. Devido a esses motivos é necessária uma representação eficiente que permita uma boa manipulação de sua estrutura em etapas posteriores.

Existem várias maneiras de representar um circuito. Entre as representações mais

utilizadas pode-se citar Diagramas de Decisão Binária, Árvores e Grafos. Para circuitos que podem ser representados como árvores, a solução ótima para área pode ser obtida em tempo polinomial (MARQUES et al., 2007). Infelizmente, a maioria dos circuitos completos não podem ser representados por uma única árvore, sendo necessária a criação de uma floresta, onde cada árvore teria sua solução mapeada individualmente, restringindo as soluções a mínimos locais.

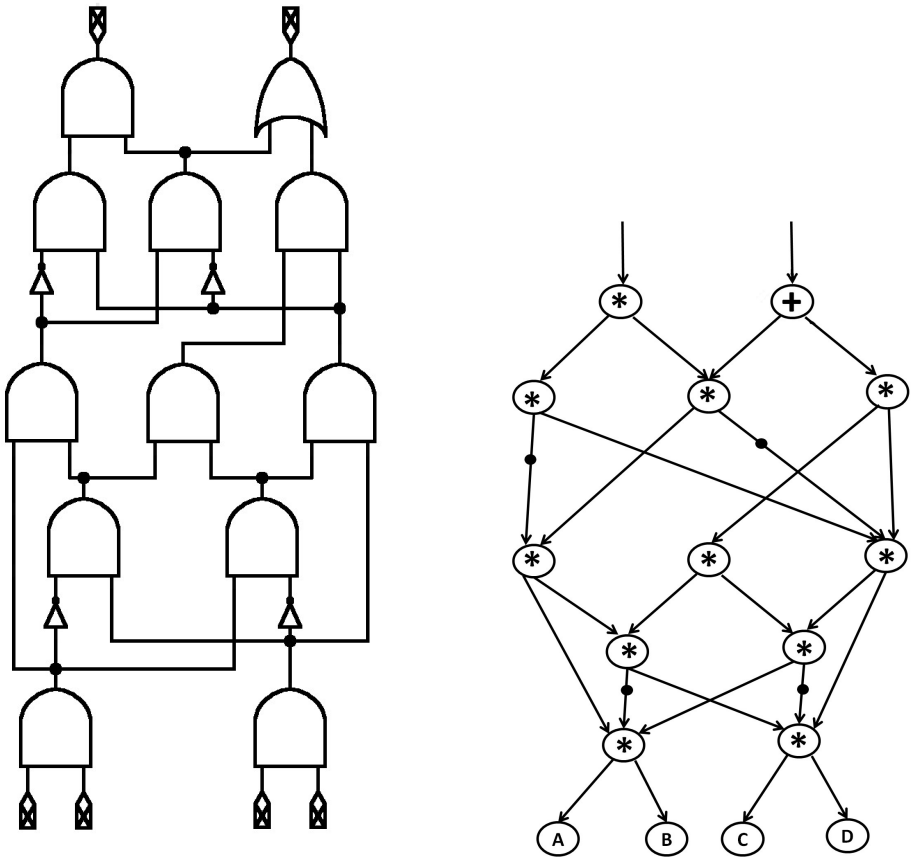
A representação mais comum na síntese logica é feita através de grafos, onde frequentemente é utilizada a representação com Grafos Acíclicos Direcionados (do inglês, *Directed Acyclic Graph*, DAG) (KEUTZER, 1987). Nessa representação, cada porta lógica do circuito é representada por um vértice e as conexões(entradas e saídas de cada porta) são representadas pelas arestas. DAGs podem representar o circuito completo e trabalhar com visibilidade global, sendo assim possível encontrar a solução ótima. O maior problema dessa abordagem é o custo de memória e o esforço computacional necessário para encontrar as soluções, devido a este problema ser considerado *NP-Completo* (MARQUES, 2008). Já existem soluções com DAGs que minimizam o atraso do circuito em tempo polinomial (KUKIMOTO; BRAYTON; SAWKAR, 1998), porém o mesmo não é verdadeiro quando se trata de minimização de área.

A Figura 1 mostra como é realizada a transformação de um circuito para uma representação em DAG. A Figura 1(a) mostra o circuito original e a Figura 1(b) é a reapresentação do mesmo circuito em formato de grafo.

Entre as várias representações de DAGs existentes, vale ressaltar uma especialização que está bem difundida em diversas ferramentas da síntese lógica, devido a sua simplicidade de representação. Esta estrutura, chamada de *And Inverter Graph* (AIG), consegue representar todo o circuito utilizando apenas portas primitivas (AND e Inversor) e facilita as etapas seguintes de mapeamento. Utilizando lógica booleana, é possível converter uma porta OR em portas AND e Inversores, como mostra a Figura 2. A Figura 3 mostra o resultado da conversão do circuito mostrado anteriormente em formato DAG para o formato AIG.

Este trabalho adota a estrutura AIG como base para conversão dos circuitos para grafos, utilizando a representação criada por (AIGER, 2012), chamada de formato AIGER. Esse modelo é bem divulgado no meio acadêmico e possui fácil interpretação, transformando o circuito em uma representação textual binária, onde cada linha representa um vértice ou aresta do AIG. A Figura 4 mostra o circuito apresentado acima convertido para o formato AIGER, junto com sua representação binária.

A conversão do circuito para o formato AIGER é o último passo da etapa de decomposição. O AIG é então encaminhado para a etapa seguinte do mapeamento, chamada de *casamento*.



(a) (b)

Figura 1: Transformação de um circuito em DAG

2.1.2 Casamento

A etapa de casamento (*Matching*), ou etapa de identificação de padrões, consiste na busca e identificação de padrões em subgrafos do circuito com elementos da tecnologia alvo. Devido a necessidade de avaliar todas as partes do AIG em busca de padrões, esta etapa utiliza um alto custo computacional. É necessário a utilização de técnicas otimizadas para a geração e verificação destes padrões.

Casamento utilizando bibliotecas de células

Algumas abordagens da etapa de casamento utilizam bibliotecas de células. Uma biblioteca de células pode ser definida como um conjunto finito de portas lógicas de diferentes tamanhos e topologias que implementam várias funções booleanas. A Tabela 1 apresenta um exemplo de uma biblioteca de células.

Dado uma descrição de um circuito, o objetivo da etapa de casamento nessa abordagem é identificar todos os possíveis subgrafos do circuito e associar as células da biblioteca especificada com subgrafos do AIG, buscando regiões onde a lógica é equivalente. A Figura 5 apresenta a etapa de casamento a partir da descrição de um

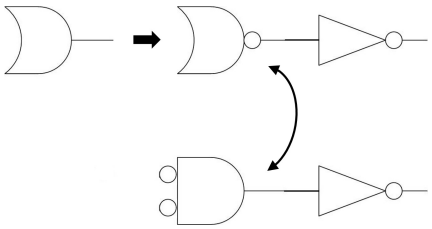
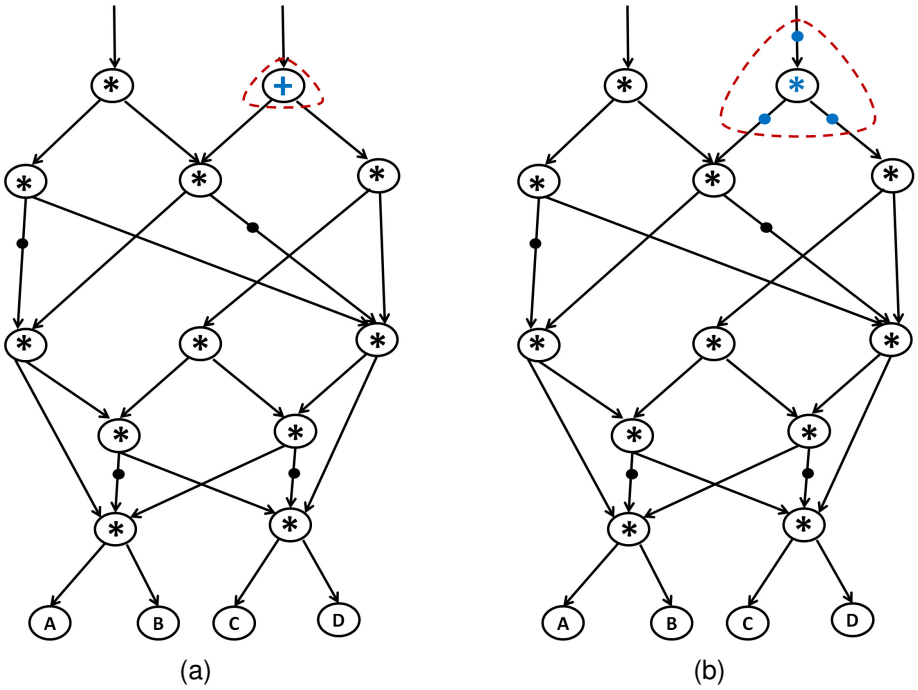


Figura 2: Porta OR convertida em AND e Inversores



(a) (b)

Figura 3: Conversão DAG $\Rightarrow$ AIG

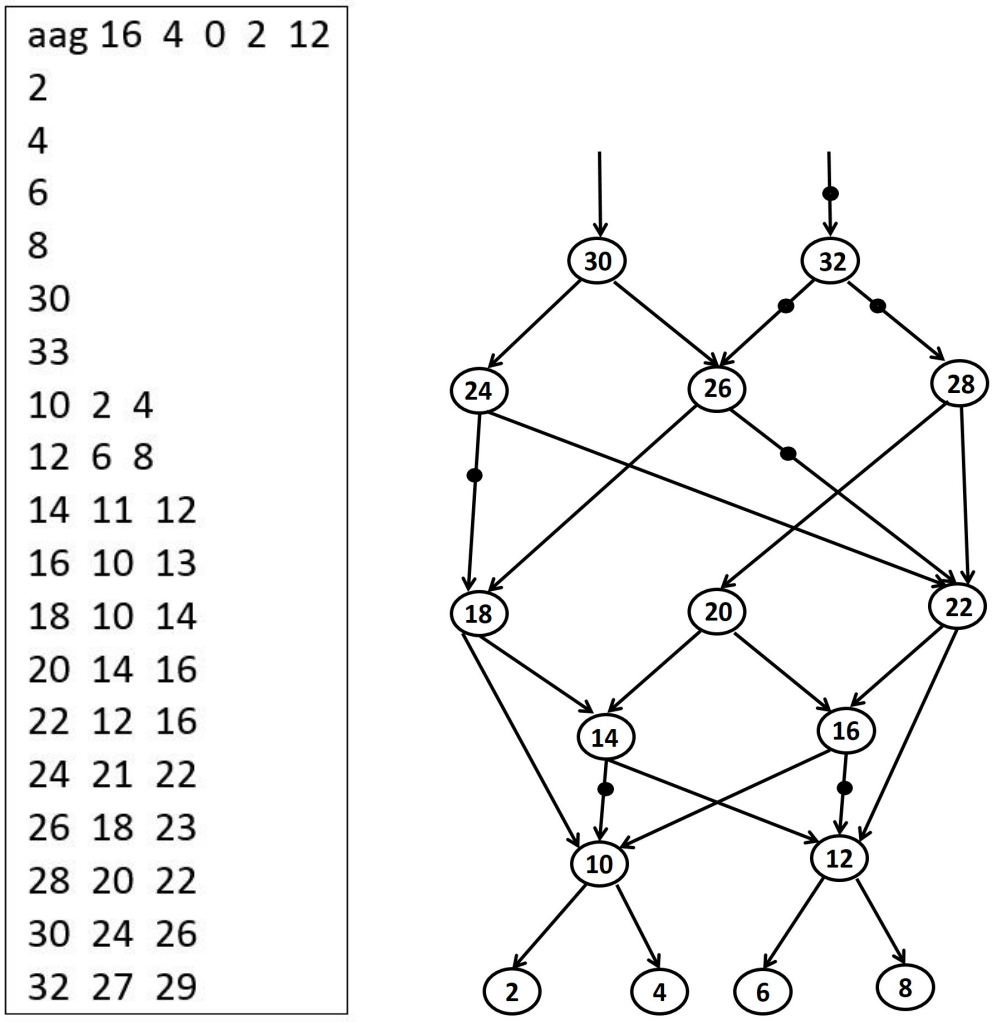
circuito AIG e a biblioteca de células apresentada na Tabela 1.

A maior dificuldade ao se trabalhar com bibliotecas de células está na definição do tamanho da biblioteca. Uma biblioteca grande irá trazer resultados melhores, porém também aumentará o espaço de busca de soluções da etapa de casamento. Por outro lado uma biblioteca com um número muito reduzido de células pode restringir o espaço de busca e retornar soluções ineficientes.

Casamento utilizando matrizes de elementos lógicos reconfiguráveis

A abordagem para FPGAs (*Field Programmable Gate Arrays*) difere-se da anterior por não utilizar bibliotecas de células. Essa abordagem conta com o elemento lógico incluso na tecnologia, chamado de *Look-Up Table* (LUT). A LUT é o componente lógico da FPGA responsável por armazenar a tabela verdade de uma função.

Um dos critérios mais utilizados em abordagens com FPGAs é o número de LUTs



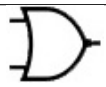
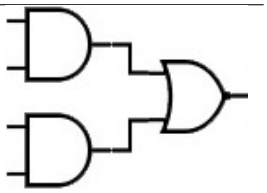
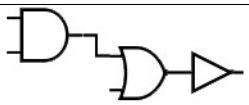
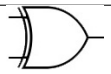
(a) AIGER-ASCII (b) AIGER-Grafo

Figura 4: Representação de um AIG no formato AIGER

necessárias para implementar o circuito completo. Dessa maneira, é possível utilizar algoritmos que enumerem todas as LUTs do circuito para realizar a cobertura na etapa seguinte. Um dos algoritmos mais utilizados na literatura avaliada é chamado de *cortes-K* ou *K-cuts* (CONG; C.; Y., 1999). Esse algoritmo tem como objetivo aplicar uma enumeração de cortes no grafo, identificando todos os subgrafos de cada nodo. Os cortes são limitados pelo número de entradas *"K"* de cada célula.

O particionamento e enumeração dos cortes é extremamente útil na abordagem para FPGAs, pois é possível aplicar a lógica de um *corte-K* diretamente em uma LUT. Assim, é possível simplificar a heurística em função da minimização do número total de *cortes-K* ou *LUTs* utilizados na fase de cobertura do Mapeamento Tecnológico. A Figura 6(a) apresenta os padrões identificados pelo algoritmo de *cortes-K* de um único nodo (nodo 30) do grafo apresentado anteriormente, enquanto a Figura 6(b) mostra o resultado da execução do algoritmo *k-cut* no AIG completo com um valor *"K"* de 3. Os Nodos 2 a 8 não possuem cortes porque eles são entradas do circuito.

Tabela 1: Biblioteca de células

Célula	Custo	Símbolo
INV	2	
AND2	3	
OR2	3	
AO22	4	
AOI21	5	
XOR2	6	

Nessa dissertação a abordagem escolhida foi com base no mapeamento para FPGAs de Manohararajah (MANOHARARAJAH; BROWN; VRANESIC, 2006), utilizando o algoritmo de *cortes-K* para mapear os cortes diretamente para as LUTs da FPGA. O objetivo escolhido para a execução da abordagem foi a minimização da área total do circuito gerado.

2.1.3 Cobertura

A última etapa é chamada de *cobertura* (ou *Covering*), e têm como objetivo selecionar o melhor conjunto de células para a implementação final do circuito. O objetivo desta etapa é encontrar a melhor configuração de elementos da tecnologia alvo que atendam a restrição dada no projeto. Essa restrição é dada por uma função objetivo e o objetivo da etapa de cobertura é a minimização da mesma.

Após realizada a etapa de casamento, são gerados várias combinações de soluções possíveis para uma cobertura do circuito. Essas coberturas entretanto precisam ser cuidadosamente selecionadas para tentar cumprir ao máximo a função objetivo e encontrar quais células farão parte da solução final.

A etapa de cobertura se torna extremamente importante dentro do processo de mapeamento, pois é responsável por encontrar a configuração que melhor corresponda a função custo e que diminua as áreas de redundância (*overlap*) no circuito. Como a alternativa de busca exaustiva da cobertura ótima é impraticável para circuitos grandes, é necessária uma heurística inteligente que selecione a melhor solução

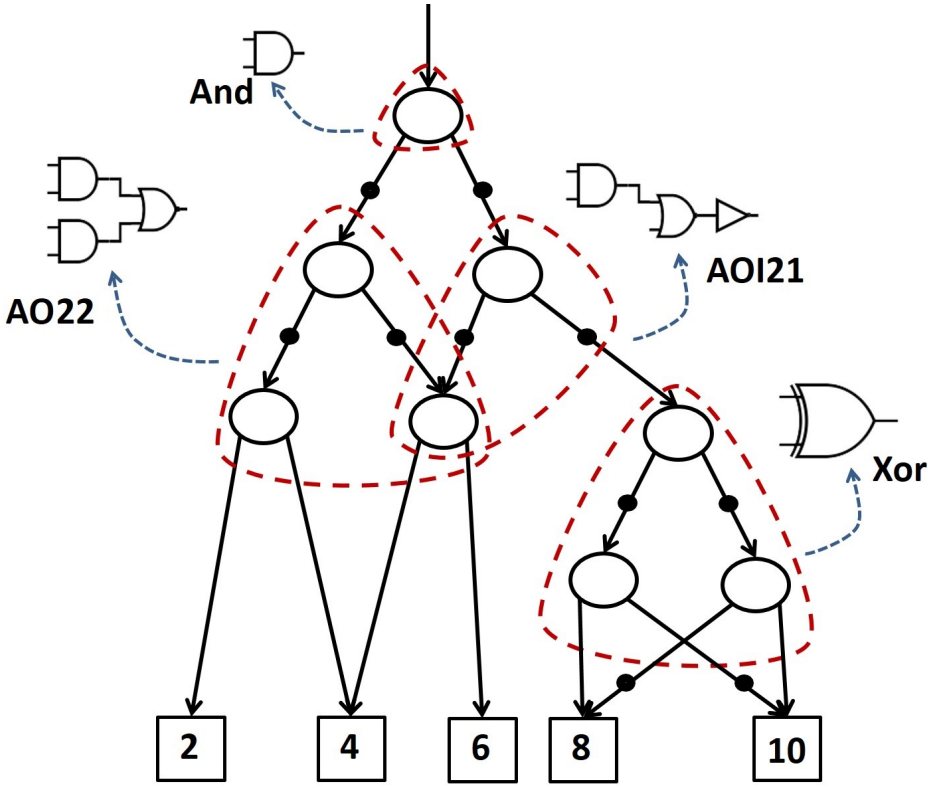
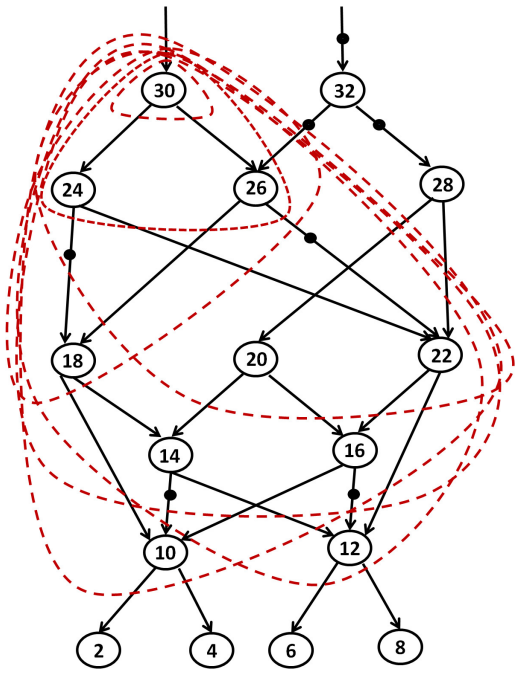


Figura 5: Casamento de um AIG com uma biblioteca de células



(a)

Nodo	Kcut
10	[4,2]
12	[8,6]
14	[8,10,6][4,12,2][12,10]
16	[8,10,6][4,12,2][12,10]
18	[8,10,6][4,12,2][12,10] [14,10][14,4,2]
20	[14,16][10,8,6][4,12,2] [10,12]
22	[10,8,6][8,16,6][4,12,2] [10,12][12,16]
24	[10,8,6][18,12,16][4,12,2] [10,12][14,10,22][18,22]
26	[10,8,6][18,12,16][4,12,2] [10,12][14,10,22][18,22]
28	[22,20][8,10,6][4,12,2] [14,12,16][12,10][12,20,16] [14,22,16]
30	[24,26][8,10,6][18,12,16] [4,12,2][12,10][18,22] [14,10,22]
32	[18,28,22][28,26][8,10,6] [22,20,26][18,22,20] [4,12,2][12,10]

(b)

Figura 6: Algoritmo de *cortes-K* aplicado no AIG

gerada na etapa de casamento.

Cobertura utilizando bibliotecas de células

Em uma abordagem de biblioteca de células, a cobertura tem como objetivo aplicar as células que foram relacionadas a subgrafos do AIG da melhor maneira possível. Cada célula da biblioteca possui um custo para ser utilizada na cobertura do AIG. A cobertura ideal para o circuito é a cobertura cuja combinação de células retorne o custo mínimo. A Figura 7 apresenta uma possível cobertura para o AIG apresentado na Figura 5. Pode-se notar que, para a utilização de algumas células da biblioteca, foi necessária a duplicação de algumas portas primitivas, como por exemplo a porta AND conectada às células AOI21 e AO22.

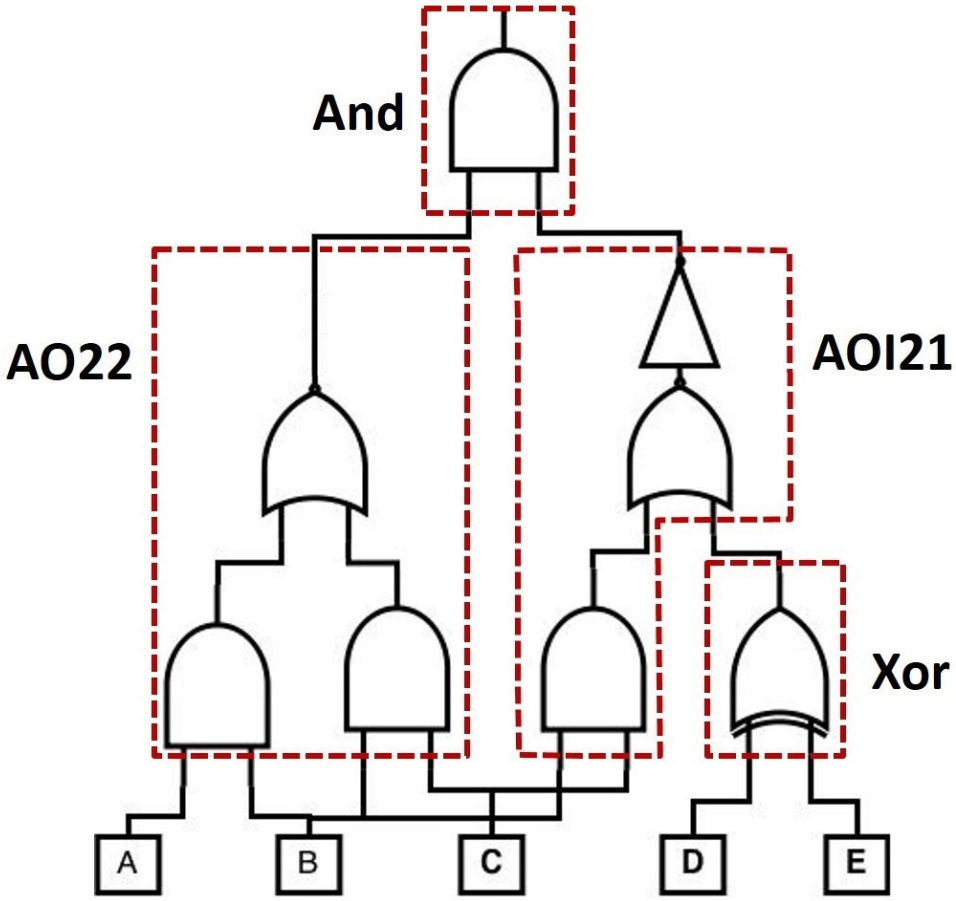


Figura 7: Cobertura do circuito utilizando biblioteca de células

Cobertura utilizando matrizes de elementos lógicos reconfiguráveis

Em uma abordagem para FPGAs, o objetivo da etapa de cobertura é buscar na tabela de *cortes-K* gerada para o AIG quais elementos conseguem aplicar a melhor

cobertura utilizando o menor número possível de *cortes-K*, visto que estes se relacionam diretamente às *Look-Up Tables* da tecnologia em questão.

A Figura 8 apresenta algumas possíveis coberturas do circuito exemplo: a Figura 8(a) tenta minimizar o número de *cortes-K* do circuito; e a Figura 8(b) evita ao máximo sobreposição do circuito.

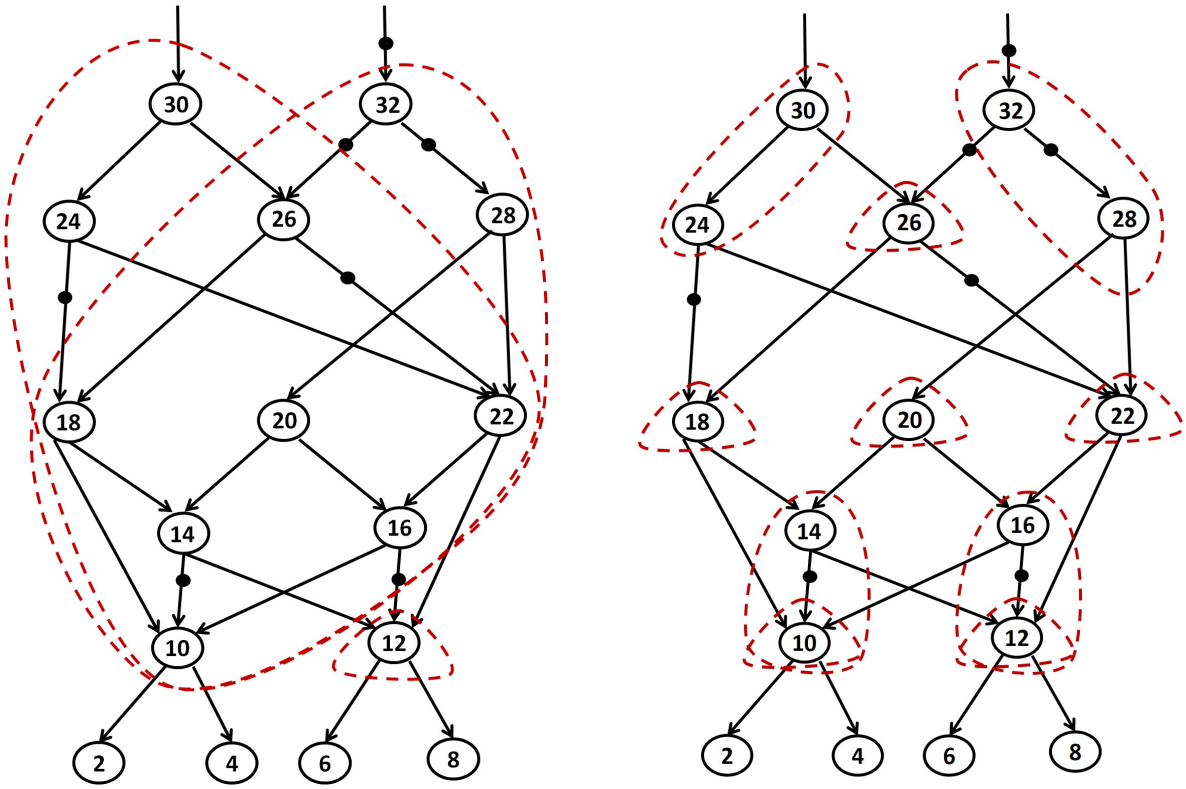


Figura 8: Exemplos de coberturas do AIG

2.2 Têmpera Simulada

A Têmpera Simulada (do inglês, *Simulated Annealing*) (BERTSIMAS; TSITSIKLIS, 1993; VAN LAARHOVEN; AARTS, 1987) é um algoritmo de otimização global utilizado na otimização de funções (AARTS; KORST, 1988; GOODSELL; OLSON, 1990; SVERGUN, 1999). Esta técnica foi inspirada no processo metalúrgico de resfriamento de metais, onde o material é aquecido até altas temperaturas com a intenção de agitar suas moléculas. Após essa agitação, o material começa um processo de resfriamento lento e controlado, para que as moléculas possam se organizar de uma maneira mais eficiente, aumentar a durabilidade e reduzir os defeitos do material.

A metáfora se relaciona de modo que cada configuração diferente da solução no espaço de busca representa um estado diferente do sistema. Ao aumentar a temperatura do sistema, é ampliado o espaço de busca de soluções. À medida que a tempe-

ratura é reduzida, ou seja, que determinado número de iterações passe, o espaço de busca de soluções para o problema é reduzido, fazendo com que o algoritmo restrinja a busca à soluções mais próximas ou melhores que a atual. Quanto menor a temperatura do sistema, mais o algoritmo tende a apresentar um comportamento guloso. No final da execução, quando um número de iterações foi executado ou a temperatura do ambiente alcançou valores próximos à zero, a técnica retorna a melhor solução encontrada, que é geralmente próxima da solução ideal do problema.

O principal diferencial dessa abordagem em comparação à abordagens gulosas é o fato de que a técnica permite que sejam aceitas soluções consideradas inferiores à atual. Essa decisão de escolha é feita com base em um cálculo probabilístico, que leva em consideração o custo da solução atual onde o problema se encontra e o custo da solução inferior.

Além desses fatores, é utilizada também uma variável denominada *temperatura*. Essa variável é diretamente responsável pelo grau de agitação da solução e é através dela que se determina a frequência de aceitação dessas soluções. Quanto maior for a temperatura, maiores são as chances de que a solução considerada inferior seja aceita. Por outro lado, quando a temperatura do ambiente é reduzida, o algoritmo começa a apresentar uma estratégia gulosa, aceitando somente soluções consideradas boas.

2.2.1 Inicialização do algoritmo

A técnica de Têmpera Simulada começa sua busca a partir de um estado inicial. A forma mais comum de inicialização é a utilização de um estado inicial aleatorizado, para garantir que o algoritmo não irá convergir para soluções ótimas locais. Após essa inicialização, o TS entra em seu laço principal, que é responsável por gerar soluções vizinhas à solução atual.

Tomando por exemplo o problema do caixeiro viajante (DORIGO; GAMBARDELLA, 1997; GREFENSTETTE et al., 1985), onde o objetivo do problema é a busca por uma rota que passe por toda as cidades do sistema e retorne à cidade original com um custo mínimo, uma solução inicial seria a geração de uma rota aleatória, sem dar importância à proximidade de qualquer duas cidades do sistema.

Existem algumas implementações do TS que optam por não iniciar o algoritmo a partir de uma solução inicial aleatória. Isso se deve ao fato do problema onde o algoritmo está inserido possuir um espaço de busca de soluções muito grande, e uma inicialização aleatória neste tipo de problema pode gerar soluções consideradas péssimas. Dessa maneira o TS precisa de muitas iterações, e consequentemente de muito custo computacional, apenas para conseguir convergir para soluções minimamente aceitáveis para o problema.

Outro ponto importante a considerar é que a utilização de inicialização aleatória

pode levar o algoritmo a se restringir a mínimos locais. No momento ao qual o ambiente encontra-se em altas temperaturas, onde o algoritmo deveria estar buscando escapar dos mínimos locais, o mesmo encontra-se buscando melhorar uma solução ineficiente para o problema. Com o passar das iterações, a temperatura do ambiente será reduzida e o algoritmo tende a adotar uma abordagem gulosa, se restringindo a aceitar soluções apenas melhores que a atual.

Assim, algumas abordagens realizam a inicialização do problema com a aplicação de uma solução inicial já semi-otimizada, com o objetivo de direcionar o algoritmo para um espaço de soluções favoráveis e tentar vasculhar o espaço de busca ao redor da solução para encontrar uma solução superior. No problema do caixeiro viajante, pode-se por exemplo tentar utilizar alguma heurística para determinar uma solução inicial mais favorável do que uma rota aleatória (PERTTUNEN, 1994; LIU, 2010). Apesar dessa abordagem se mostrar eficiente e evitar o consumo de custo computacional, ela pode direcionar o algoritmo na direção errada, convergindo o TS para um ótimo local. Assim é necessário um estudo cauteloso do tipo de inicialização do algoritmo de TS no problema ao qual ele será aplicado.

2.2.2 Busca de soluções vizinhas e probabilidade de aceitação

Uma solução considerada “vizinha” é uma solução do problema que foi gerada a partir de uma alteração na solução atual do problema. No problema do caixeiro viajante, uma solução vizinha à uma rota é, por exemplo, a troca de ordem de 2 cidades da rota atual. A busca por soluções vizinhas é fundamental para a otimização do problema, já que a solução final será resultado de uma sucessão de trocas do estado inicial até o estado final encontrado.

Uma solução vizinha, no entanto, não é garantidamente melhor que a solução atual. Se este fosse o caso, um algoritmo com uma abordagem gulosa poderia facilmente encontrar a solução ótima do problema, simplesmente fazendo trocas de vizinho para vizinho até chegar à um ponto onde nenhum vizinho possui uma solução melhor. Esse ponto encontrado por algoritmos gulosos é chamado de *ótimo local*, enquanto a melhor solução existente para o problema é chamada de *ótimo global*.

Em algoritmos como o Têmpera Simulada, existe a possibilidade de navegação para estados vizinhos inferiores ao estado atual. Essa troca serve justamente para que o algoritmo não se prenda aos mínimos locais encontrados. Cada vez que um novo vizinho é criado, é utilizado uma avaliação de sua solução a partir de uma função custo, específica para o problema. A partir deste custo, é avaliada a variação Δ do sistema, dado pela Equação 1. Quando $\Delta < 0$, isto significa que ocorreu uma redução de energia no problema, implicando que a nova solução encontrada é melhor do que a atual. Nesta situação, a técnica aceita a solução gerada e a solução vizinha passa a ser a solução atual.

Quando são encontradas soluções consideradas inferiores à atual, ou seja, $\Delta > 0$, a técnica executa um procedimento que determina a probabilidade de aceitar a solução inferior. Esse procedimento é geralmente dado pela Equação 2. Quando a temperatura do ambiente encontra-se elevada, o valor resultante deste cálculo tende a ser alto, possibilitando a troca para estados inferiores com maior frequência. À medida que o algoritmo é executado e a temperatura do ambiente decai, o algoritmo começa a rejeitar com mais frequência soluções inferiores e tende a uma estratégia gulosa.

$$\Delta = custo(estado\_vizinho) - custo(estado\_atual) \tag{1}$$

$$e^{(-\Delta/Temperatura)} \tag{2}$$

Tipicamente, o algoritmo repete o passo descrito até que um número máximo de iterações seja executado ou quando a temperatura do ambiente chega a um valor muito próximo de zero. A Figura 9 apresenta um fluxograma do funcionamento tradicional do algoritmo de Têmpera Simulada.

2.3 Considerações Finais

Neste capítulo foram apresentados os principais conceitos relacionados ao Mapeamento Tecnológico, mostrando como é feita a divisão de cada etapa do seu funcionamento. Adicionalmente, foi apresentada a técnica de Têmpera Simulada, que apresenta resultados favoráveis em diversas aplicações na literatura. Utilizando esses conhecimentos foi possível realizar a integração do TS no Mapeamento Tecnológico para a criação de uma nova abordagem para o problema. No capítulo seguinte será apresentado uma revisão das principais técnicas de Mapeamento Tecnológico existentes na literatura.

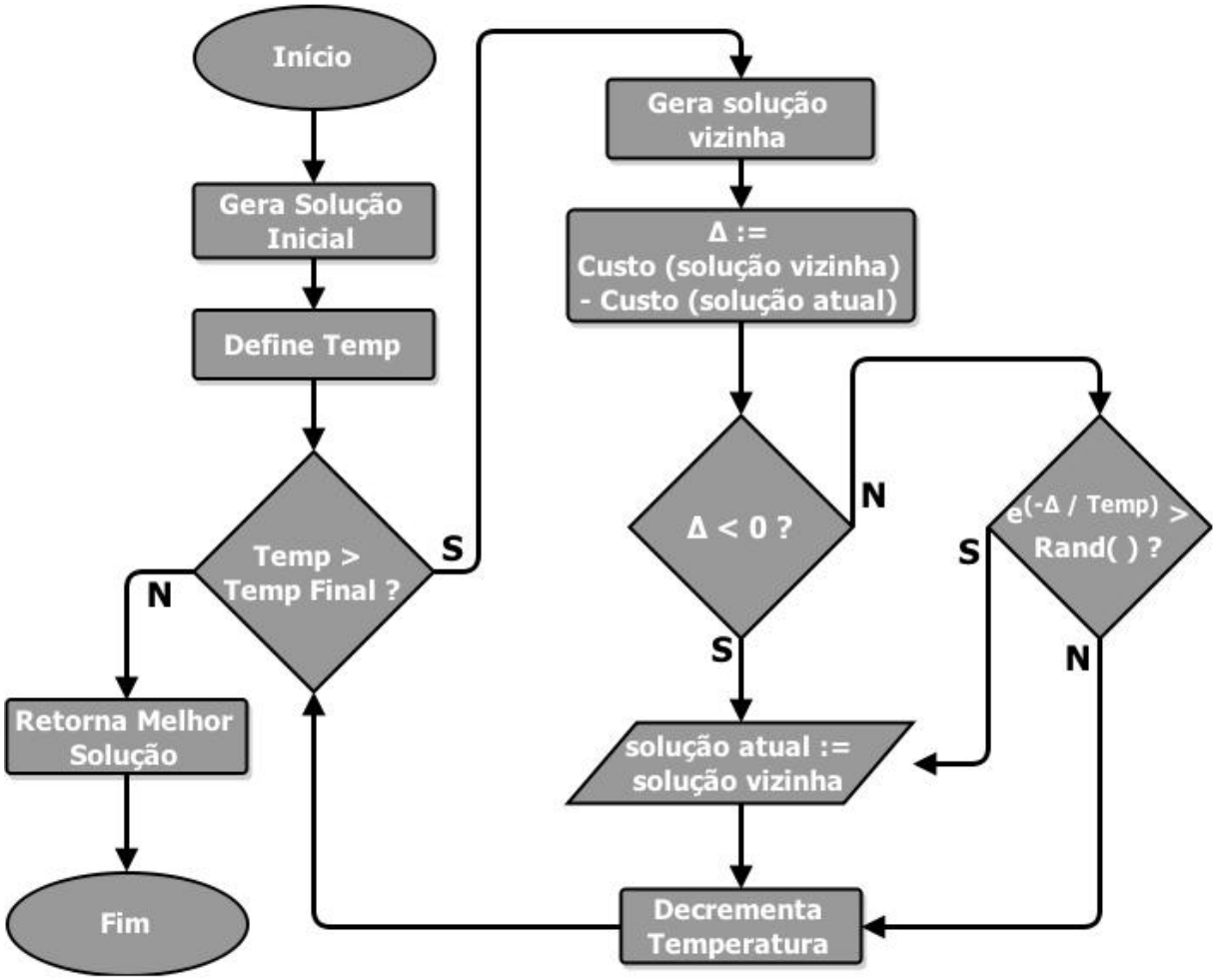


Figura 9: Fluxograma do Têmpera Simulada

3 TRABALHOS RELACIONADOS

Neste capítulo será apresentada uma revisão bibliográfica de alguns dos principais métodos de Mapeamento Tecnológico da literatura, bem como algumas das abordagens que utilizam técnicas de inteligência Artificial.

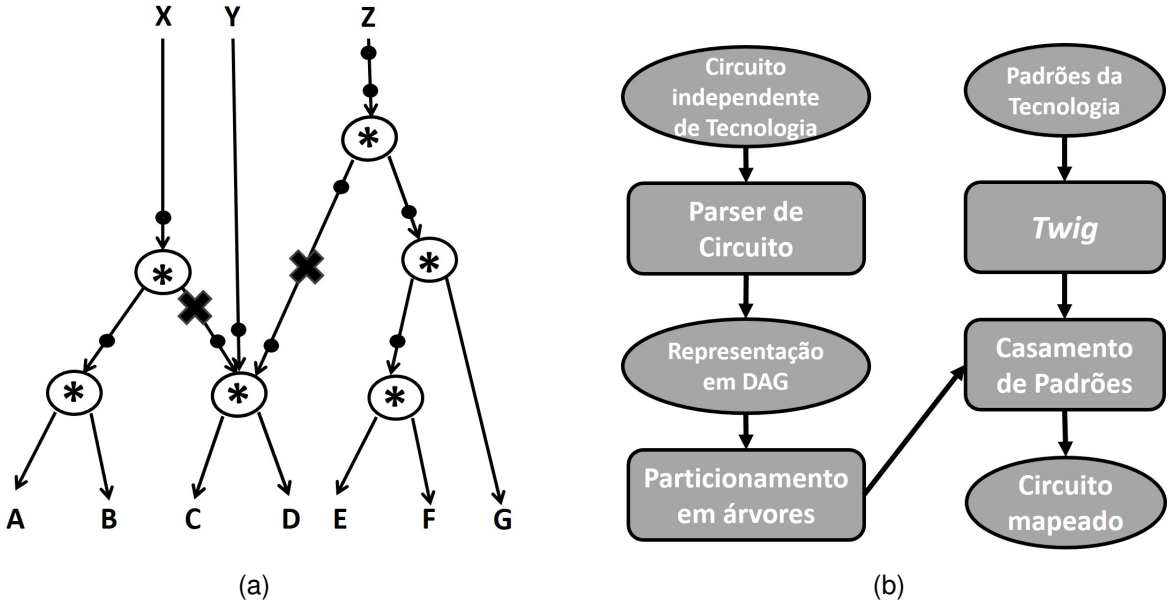
3.1 Dagon

A técnica de Keutzer (KEUTZER, 1987) utiliza ferramentas de otimização de compiladores para o casamento de padrões no Mapeamento Tecnológico. O processo de realizar o casamento de uma descrição de grafo independente de tecnologia com uma biblioteca de células é semelhante a realizar o casamento de uma representação em grafo de um programa de computador com os padrões do conjunto de instruções do mesmo.

O circuito é inicialmente mapeado para um DAG e em seguida é decomposto em uma floresta de árvores. Depois desse particionamento, o DAGON faz uma busca em cada uma das árvores, encontrando subgrafos logicamente equivalente às células na biblioteca. Para essa busca Keutzer utiliza a ferramenta *Twig* (TJIANG, 1986), utilizada em geradores de código para compiladores de linguagens de programação. Após a ferramenta encontrar os casamentos das células, o algoritmo de cobertura procura a melhor combinação de células para a solução final. A Figura 10 mostra o particionamento do DAG em árvores e o fluxo da abordagem de Keutzer.

3.2 WaveFront

O algoritmo *WaveFront*, proposto por Stok (STOK; IYER; SULLIVAN, 1999) propõe uma abordagem para otimização de atraso utilizando DAGs. A principal característica desta abordagem é a utilização de uma janela, chamada de *WaveFront*, que tem como objetivo delimitar a área do circuito a ser decomposta. Essa janela limita o espaço onde será aplicado o mapeamento a cada momento, evitando consumos elevados de alocação de memória e custo computacional. A maior barreira desta abordagem é o tamanho do *WaveFront*. Uma janela muito pequena fará com que o algoritmo se limite



(a) (b)

Figura 10: Particionamento do DAG e Fluxograma do DAGON

a mínimos locais, gerando apenas soluções sub-ótimas. Em contrapartida, uma janela muito grande gera problemas de memória que podem tornar o algoritmo impraticável para circuitos grandes.

3.3 ELIS - *Environment for Logic Synthesis*

O algoritmo proposto por Correia (CORREIA, 2004), foi incorporado a Ferramenta ELIS. Esse algoritmo conta com o particionamento do DAG em múltiplas árvores. Após o particionamento, a técnica aplica o Teorema de DeMorgan em cada árvore da floresta gerada, iniciando sua aplicação na raiz da árvore em direção às folhas. A aplicação do Teorema de DeMorgan faz com que ocorra um reposicionamento de inversores nos nodos intermediários do subgrafo, os enviando em direção aos nodos folha da árvore. A seguir, os nodos com operadores equivalentes são agrupados. A estratégia de Correia tenta minimizar os problemas causados por dependências estruturais no circuito.

A Figura 11 demonstra como acontece o agrupamento de operadores. Após o agrupamento, as árvores geradas são atravessadas através de uma *Depth-First Search* (DFS), realizando operações de cálculo de custo de cada nodo. Se o custo calculado de um nodo ultrapassa as restrições do algoritmo de Correia, esse nodo é desconectado da árvore onde se encontra e se torna uma nova árvore. Esse processo é repetido até que todos os nodos da floresta obedeçam a restrição. Por fim, a descrição das árvores geradas é a descrição das células que serão necessárias para a implementação do circuito final. Embora o algoritmo retorne resultados favoráveis, a utilização de árvores no mapeamento restringe a visualização do circuito global, o que

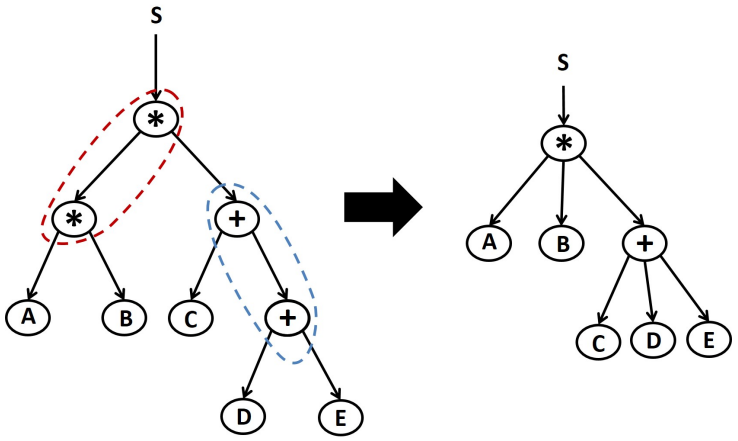


Figura 11: Agrupamento de nodos da ferramenta ELIS

pode gerar apenas soluções sub-ótimas para o circuito.

3.4 VIRMA - *Virtual Technology Mapping Tool*

A ferramenta desenvolvida por Marques (MARQUES, 2008) apresenta duas abordagens para o Mapeamento Tecnológico baseadas em DAGs. As duas abordagens propostas combinam o método de computação Booleana do número de transistores em série apresentado por Schneider (SCHNEIDER et al., 2005) com os algoritmos considerados estado-da-arte desenvolvidos por Stok (STOK; IYER; SULLIVAN, 1999) e Mishchenko (MISHCHENKO et al., 2005).

Os algoritmos desenvolvidos buscam reduzir o número de transistores em série do caminho mais longo do circuito. Assim, cada célula é implementada por uma rede de transistores. Essa rede obedece a uma restrição que limita o número máximo de transistores em série, que são calculados de forma Booleana.

A primeira abordagem da ferramenta VIRMA é chamada de *VIRMA-WF*, derivada do algoritmo *WaveFront* apresentado por Stok (STOK; IYER; SULLIVAN, 1999), enquanto a segunda abordagem foi adaptada a partir da técnica apresentada por Mishchenko (MISHCHENKO et al., 2005) que utiliza o algoritmo de *cortes-K* desenvolvido por Cong (CONG; C.; Y., 1999), onde cada *corte-K* é equivalente a uma célula da biblioteca.

3.5 Manohararajah

A abordagem de Manohararajah (MANOHARARAJAH; BROWN; VRANESIC, 2006) utiliza a técnica de *cortes-K* para mapear os cortes diretamente para as LUTs da FPGA. Após a enumeração de todos os *cortes-K* de um nodo é utilizado um algoritmo que calcula os custos desses cortes, chamado de *Area Flow*.

Para calcular o custo de cada corte de um nodo, o *Area Flow* leva em consideração o custo da área acumulada das entradas do corte até a saída. Depois que o custo de cada *corte-K* é contabilizado, o algoritmo seleciona os cortes que retornaram os melhores resultados para a cobertura do circuito. Vale notar que a escolha de um melhor corte a partir dessa tabela não necessariamente irá produzir uma cobertura ótima, já que esta é uma abordagem gulosa.

3.6 Mishchenko

Com base na técnica de *Area Flow* desenvolvida por Manohararajah, Mishchenko (MISHCHENKO; CHATTERJEE; BRAYTON, 2007) desenvolveu uma técnica otimizada que associa heurísticas ao método, buscando evitar esforço computacional desnecessário.

O principal ponto desta técnica é a utilização de uma heurística para a computação dos *cortes-K* de um grafo. Nas abordagens tradicionais da literatura, o maior esforço computacional é dado na enumeração de todos os *cortes-K* do circuito. Essa enumeração aumenta consideravelmente à medida que o valor de *k* é incrementado, dificultando o processo. A abordagem tradicional das técnicas é a utilização de "podas" na enumeração, que podem limitar o espaço de soluções.

Mischenko tenta não efetuar "podas" na enumeração mantendo um histórico de cortes do circuito. Assim, o algoritmo desenvolvido consegue evitar cortes redundantes no circuito. Com esta técnica é possível realizar o mapeamento utilizando coberturas com *cortes-K* com um valor de *k* relativamente grande.

A ferramenta mais atual na área síntese lógica é chamada de ABC (ABC, 2014). Esta ferramenta possui diversos métodos considerados estado-da-arte na área de MT, assim como outros algoritmos de síntese lógica que são capaz de realizar balanceamentos no grafo e otimizações independentes de tecnologia. O algoritmo desenvolvido por Mishchenko é um dos algoritmos de MT implementados dentro da ferramenta.

O ABC utiliza uma estrutura específica de DAGs, denominada *And-Inverter Graph* (AIG), que decompõe a lógica do circuito, o reduzindo a operadores primitivos AND e Inversores. Essa estrutura permite que a ferramenta evite problemas de dependências estruturais e consegue encontrar soluções eficientes e rápidas para o mapeamento de um circuito e é capaz de lidar com circuitos de grande porte.

A ferramenta ABC é considerada a ferramenta mais atual e completa na área de mapeamento Tecnológico. Desta forma, o ABC foi utilizado como referência neste trabalho para a avaliação e comparação dos experimentos executados.

3.7 Inteligência Artificial no Mapeamento Tecnológico

Devido às técnicas de otimização utilizando Inteligência Artificial conseguirem produzir resultados favoráveis em diversas áreas na literatura (GREFENSTETTE et al., 1985; RAHMAT-SAMII; MICHIELSSEN, 1999; POND et al., 2006), a utilização dessas técnicas no Mapeamento Tecnológico torna-se uma alternativa interessante. A seguir serão mostradas abordagens encontradas na literatura que utilizam IA, mais precisamente a técnica de algoritmos genéticos (GOLDBERG, 1989; MELANIE, 1999; DAVIS, 1991) para a otimização de circuitos integrados.

Algoritmos Genéticos (AG) são algoritmos criados com base no conceito de adaptação e evolução descritos por Darwin. A teoria de Darwin relatava que um indivíduo evoluía com base na sua adaptação ao meio em qual estava inserido. Indivíduos mais adaptados ao meio possuíam uma maior chance de sobrevivência, enquanto os indivíduos menos adaptados tenderiam a desaparecer da população e se extinguir. Essas características que permitiam a sobrevivência de um indivíduo seriam transmitidas aos descendentes dos indivíduos sobreviventes. Esse processo, chamado de Seleção Natural é é um dos fundamentos principais dos Algoritmos Genéticos(AGs).

O objetivo dos algoritmos genéticos é trabalhar com uma população, onde cada indivíduo desta população representa uma solução para o problema em questão. Os indivíduos dessa solução seriam classificados utilizando-se de uma função de avaliação, que tem o propósito de classificar quais indivíduos da população são mais evoluídos e possuem maior chance de produzirem soluções satisfatórias através de técnicas de reprodução e mutação.

3.7.1 GAFPGA - *Genetic Algorithm for FPGA Technology Mapping*

A técnica de GAFPGA proposta por Kommu (KOMMU; POMERANZ, 1993), apresenta um método baseado em algoritmos genéticos para o mapeamento de *Field Programable Gate Arrays* (FPGAs). Essa abordagem utiliza o algoritmo de *cortes-K* desenvolvido por Cong (CONG; C.; Y., 1999) para realizar a minimização de área (número de LUTs utilizadas na cobertura do circuito).

A abordagem proposta transforma uma cobertura para o problema em duas *strings*. A primeira *string* armazena os pontos de segmentação das saídas das portas, enquanto a segunda *string* representa a maneira que uma porta lógica foi decomposta no circuito. A partir desses indivíduos, foi aplicado as técnicas de Crossover e Mutação, amplamente conhecidas na área de AG. Após executadas as técnicas, é necessária a validação do indivíduo para garantir que o indivíduo gerado seja um indivíduo válido, ou seja, que a lógica do circuito gerado não foi alterada. Caso o indivíduo gerado seja inválido, o indivíduo em questão passava por uma etapa de alteração para ser

transformado em um indivíduo válido.

Devido a complexidade do problema de mapeamento e o tamanho de circuitos de teste, não é possível trabalhar com uma população de indivíduos muito grande. Kommu utilizou uma população de apenas 4 indivíduos na sua abordagem. Apesar do tamanho da população ser pequena, a população conseguiu manter sua diversidade devido à técnica de validação implementada.

3.7.2 MOGAMAP

O algoritmo MOGAMAP (SOUZA; SILVA-FILHO, 2013), desenvolvido por Sousa, é a abordagem mais recente que utiliza Inteligência Artificial no mapeamento genético. Essa abordagem também busca tratar o problema de Mapeamento Tecnológico para FPGAs.

A técnica implementada consiste de 4 etapas. Na primeira etapa ocorre uma análise do AIG do circuito em busca de informações de cada nodo do grafo, com a intenção de orientar a estimativa de consumo. A enumeração de *cortes-K* do circuito é realizada na segunda etapa do algoritmo, enquanto que na terceira etapa, é executada uma modificação do algoritmo de *Area Flow* (MANOHARARAJAH; BROWN; VRANESIC, 2006) de Manohararajah para computar os custos do circuito. Finalmente, na quarta etapa, a cobertura obtida é aplicada na população de indivíduos para que possam ser realizadas as operações de seleção, casamento e mutação do AG.

3.8 Considerações Finais

Encontrar a melhor solução possível para o Mapeamento Tecnológico é um problema de otimização combinatória. A solução ideal para a resolução deste problema é realização de uma busca exaustiva em todo o espaço de soluções existente. Porém, devido à enorme quantidade de soluções do problema, essa solução torna-se impraticável. É necessário então a utilização de técnicas dependentes de heurísticas.

Este capítulo apresentou os principais algoritmos e ferramentas que tentam minimizar os atributos de circuitos integrados. Enquanto algumas abordagens tentam minimizar o atraso do circuito, as abordagens mais atuais buscam reduzir ao máximo a área que esse circuito irá ocupar no projeto final. A Tabela 2 apresenta as principais características de cada método apresentado, mostrando o tipo de descrição de cada método, o foco de minimização a estratégia de cobertura utilizada para atingir os resultados.

Entre as abordagens estudadas, foram encontradas abordagens que utilizam a técnica de algoritmos genéticos da Inteligência Artificial na a etapa de Mapeamento Tecnológico. Essas abordagens, no entanto, já se encontram ultrapassadas e não conseguem superar a ferramenta estado-da-arte. A seguir, será apresentada a solução

Tabela 2: Tabela comparativa dos principais métodos de mapeamento

Método	**Descrição Sujeito**	**Foco de Minimização**	**Abordagem**
DAGON (1987)	Árvore	Atraso	Biblioteca Estática
WaveFront (1999)	DAG	Atraso	Biblioteca Estática
ELIS (2004)	Árvore	Área	Biblioteca Dinâmica
VIRMA (2008)	DAG	Atraso	Biblioteca Dinâmica
MANOHARARAJAH (2006)	DAG	Área	FPGA-LUT
MISHCHENKO (2007)	DAG	Área	FPGA-LUT
GAFPGA (1993)	DAG	Área	FPGA-LUT
MOGAMAP (2013)	DAG	Área	FPGA-LUT
Abordagem Proposta	**DAG**	**Área**	**FPGA-LUT**

proposta neste trabalho para resolver o problema do Mapeamento Tecnológico.

4 ABORDAGEM PROPOSTA

Analisando as abordagens vistas no capítulo anterior, é possível verificar que embora existam soluções que encontrem bons resultados para o Mapeamento Tecnológico, ainda existe espaço para a descoberta de novas técnicas e abordagens para o problema. Tendo em vista a necessidade de desenvolvimento de técnicas de mapeamento mais eficientes para a ferramenta FlexMap e a utilização de técnicas baseadas em algoritmos aproximativos de otimização, uma abordagem para o MT baseada em Têmpera Simulada está sendo proposta neste trabalho. A Abordagem, nomeada *FlexMap-TS* será apresentada no capítulo a seguir.

Não foram encontrados trabalhos na literatura que utilizem a técnica de Têmpera Simulada no processo de Mapeamento Tecnológico. Desta forma, este trabalho propõe uma nova abordagem para o problema, sendo o objetivo principal a redução do número de elementos utilizados para implementar o circuito, visando a redução da área.

4.1 Têmpera Simulada aplicada no Mapeamento Tecnológico

A abordagem proposta neste trabalho utiliza uma adaptação do algoritmo de Têmpera Simulada no processo de minimização de área de circuitos FPGAs no Mapeamento Tecnológico. A escolha desta abordagem se deu devido ao fato de que o algoritmo de TS possui um bom desempenho em problemas de otimização onde é necessário fugir de mínimos locais em busca de uma solução ótima global.

Devido ao foco do trabalho ser a minimização de circuitos para FPGAs, a função custo escolhida para avaliar a eficiência das soluções foi o somatório de todos os *cortes-K* utilizados em uma cobertura do circuito, já que cada *corte-K* mapeado pode ser diretamente relacionado a uma *Look-Up Table* (LUT) do FPGA.

4.1.1 Cobertura Inicial e Processo de Limpeza

A execução do algoritmo começa com a leitura do circuito. A etapa de decomposição converte o circuito em um grafo AIG no formato AIGER (AIGER, 2012).

Após convertido, o algoritmo dá início a etapa de casamento, onde é executado o algoritmo de *cortes-K* (CONG; C.; Y., 1999), que realiza a enumeração de todos os cortes de cada nodo para a utilização na etapa de cobertura do problema.

Algoritmo 1: Pseudo-Algoritmo da cobertura inicial

Input: aig,cortesK
Output: coberturaInicial

```
1 coberturaInicial ← new cobertura( );
2 foreach Nodo nodo in aig do
3     if ( nodo != input ) then
4         nodoCortes ← retorna_cortesK(nodo,cortesK);
5         coberturaNodo ← corte_Aleatorio(nodoCortes);
6         coberturaInicial.add(coberturaNodo);
7     end
8 end
```

A cobertura inicial aleatória pode ser vista no Algoritmo 1. Para cada nodo do AIG que não seja uma entrada do circuito, é selecionado um *corte-K* aleatoriamente da tabela de cortes gerada na etapa de casamento. Este corte é adicionado à cobertura, cobrindo o nodo e possivelmente outros nodos que são entradas a esse. Esse processo é executado em todos os nodos do AIG, independente da ocorrência da sobreposição de *cortes-K* na cobertura gerada. A Figura 12(a) mostra um exemplo de geração de uma cobertura inicial aleatória no grafo exemplo.

Depois que todos os nodos foram cobertos é realizada uma etapa denominada de *limpeza*. A limpeza consiste na realização de uma busca em amplitude (do inglês *Breadth First Search*, BFS) no AIG, das saídas do grafo em direção às entradas. Essa busca tem como objetivo eliminar áreas que tiveram coberturas desnecessárias.

O Algoritmo 2 exemplifica o funcionamento da função de limpeza. Para cada nodo "Saída" do AIG, o seu *corte-K* é adicionado na cobertura a ser gerada (linha 6). A seguir, são verificadas quais são as entradas do *corte-K* adicionado. Se as entradas desse nodo não forem as entradas do circuito, elas são adicionadas à lista de nodos "Saída", para ter seu *corte-K* inserido na cobertura a ser gerada (linha 9). Este processo é repetido até que o algoritmo percorra todos os nodos até a entrada do circuito. A Figura 12(b) mostra a cobertura do estado inicial aleatório depois do processo de limpeza.

Algoritmo 2: Pseudo-Algoritmo função de limpeza

Input: aig,cortesKcobertura
Output: cobertura

```
1  bfsSaidas ← new saidas(aig);
2  entradasAIG ← new entradas(aig);
3  while ( bfsSaidas ! = nil ) do
4  |   nodoAtual ← bfsSaidas.pop();
5  |   corteNodo ← cortesKCobertura(nodoAtual);
6  |   cobertura.add(corteNodo);
7  |   foreach entrada in entradas(corteNodo) do
8  |   |   if ( entrada ! = entradasAIG ) then
9  |   |   |   nodoSaidas.add(entrada);
10 |   |   end
11 |   end
12 end
```

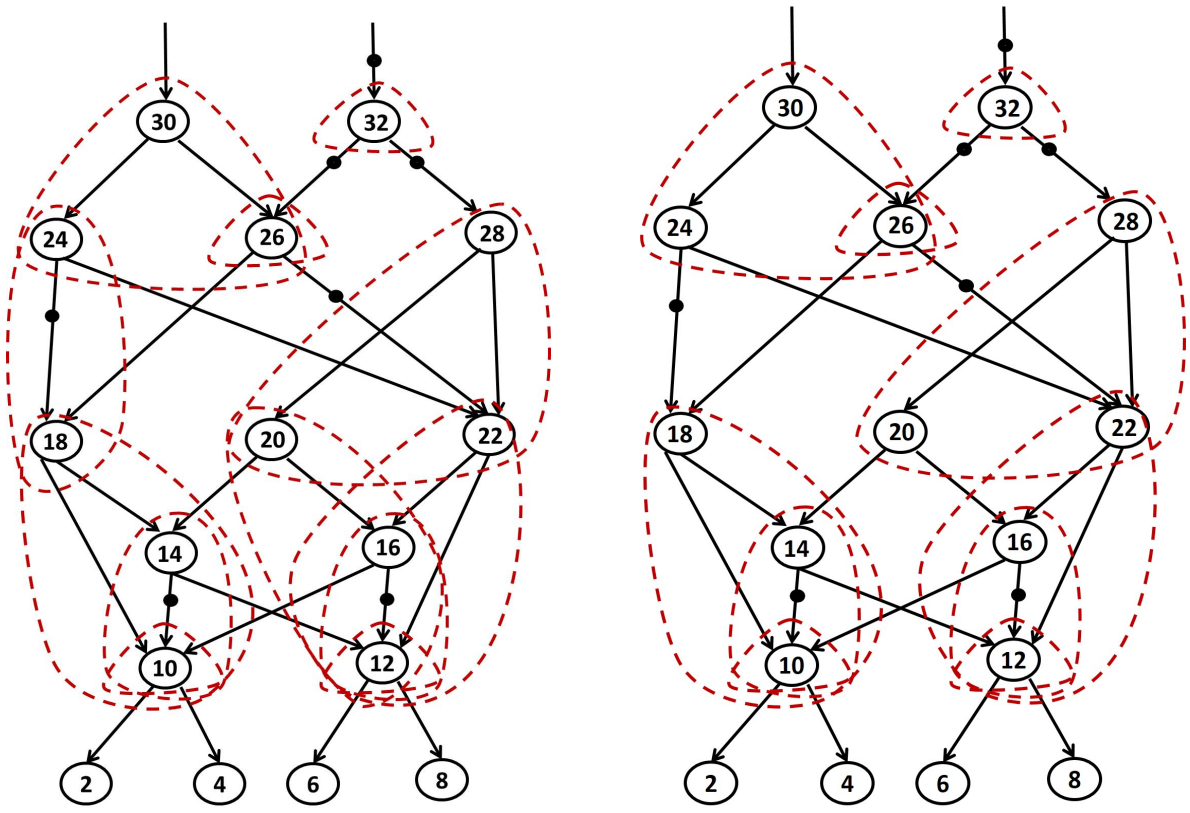


Figura 12: Geração da cobertura inicial e processo de limpeza

4.1.2 Iteração do Têmpera Simulada

Após a construção da cobertura inicial do AIG, o processo principal do Têmpera Simulada inicia. A Figura 13 mostra a execução de uma iteração do Flexmap-TS. A cada iteração do algoritmo, um nodo do grafo é escolhido de forma aleatória parar rea-

lizar a modificação do *corte-K*. Após selecionado (Figura 13(a)), é necessário acessar a tabela de *cortes-K* com a lista de cortes do nodo em questão. O corte substituto é escolhido aleatoriamente dessa lista e inserido no grafo (Figura 13(b)), substituindo o corte original.

Depois da substituição do corte, é necessário refazer toda a cobertura do grafo, pois a substituição de um corte pode gerar dependências e coberturas desnecessárias. Infelizmente essa cobertura não pode ser realizada localmente, originando do nodo em questão até as entradas do circuito. É necessário realizar novamente o processo de limpeza do AIG, aplicando uma busca em amplitude partindo das saídas do circuito com destino às entradas. O resultado deste procedimento pode ser visualizado na Figura 13(c).

Caso a nova cobertura encontrada seja superior à cobertura atual, é realizada a substituição e a solução antiga é descartada. Se a cobertura gerada não for superior à atual, o Flexmap-TS realiza um cálculo de probabilidade baseado no custo de cada uma das coberturas e na variável temperatura para decidir se a solução gerada deve ou não ser aceita. O cálculo dessa probabilidade de aceitação é geralmente dado pela Equação 2, onde o valor de $\triangle$ é dado pela Equação 1, ambas apresentadas na seção 2.2.2. No início da execução do algoritmo, quando a variável temperatura ainda está com valores elevados, a probabilidade de troca de estados é relativamente alta, permitindo uma ampla exploração do espaço de soluções do problema.

A cada iteração do algoritmo a variável temperatura é reduzida (processo de resfriamento). Isso faz com que o Flexmap-TS evite escolher as coberturas inferiores com tanta frequência. Ao decorrer da execução, a temperatura vai diminuindo e o Têmpera Simulada começa a convergir apenas para soluções superiores, em busca da solução ótima globalmente. O algoritmo termina quando a temperatura chega a zero ou a um valor muito próximo de zero.

4.2 Desenvolvimento da Abordagem

A proposta inicial deste trabalho foi o desenvolvimento de uma abordagem para o Mapeamento Tecnológico que utilizasse o algoritmo de Têmpera Simulada. Uma versão preliminar foi implementada com base no algoritmo clássico de TS, para avaliar se a abordagem de IA era aplicável no processo de Mapeamento Tecnológico. Essa versão preliminar do Flexmap-TS será referenciada como *Cenário 1*.

Após implementado o Cenário 1, foi realizado uma bateria de testes com o objetivo de testar sua eficiência. A partir da avaliação dos testes e da análise de comportamento do Cenário 1, foram efetuadas alterações no Flexmap-TS para melhor adaptar a implementação para o problema do Mapeamento Tecnológico. A partir dessas alterações foi desenvolvido a versão final da técnica, apresentada neste capítulo

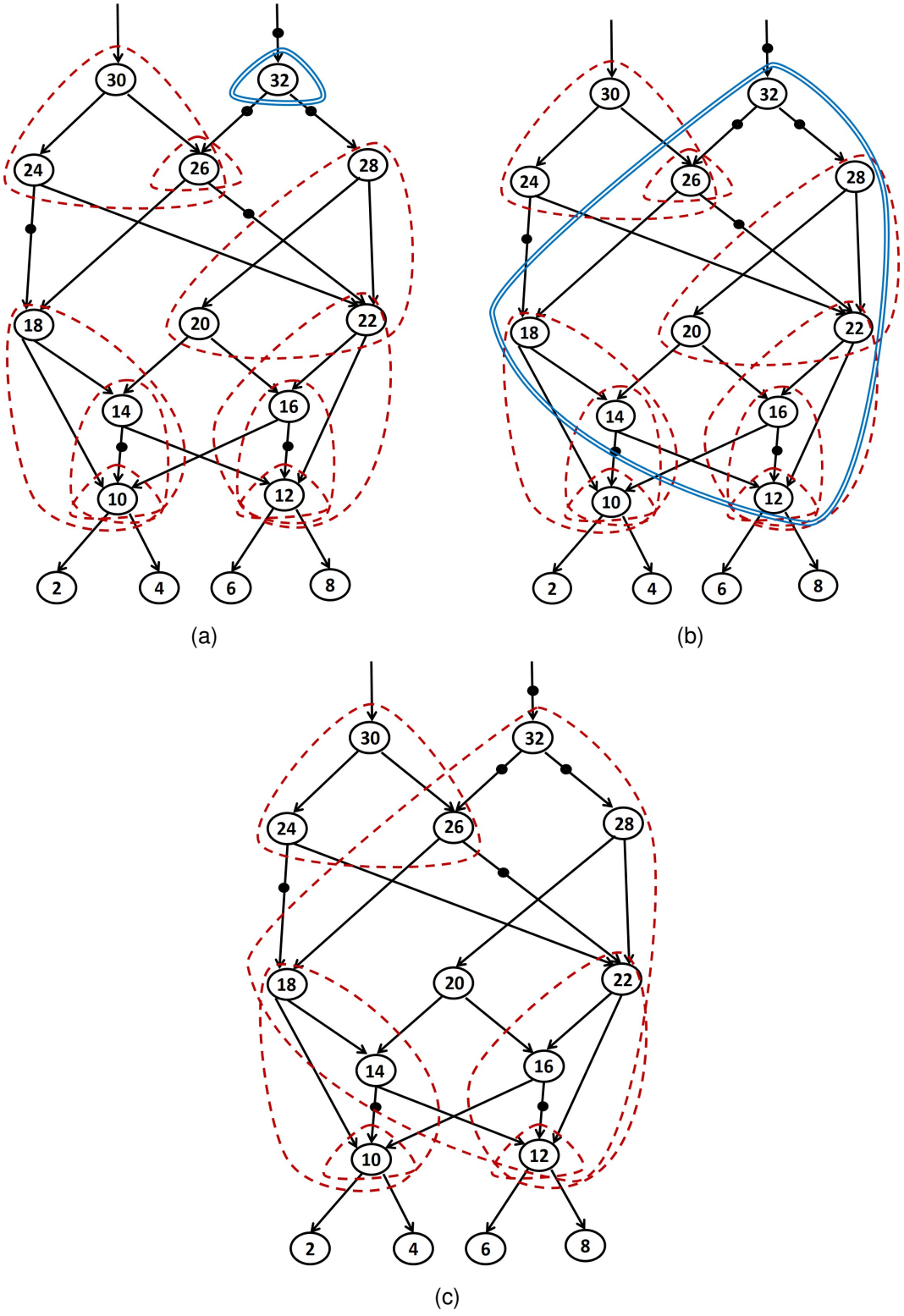


Figura 13: Execução de uma iteração da técnica de Têmpera Simulada

como *Cenário 2*. Os cenários e as alterações implementadas serão apresentados neste ca?ítulo.

4.2.1 Cenário 1

O Cenário 1 foi desenvolvido com base na técnica convencional de Têmpera Simulada. Uma série de experimentos permitiram a análise exaustiva do comportamento do algoritmo com diferentes valores em seus parâmetros, levando a escolha empírica daqueles que obtiveram os melhores resultados. Assim, a velocidade de decaimento da variável temperatura foi ajustada para possuir um decaimento de 0.1% de temperatura por iteração do Têmpera Simulada, permitindo que o Flexmap-TS executasse uma quantidade significante de iterações antes de começar a convergir para uma solução final. O Flexmap-TS é executado até que a temperatura se reduza a um valor muito próximo de zero, finalizando a execução. O Algoritmo 3 mostra um pseudo-algoritmo da execução do Flexmap-TS no Cenário 1, enquanto a Figura 14 apresenta o fluxograma do pseudo-algoritmo em questão.

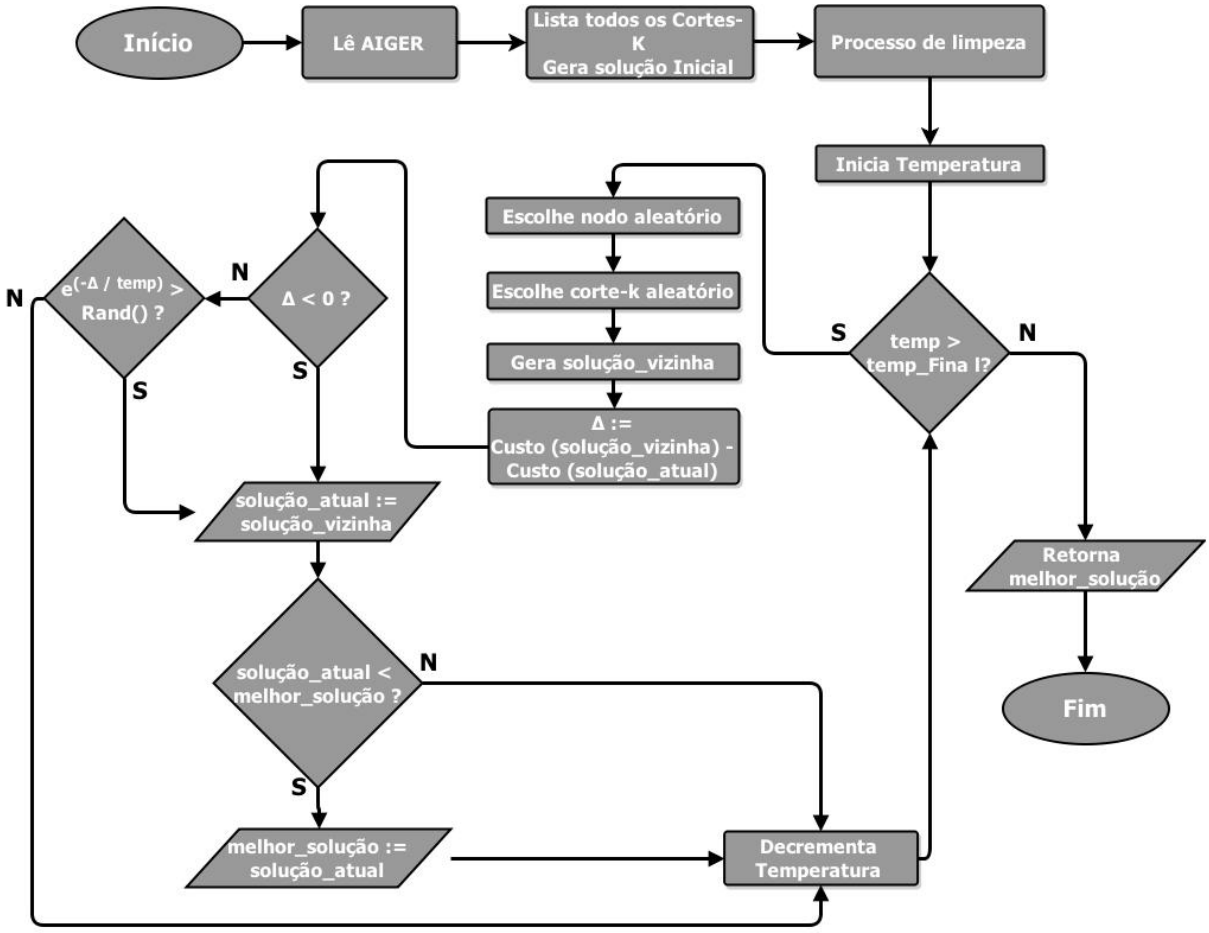


Figura 14: Fluxograma do Flexmap-TS no Cenário 1

Com a execução de várias simulações no Cenário 1, foi possível realizar uma análise profunda do comportamento do Flexmap-TS. O primeiro fator percebido foi que a utilização de um estado inicial aleatório para a inicialização do FlexMap-TS gera uma cobertura consideravelmente inferior a qualquer abordagem de mapeamento

Algoritmo 3: Pseudo-Algoritmo do Flexmap-TS no Cenário 1

```
1  ig ← le_AIGER();
2  cortesK ← encontra_todos_cortesK(aig);
3  coberturaAtual ← solucao_Inicial_Aleatoria(aig,cortesK);
4  proc_Limpeza(coberturaAtual);
5  melhorCobertura ← coberturaAtual;
6  while ( temperatura > valorMinimo ) do
7      nodo ← nodo_Aleatorio(aig);
8      novoCorteK ← cortek_Aleatorio(nodo,cortesK);
9      coberturaVizinha ← gera_Vizinho(coberturaAtual,nodo,novoCorteK);
10     Δ ← custo(coberturaVizinha) - custo(coberturaAtual);
11     if ( Δ < 0 ) then
12         coberturaAtual ← coberturaVizinha;
13         if ( coberturaAtual < melhorCobertura ) then
14             melhorCobertura ← coberturaAtual;
15         end
16     else
17         if ( e^(−Δ/temperatura) < Rand() ) then
18             coberturaAtual ← coberturaVizinha;
19         end
20     end
21     temperatura ← temperatura - 0.1%;
22 end
```

existente, incluindo técnicas simples e de rápida solução. Essa característica, apesar de ser padrão nos problemas de Têmpera Simulada, atrasa significantemente o algoritmo até que o mesmo consiga convergir para uma cobertura que possa ser considerada aplicável no circuito. O maior problema encontrado entretanto, foi durante a execução das iterações do FlexMap-TS.

Antes da execução do laço principal do Têmpera Simulada, é necessário o ajuste da variável temperatura, que além de controlar a probabilidade de troca de estados, é responsável também pelo número de iterações que o TS vai executar antes de retornar a solução final. Assim, quanto maior for o valor da temperatura fixado, maior será o número de iterações executadas pelo TS. O problema dessa abordagem é a falta de conhecimento da grandeza do circuito de entrada. A utilização de um valor limitado de temperatura em um circuito consideravelmente grande acarretará no término precoce do algoritmo, retornando uma cobertura muito inferior a considerada ótima. Em contrapartida, um valor de temperatura muito elevado aplicado em um circuito pequeno implicará em um custo computacional elevado e desnecessário, tornando o processo longo mesmo para circuitos simples.

Além do problema do ajuste do valor da variável temperatura, outro ponto desfavorável foi encontrado ao analisar essa variável. Foi notado que estados vizinhos ao

atual no MT não possuem diferenças que possam ser consideradas fortemente significativas. Assim, o valor retornado por Δ normalmente é um valor pequeno. Como mostrado anteriormente, quando o TS realiza o cálculo do Δ e encontra um valor maior que zero, o algoritmo executa um cálculo probabilístico para decidir se a solução vizinha deve ou não ser aceita. No momento da avaliação de probabilidade, se a variável temperatura possuir um valor elevado e o valor calculado de Δ for muito baixo, a probabilidade de que a troca ocorra será incrivelmente alta. Sendo assim o FlexMap-TS perde muito tempo avançando para situações ligeiramente inferiores até que o mesmo consiga limitar a probabilidade de troca e volte a encontrar soluções melhores.

Para melhor explicar o problema relatado acima, propõe-se a análise da Figura 13 de maneira inversa (ou seja, da Figura 13(c) em direção à 13(a)). Supõe-se que no momento da Figura 13(c), o nodo 32 foi escolhido aleatoriamente para ter seu *corte-K* modificado e a solução vizinha gerada foi a solução da Figura 13(a). A solução atual possui 6 *cortes-K* na sua cobertura, enquanto a solução vizinha possui 10 *cortes-K*. Com essa base, $\Delta = 4$, o que faz com que o FlexMap-TS inicie a etapa de cálculo de probabilidade para troca de soluções.

Como a probabilidade da troca é calculada utilizando a Equação 1, qualquer valor de temperatura do sistema que seja maior que “40” irá retornar uma probabilidade de aceitação superior a 90%. Se na iteração seguinte a solução encontrada também tiver um Δ apenas ligeiramente maior que zero, essa solução inferior também terá uma probabilidade muito alta de ser aceita, já que o decaimento da temperatura acontece muito lentamente (0.1%). A Figura 15 mostra o comportamento da equação de probabilidade em temperaturas muito elevadas.

Esse processo de escolhas inferiores se repete até que que o FlexMap-TS alcance um ponto onde as probabilidades de troca não ocorra com tanta frequência. Entretanto, devido a grande ocorrência de trocas por estados inferiores ocorridas na etapa de temperatura alta, a abordagem precisa de um grande esforço computacional para conseguir chegar numa solução próxima da gerada inicialmente e consequentemente evoluir até uma solução aceitável para o problema.

Analisando os problemas encontrados no Cen[ario 1, uma nova implementação foi sugerida, com o objetivo de corrigir os problemas encontrados e otimizar a técnica.

4.2.2 Cenário 2

Os estados vizinhos a uma solução de cobertura do Mapeamento Tecnológico raramente possuem diferenças significativas sobre a solução atual quando se considera o número de *cortes-K* utilizados. Devido ao fato da probabilidade de aceitação do Têmpera Simulada ser normalmente dada pela Equação 1, quando se possui um valor muito baixo para Δ em iterações do têmpera simulada onde a temperatura está alta, a probabilidade da troca de soluções ocorrer é consideravelmente alta (foi analisado

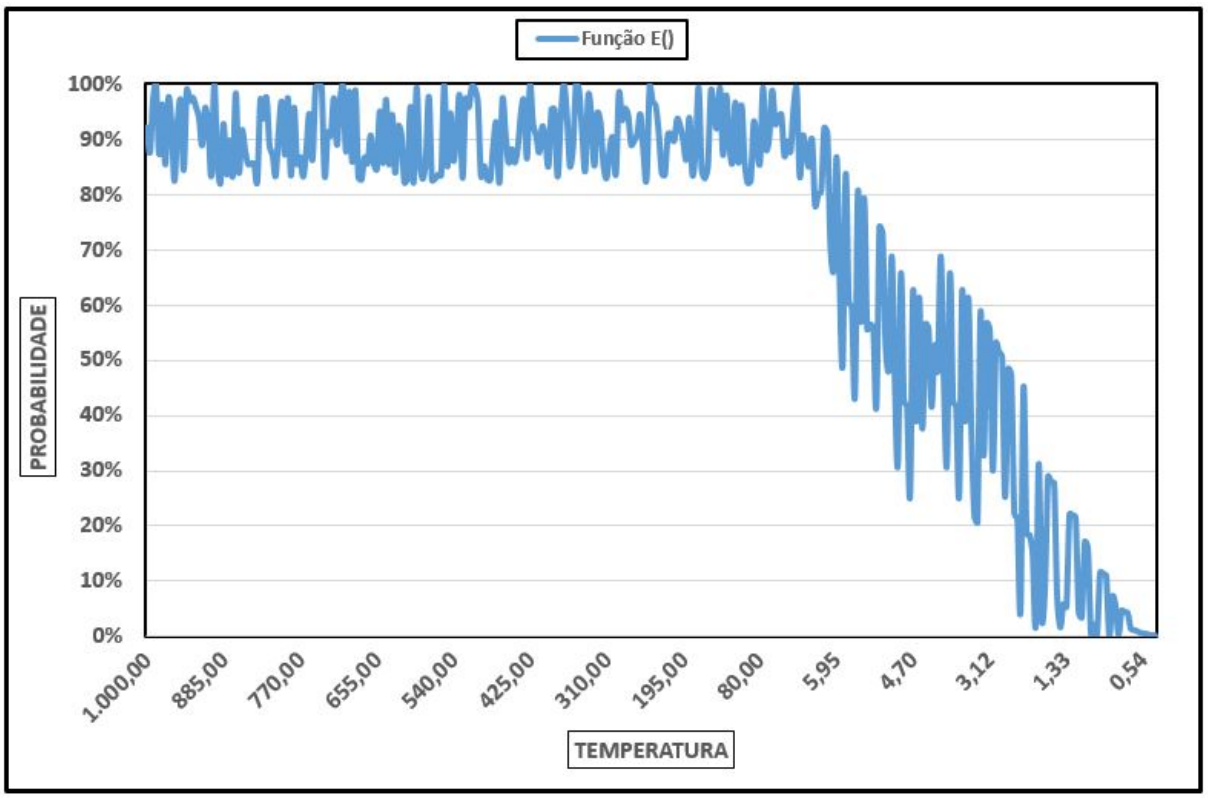


Figura 15: Comportamento da equação de probabilidade no Cenário 1

que nas situações iniciais a probabilidade de troca alternava entre 90% e 99%).

Sabendo que os valores de Δ são normalmente baixos, o Cenário 2 limitou essa probabilidade de troca, utilizando valores de temperatura inicial que gerassem probabilidades menores de troca de estado. Foi feita uma análise para a escolha de valores de temperatura inicial e foi encontrado que um valor de temperatura que gerasse probabilidades de aproximadamente 50% de troca de estado nas iterações iniciais do algoritmo retornava os resultados mais satisfatórios. A Figura 16 mostra o comportamento da equação de probabilidade com a temperatura reduzida.

A desvantagem deste Cenario foi a redução considerável do número de iterações do FlexMap-TS, já que o número de iterações é diretamente relacionado com a variável temperatura. Dessa maneira foi proposta a utilização de um Têmpera Simulada com múltiplos reinícios (HENTSCHKE, 2002). Caso a temperatura do ambiente atinja o valor mínimo antes do algoritmo conseguir otimizar o suficiente a cobertura, um novo ciclo do TS é iniciado utilizando o estado atual de cobertura como estado inicial do ciclo seguinte.

Além do Cenário2 conseguir resolver os problemas apresentados no Cenário anterior, ele também conseguiu resolver o problema mostrado na Figura 17. O gráfico apresentado mostra a execução de 4 ciclos do FlexMap-TS no Cenário 2 em um dos benchmarks utilizados, apresentando apenas o número de *cortes-K* da melhor solução obtida até a iteração corrente do algoritmo.

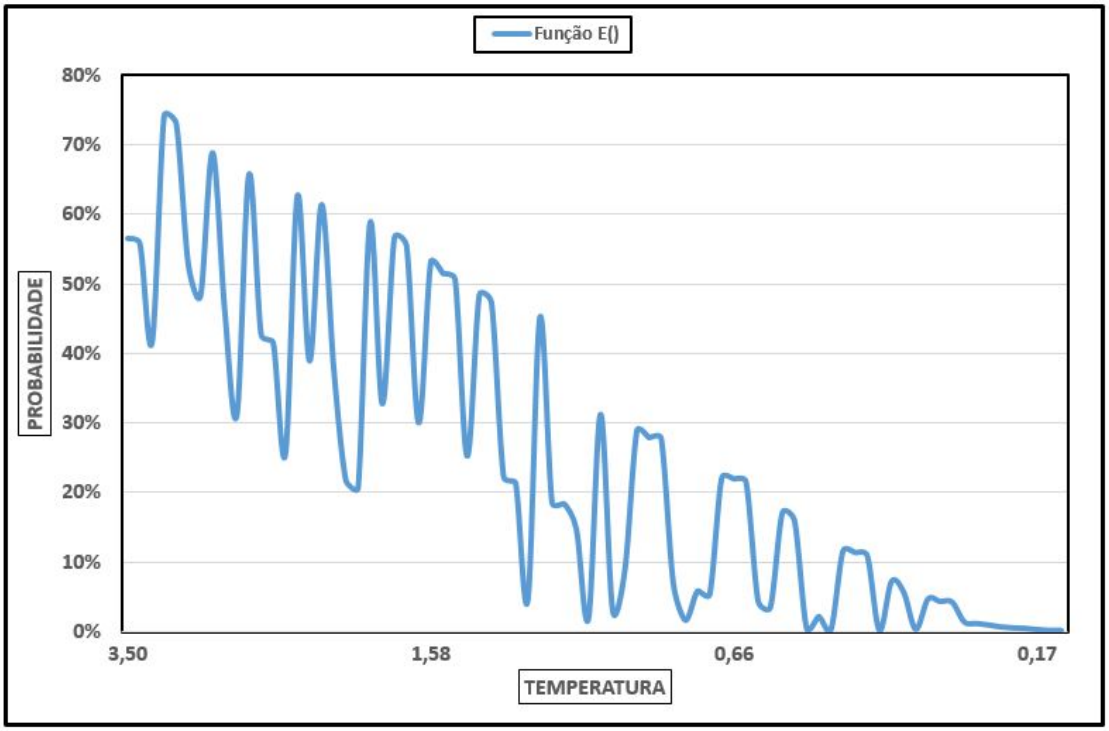


Figura 16: Comportamento da equação de probabilidade no Cenário 2

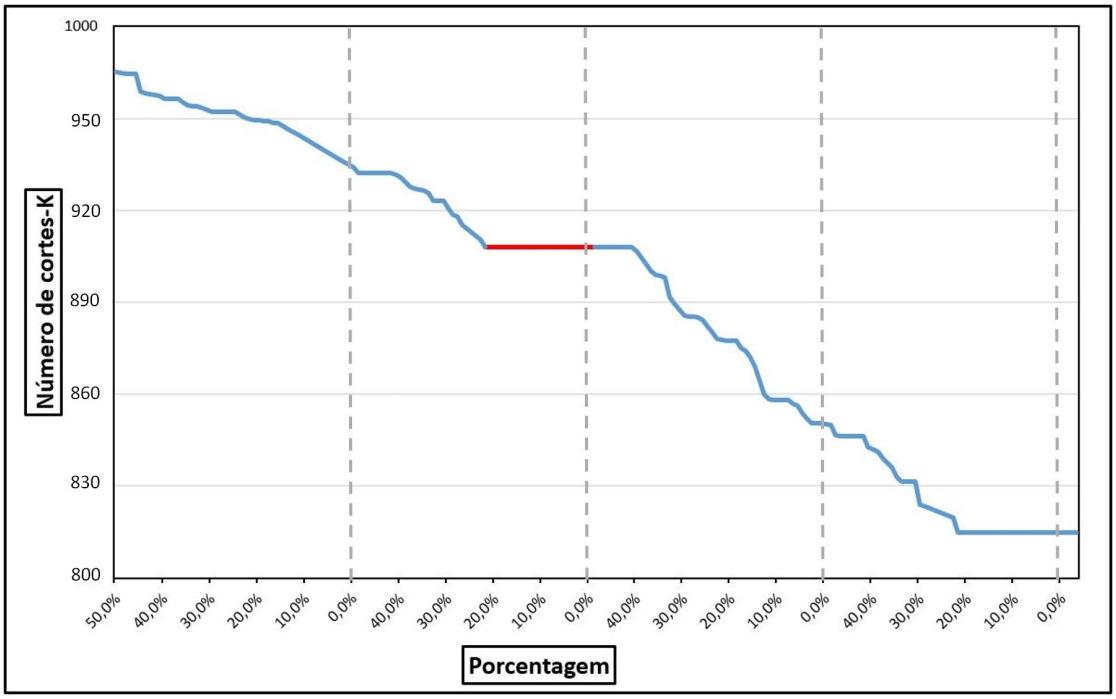


Figura 17: Gráfico de execução de um benchmark no Cenário 2

Como pode-se notar, mesmo com uma redução da variável temperatura (e consequentemente a redução do número de iterações do algoritmo), a utilização de vários ciclos de reinício da abordagem consegue otimizar o problema do Mapeamento Tecnológico. É importante notar o acontecimento realçado no segundo ciclo da figura em

questão. Quando a temperatura do sistema chega a um valor muito baixo, a probabilidade de troca para estados inferiores diminui e o algoritmo adota um comportamento guloso. Neste instante pode-se notar que o algoritmo não consegue encontrar soluções melhores, possivelmente tendo encontrado um *mínimo local*.

Ao iniciar um novo ciclo de iterações, onde a temperatura é reajustada para aumentar a probabilidade, a técnica volta a encontrar soluções melhores, tendo assim escapado do mínimo local e continuado a otimização.

No novo Cenário a condição de parada do algoritmo também foi alterada. Dado que o algoritmo agora torna-se dependente do número de ciclos de TS para o fim de sua execução, um valor fixo de ciclos iria gerar os mesmos problemas encontrados na versão preliminar. Assim foi determinado que o algoritmo só deve parar sua execução quando fosse executado um número máximo de ciclos sem encontrar uma solução superior à melhor obtida até o momento. Quando o FlexMap-TS encontra uma solução superior à melhor solução global obtida, o contador de ciclos é resetado e o algoritmo continua sua execução. O Algoritmo 4 demonstra o pseudo-algoritmo do Cenário 2, enquanto a Figura 18 representa o fluxograma desse mesmo algoritmo. As áreas com a coloração mais escura mostram as principais diferenças desta abordagem para implementação do Cenário 1.

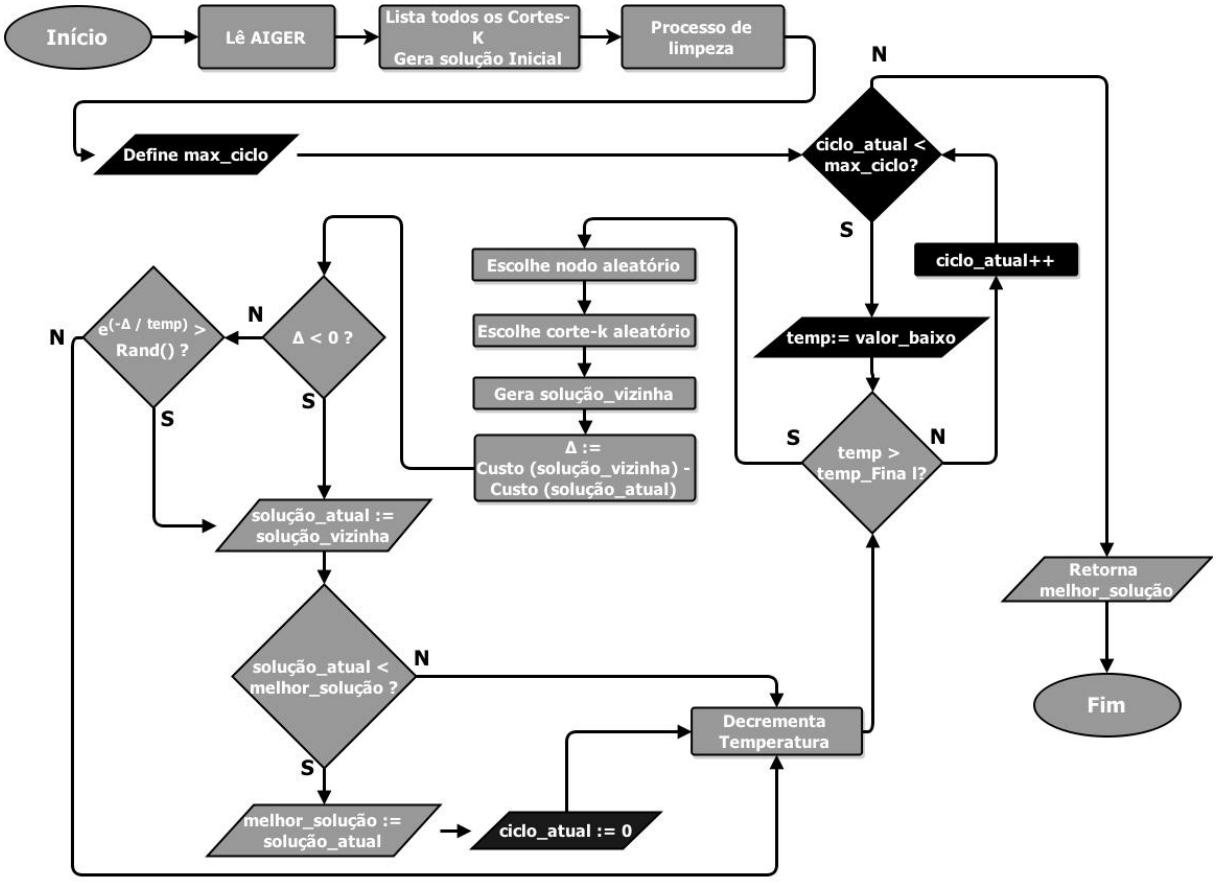


Figura 18: Fluxograma do Cenário 2

Algoritmo 4: Pseudo-Algoritmo do Cenário 2

```
1  ig ← le_AIGER();
2  cortesK ← encontra_todos_cortesK(aig);
3  coberturaAtual ← solucao_Inicial_Aleatoria(aig,cortesK);
4  proc_Limpeza(coberturaAtual);
5  melhorCobertura ← coberturaAtual;
6  maxCiclo ← valor;
7  cicloAtual ← 0 ;
8  while ( ciclo < maxCiclo ) do
9      temp ← valorBaixo;
10     while ( temp > valorMinimo ) do
11         nodo ← nodo_Aleatorio(aig);
12         novoCorteK ← cortek_Aleatorio(nodo,cortesK);
13         coberturaVizinha ← gera_Vizinho(coberturaAtual,nodo,novoCorteK);
14         Δ ← custo(coberturaVizinha) - custo(coberturaAtual);
15         if ( Δ < 0 ) then
16             coberturaAtual ← coberturaVizinha;
17             if ( coberturaAtual < melhorCobertura ) then
18                 melhorCobertura ← coberturaAtual;
19                 cicloAtual ← 0 ;
20             end
21         else
22             if ( e^(−Δ/temperatura) < Rand() ) then
23                 coberturaAtual ← coberturaVizinha;
24             end
25         end
26         temp ← temp - 0.1%;
27     end
28     cicloAtual++;
29 end
```

A utilização de ciclos de decaimento de temperatura possibilitou que o FlexMap-TS conseguisse escapar de mínimos locais e prosseguir com a otimização do circuito. A Figura 19 mostra um subconjunto de pontos dos últimos 30 ciclos de um *benchmark* testado no Cenário 2. Pode-se notar que mesmo com valores de temperatura reduzidos, a equação de probabilidade permite com que ocorram trocas suficientes de estado para que o algoritmo consiga chegar em um estado otimizado.

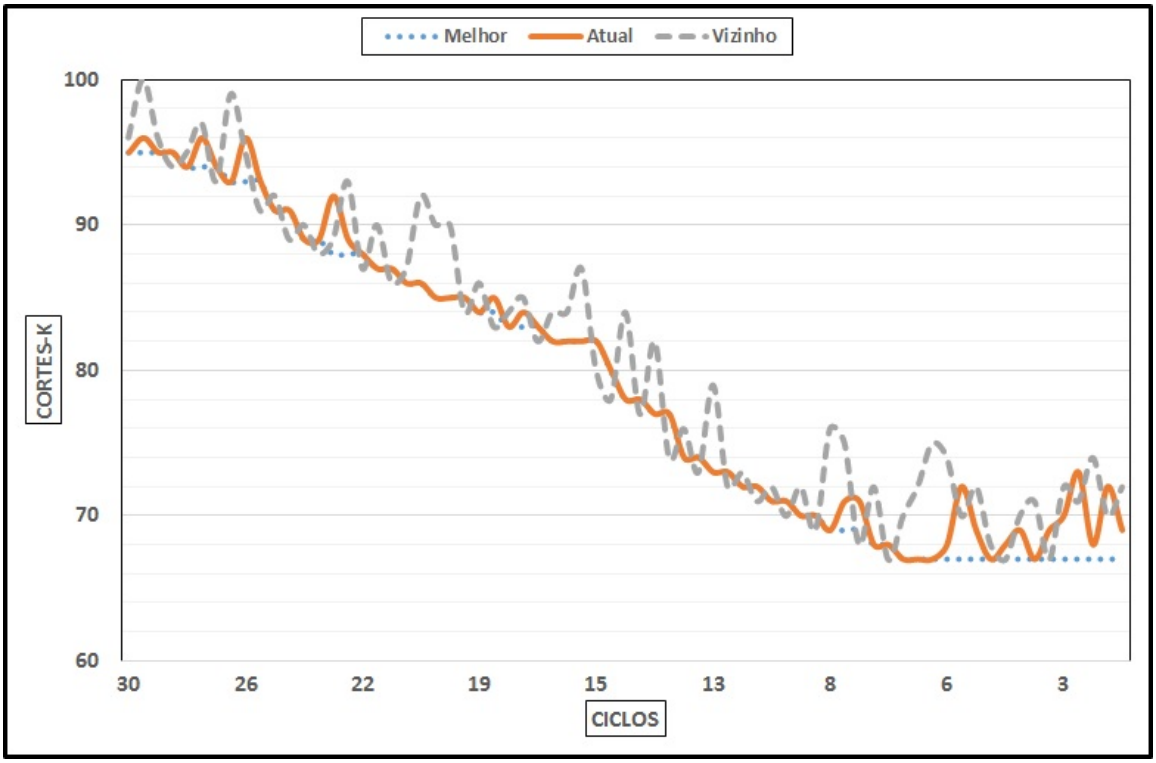


Figura 19: Subconjunto de pontos da execução do FlexMap-TS

4.3 Considerações Finais

Este capítulo apresentou uma abordagem alternativa para o problema do Mapeamento Tecnológico. Esta abordagem foi adaptada do algoritmo de Têmpera Simulada e tem como objetivo evitar mínimos locais no espaço de busca de soluções do problema em questão. O algoritmo desenvolvido foi inserido na ferramenta FlexMap para servir como alternativa às abordagens já existentes. No capítulo seguinte será detalhado os experimentos realizados para a validação do FlexMap-TS, juntamente com os resultados e comparações com a ferramenta ABC (ABC, 2014), considerada estado-da-arte no processo de Mapeamento Tecnológico.

5 EXPERIMENTOS E RESULTADOS

Para que se possa avaliar a qualidade do FlexMap-TS, foram realizados diversos experimentos. Os resultados que serão mostrados nesse capítulo foram obtidos através da execução de experimentos na ferramenta FlexMap, utilizando tanto o Cenário 1, baseado no algoritmo padrão do TS como o Cenário 2, que foi otimizado após análise de comportamento do Cenário 1. Assim, foi possível observar que a utilização de Têmpera Simulada no Mapeamento Tecnológico consegue encontrar bons resultados.

As descrições de entrada são fornecidas utilizando benchmarks dos pacotes ISCAS85 (BRYAN, 1985) e MCNC91 (YANG, 1991), considerados padrão para comparações na área de síntese lógica. Devido a problemas de memória e de simulações que não conseguiram terminar em tempo mensurável para essa dissertação devido ao tamanho do benchmark de entrada, não foi possível realizar o experimento com todos benchmarks dos pacotes. Foram utilizados no total 85 benchmarks diferentes. Para verificar a eficiência da abordagem implementada, 30 simulações foram executadas com cada benchmark do subconjunto para cálculos de média, totalizando-se aproximadamente 2500 simulações com cada abordagem.

Foi determinado que os experimentos teriam uma cobertura inicial aleatória, mesmo que esta abordagem tivesse um custo computacional significativamente maior que uma abordagem utilizando uma cobertura pré-otimizada. Esta escolha foi dada com o objetivo de ampliar o espaço de busca e tentar evitar que a solução convergisse para mínimos locais. Como o foco do trabalho foi o mapeamento para FPGAs, a métrica adotada para a comparação dos resultados foi o número de *cortes-K* necessários para a cobertura total do circuito. Em todas as simulações executadas, foram adotados valores fixos para *"K"*, sendo ***K=4*** e ***K=5***, visto que comumente são encontrados FPGAs com LUTs com 4 e 5 entradas.

O processo de validação foi aplicado através da ferramenta ABC (ABC, 2014). A partir da descrição inicial do circuito foi aplicado o mapeamento com a abordagem proposta utilizando a ferramenta FlexMap. Após finalizado o mapeamento, é possível verificar se o processo de mapeamento não alterou a lógica que o circuito implementa,

utilizando a descrição inicial e a descrição resultante. Este processo é feito com o comando *checking* do ABC, cuja finalidade é testar se a lógica de duas descrições são equivalentes. Todos os circuitos apresentados nas próximas seções foram devidamente validados, garantindo que as abordagens desenvolvidas não alteraram a funcionalidade do circuito em questão.

5.1 Cenário 1

Apesar de apresentar resultados favoráveis quando comparados ao estado inicial aleatório, a primeira implementação não apresenta resultados comparáveis ao ABC na grande maioria dos testes. A Figura 20 demonstra a progressão de um pequeno subconjunto dos experimentos executados no Cenário 1 para que seja possível visualizar a otimização do FlexMap-TS em relação ao estado inicial do algoritmo. Juntamente no gráfico também são apresentados os valores conseguidos utilizando a ferramenta ABC, para que possa ser feita uma comparação do Cenário 1 com a ferramenta estado-da-arte.

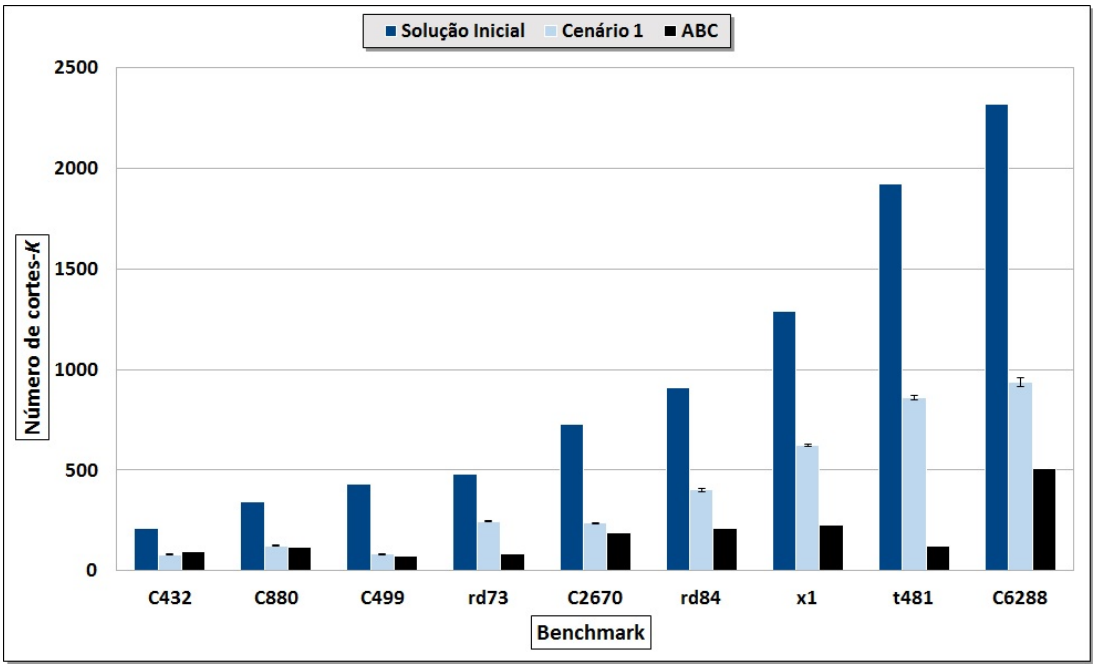


Figura 20: Alguns resultados do Cenário 1

Como se pode observar na Figura 20, à medida que o circuito aumenta de complexidade e tamanho, sua quantidade inicial de *cortes-K* aumenta consideravelmente. Ao analisar os resultados obtidos a partir das simulações foi avaliado que, dado um estado inicial aleatório para o circuito, o Cenário 1 obteve otimizações de pelo menos 55% sobre os valores iniciais em todos os benchmarks avaliados, apresentando melhoras de até 80% em alguns dos casos.

Apesar dessa redução significativa sobre o valor inicial, este Cenário não conse-

gue se comparar com os resultados da ferramenta ABC quando considera-se circuitos complexos e de grande porte. Para uma melhor comparação dos resultados, foram executados 30 testes com cada benchmark e calculados sua média e desvio padrão. Os resultados obtidos foram divididos em duas tabelas e estão apresentados nas Tabelas 4 e 5, localizadas no Anexo A. Nas tabelas é possível ver o valor do benchmark quando executado pela ferramenta ABC, a média dos resultados obtidos, a diferença entre as duas ferramentas (tanto numérica quanto percentual) e o desvio padrão do TS. Ambas as tabelas estão ordenadas com base no valor de Porcentagem da coluna Diferença, em ordem crescente. Para melhor visualização dos dados, a Figura 21 apresenta uma normalização dos dados utilizando os valores do ABC como referência. Os valores completos da normalização também encontram-se nas Tabelas 4 e 5. Os valores em verde representam os experimentos onde o FlexMap-TS superou o ABC, enquanto os valores em vermelho representam os casos onde a ferramenta estado-da-arte foi superior. Situações que nao apresentam nenhuma barra representam os casos onde as duas abordagens obtiveram resultados equivalentes.

Como se pode observar na Figura 21, o FlexMap-TS não obteve bons resultados quando comparado ao ABC. Além disso, percebe-se que existem diversos *benchmarks* onde o ABC conseguiu otimizações muito superiores ao FlexMap-TS. Como explicado anteriormente, os principais fatores que desfavorecem o Cenário 1 são a utilização de um estado inicial sem nenhuma pré-otimização e o comportamento da variável temperatura.

A cobertura inicial aleatória é consideravelmente inferior à qualquer abordagem existente e faz com que o FlexMap-TS gaste muito tempo e custo computacional para convergir para soluções minimamente satisfatórias. Além disso, quando a temperatura do ambiente é alta, a probabilidade de troca para soluções inferiores é grande e o FlexMap-TS tende a aceitar soluções inferiores à atual com muita frequência. Assim, mesmo tendo soluções iniciais péssimas, a abordagem tende a continuar buscando soluções inferiores até o momento em que a temperatura do ambiente diminua e a troca de estados ocorra com menos frequência.

Outro fator importante o qual dependente da variável temperatura é o número de iterações executados antes que o FlexMap-TS pare. A utilização de uma temperatura fixa, como foi aplicada no Cenário 1, restringe consideravelmente a abordagem. Isso se dá ao fato de que, mesmo utilizando temperaturas altas, não é possível avaliar a ordem de grandeza do *benchmark* que está sendo testado. Assim, mesmo que o FlexMap-TS encontre um espaço de soluções favorável e com a possibilidade de otimização para o circuito que está sendo testado, a execução pode parar no meio da otimização caso a temperatura se reduza a zero.

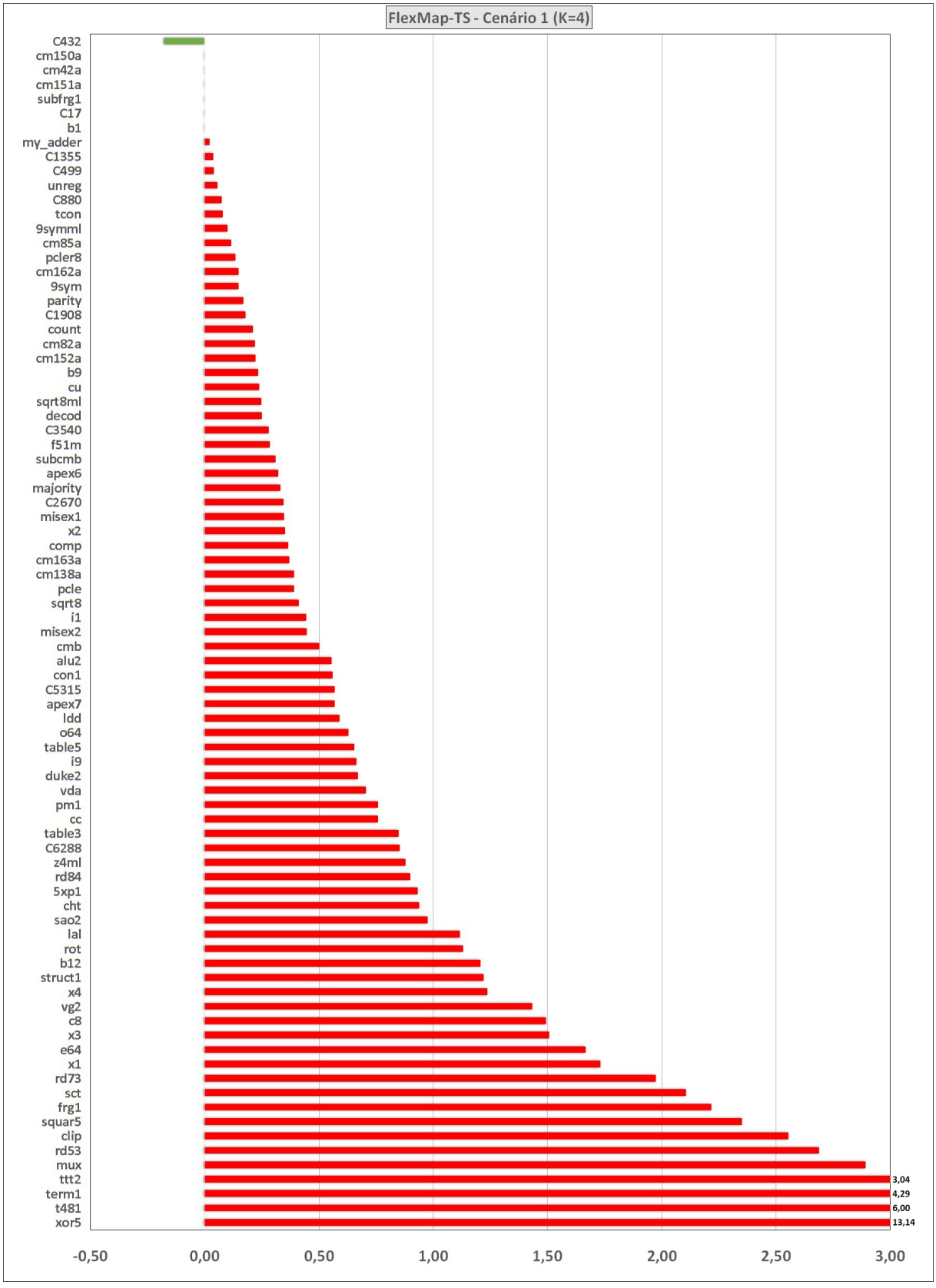


Figura 21: Cenário 1 **(K=4)**

5.2 Cenário 2

Após a análise dos testes realizados no Cenário 1, foram propostas alterações para a criação de uma nova abordagem. Essa abordagem modificou a porcentagem de aceitação de soluções inferiores para reduzir a mudança excessiva desnecessária quando o Têmpera Simulada encontra-se com temperaturas muito elevadas. Além desta mudança foram feitas alterações na condição de parada do Têmpera Simulada, interrompendo a execução com base em um número máximo de ciclos de decaimento de temperatura sem encontrar uma solução superior à melhor obtida para o circuito. Essas modificações foram incorporadas no FlexMap-TS, gerando o Cenário 2.

Para reduzir o valor da probabilidade de aceitação de escolhas inferiores foi necessário a utilização de um valor baixo para a variável temperatura. Após testes iniciais, foi encontrado que valores de temperatura entre “3” e “5” retornam probabilidades de aproximadamente até 50% para o problema em questão. Isso se deve ao fato de que as soluções vizinhas à solução atual não geram tanta perturbação no sistema, fazendo com que o cálculo do $\triangle$ retorne valores baixos. Assim, a variável temperatura foi ajustada para receber um valor *float* aleatório no intervalo [3,5].

Mesmo com os valores de temperatura baixos, o FlexMap-TS ainda executa um número considerável de iterações por ciclo. Dado que a taxa de decaimento da temperatura foi ajustada para 0,1% por iteração, um ciclo qualquer terá entre “8000” e “8500” iterações antes do final da sua execução. Finalmente, foi necessário o ajuste do número máximo de ciclos sem otimização da melhor solução encontrada até o momento. Após algumas análises realizadas em benchmarks dos pacotes, foi determinado que o FlexMap-TS termina sua execução depois de 10 ciclos sem nenhuma otimização, permitindo uma ampla busca no espaço de soluções antes da finalização do algoritmo.

Essas mudanças melhoraram significativamente a abordagem de cobertura desenvolvida, gerando resultados não só 60% a 85% superiores aos estados aleatórios iniciais, mas também comparáveis aos resultados obtidos pela ferramenta ABC. Entretanto, apesar da melhora, o Cenário 2 não conseguiu superar a ferramenta estado-da-arte em todos os casos testados, obtendo alguns resultados fortemente inferiores ao ABC.

Similarmente aos dados obtidos no Cenário 1, os valores dos experimentos realizados no Cenário 2 encontram-se nas Tabelas 6 e 7, no Anexo A. Estas tabelas apresentam os experimentos com ***k=4*** e estão ordenadas pelo valor de Porcentagem da coluna Diferença em ordem crescente. A Figura 22 apresenta os valores normalizados deste cenário.

O primeiro ponto a ser observado na Figura é que mesmo o FlexMap-TS não conseguindo encontrar soluções superiores ao ABC em todos os casos, essa abordagem

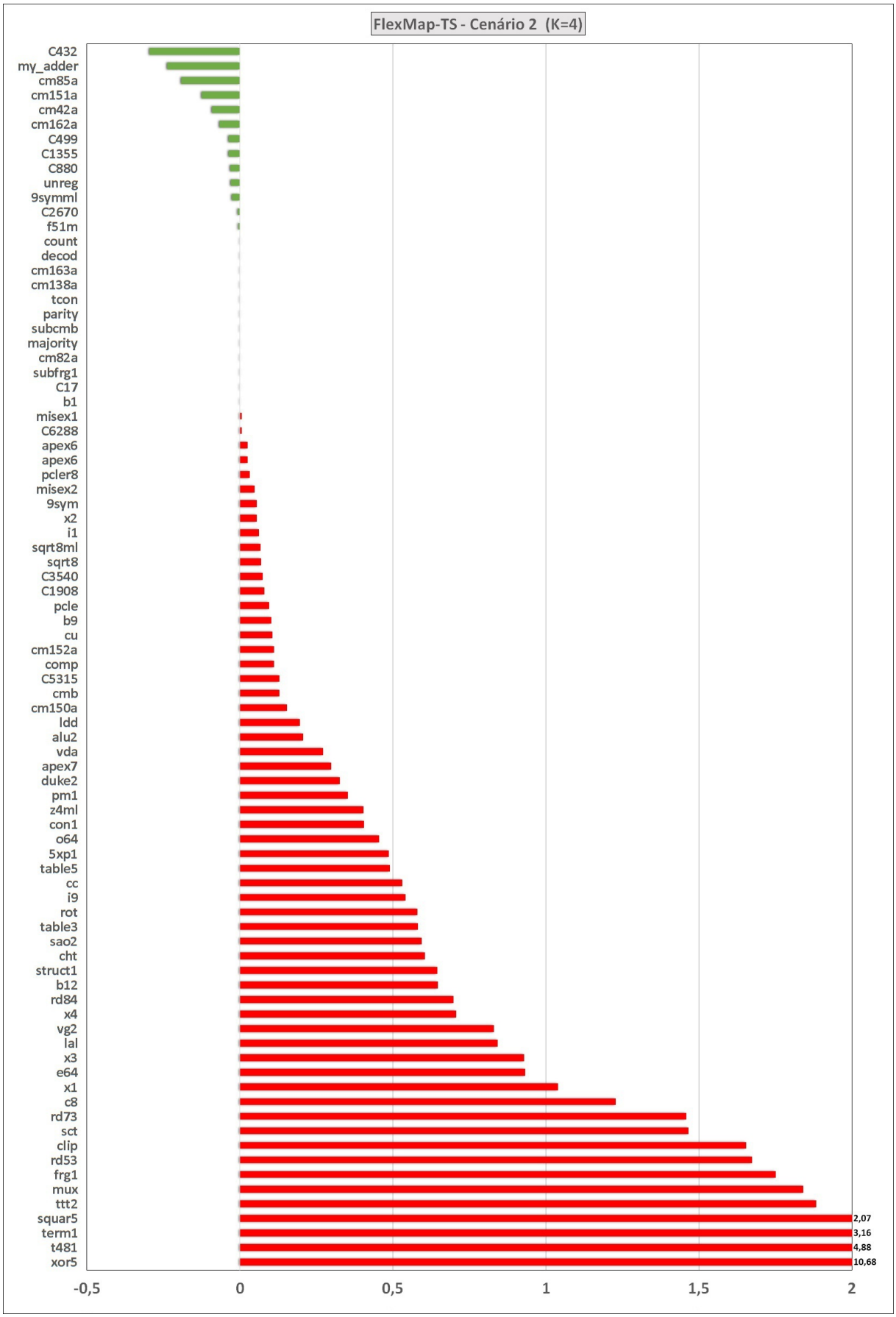


Figura 22: Cenário 2 **(K=4)**

se mostra consideravelmente superior ao Cenário 1, apresentando muito mais casos onde o FlexMap-TS consegue superar, se equiparar ou chegar a resultados muito próximos ao ABC. Assim pode-se concluir que o Cenário 2 possui potencial de encontrar soluções melhores que a ferramenta estado-da-arte.

Como se pode ver na Figura 22, apesar da técnica de Têmpera Simulada aplicada no Mapeamento Tecnológico apresentar resultados extremamente favoráveis quando comparados ao seu estágio inicial, as mesmas ainda não apresentam uma solução geral comparável à ferramenta estado-da-arte ABC em todos os benchmarks testados. Mesmo após as modificações no Cenário 1, o ABC apresenta vários casos onde o circuito foi consideravelmente superior ao FlexMap-TS.

Um ponto extremamente importante a ressaltar é o fato de que a ferramenta ABC realiza uma série de balanceamentos e associações lógicas para reduzir a função booleana. Muitas dessas associações ainda não estão implementadas na ferramenta FlexMap até o presente momento, causando uma diferença significante nos valores obtidos. Os benchmarks com maiores porcentagens de diferença foram avaliados para uma análise desses balanceamentos.

Entre todos os experimentos avaliados, o benchmark chamado de “xor5” foi o que retornou as maiores porcentagens de diferença (92,92% no Cenário 1 e 91,44% no Cenário 2). Esse circuito possui 59 vértices em seu formato AIG, ou seja, a cobertura inicial da abordagem implementada irá gerar uma cobertura com 59 *cortes-k* antes da etapa de limpeza. Após a execução do FlexMap-TS, foi constatado que o algoritmo desenvolvido conseguiu reduzir o número de *cortes-k* para uma média de “28,27” no Cenário 1 e “23,36” no Cenário 2. Esses valores no entanto encontram-se muito distantes do valor “2” encontrado pela ferramenta estado-da-arte.

Utilizando a funcionalidade *“write_dot”* do ABC, é possível obter uma representação visual do circuito final criado. Observando a Figura 24 é possível observar que houve rebalanceamentos e associações lógicas que minimizaram consideravelmente o circuito. Acredita-se que esses balanceamentos também são efetuados em alguns dos benchmarks estudados, o que acarretou a diferença significativa entre o FlexMap-TS e o ABC.

Além dos experimentos utilizando *cortes-K* com um valor de *K=4*, foram realizados também experimentos com *K=5*, ampliando o espaço de busca de soluções do problema nos mesmos benchmarks testados anteriormente. A Figura 23 os valores normalizados dos experimentos realizados com esta configuração.

Um fator interessante a se observar é o benchmark “xor5”. Na configuração anterior do FlexMap-TS, este benchmark tinha apresentado a maior porcentagem de diferença entre a abordagem desenvolvida e a ferramenta estado-da-arte. Na configuração de “K” com 5 entradas no entanto, o TS conseguiu encontrar o valor ótimo para o problema, encontrando a solução que utiliza apenas 1 *corte-K*. Isso mostra

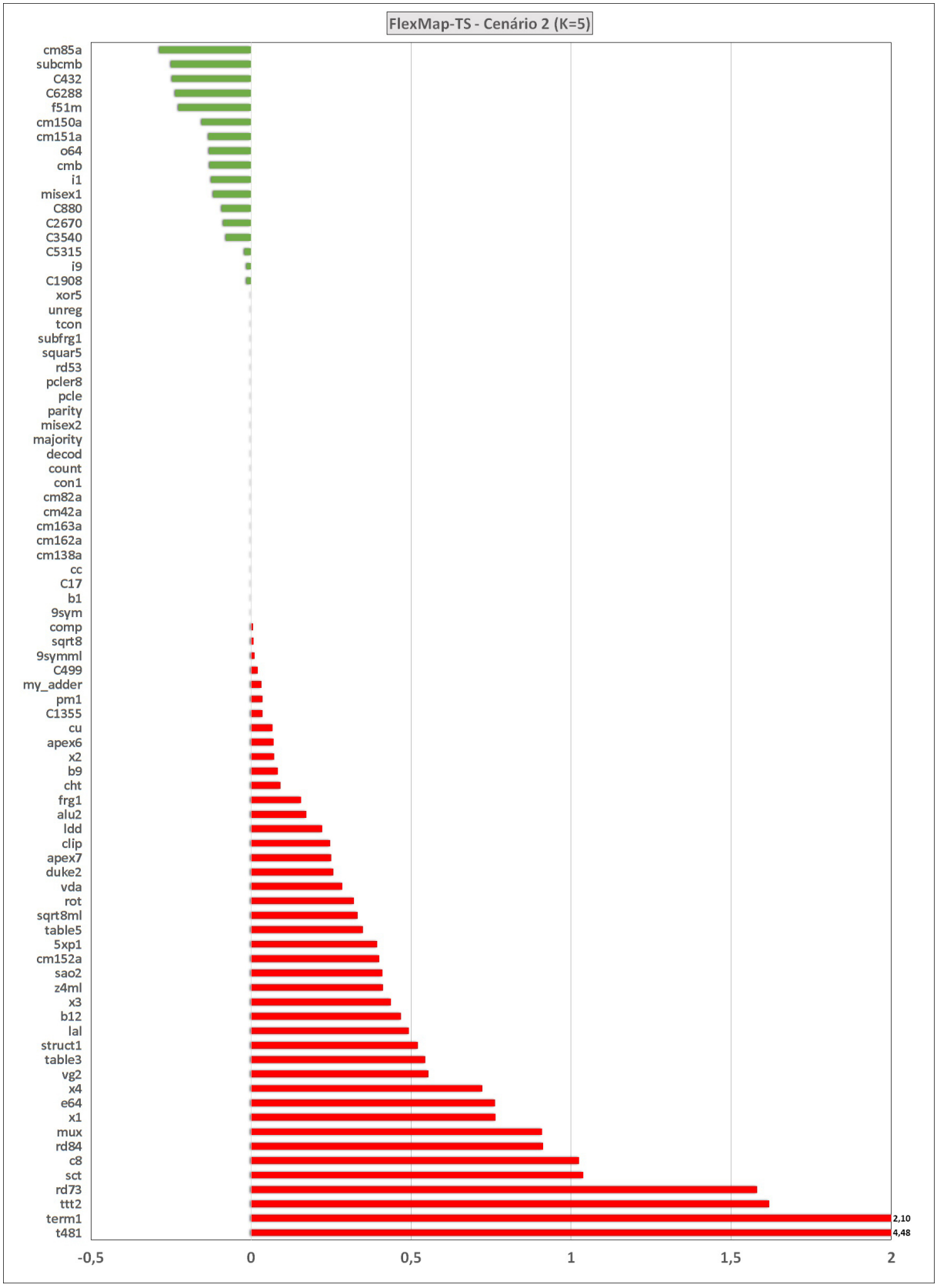


Figura 23: Cenário 2 **(K=5)**

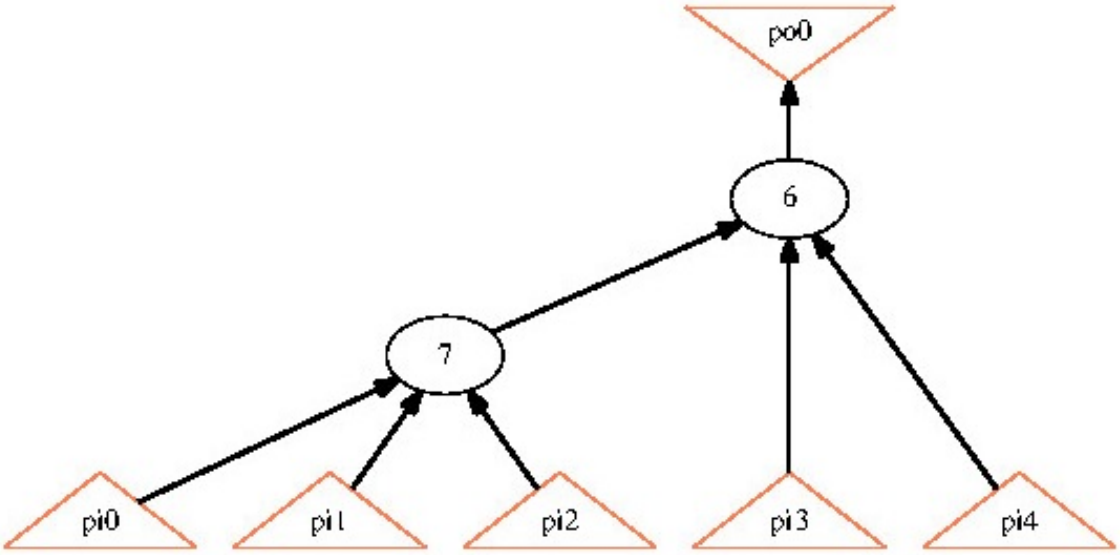


Figura 24: Representação do benchmark *xor5*

como configuração dos parâmetros do problema influi fortemente na solução encontrada pelo FlexMap-TS. Os valores dos experimentos com *K=5* podem ser observados nas Tabelas 8 e 9, encontradas no Anexo A.

5.3 Considerações Finais

Após a execução dos experimentos apresentados neste trabalho, foi possível analisar a eficiência da abordagem de Mapeamento Tecnológico implementada utilizando Têmpera Simulada. Apesar de não conseguir superar o ABC de modo geral, a abordagem proposta no Cenário 2 conseguiu encontrar algumas soluções comparáveis e até mesmo superiores que a ferramenta ABC em vários benchmarks testados, encontrando soluções que melhoram em até 42% em alguns benchmarks específicos.

Para uma análise estatística precisa dos experimentos realizados neste trabalho, foi realizado um teste-*t* não pareado em cada um dos 85 *benchmarks* utilizados nas simulações, tanto com *K=4* como *K=5* e comparando as 30 execuções da do FlexMap-TS com o resultado do ABC.

Apesar do ABC se mostrar superior em grande parte dos casos, foram encontrados experimentos onde o FlexMap-TS conseguiu superar o ABC, com uma diferença estatisticamente significante. A Tabela 3 mostra os resultados desta análise. A Tabela mostra qual técnica se mostrou superior em determinado número de *benchmarks*, enquanto a célula “Equivalente” representa o número de casos onde não existe diferença estatística entre o ABC e a abordagem implementada.

Como se pode observar no Cenário 2, o FlexMap-TS conseguiu encontrar resultados superiores em aproximadamente 19% dos casos com *K=4* e 26% dos casos com

K=5. Adicionando os experimentos onde a abordagem se equivaleu ao ABC, estes valores sobem para aproximadamente 36% e 53%, respectivamente.

Tabela 3: Tabela de análise estatística

Cenário 1 *(K=4)*		
Abordagem	**Quantidade**	**Porcentagem**
ABC	73	85,88%
FlexMap-TS	5	5,88%
Equivalente	7	8,24%

Cenário 2 *(K=4)*		
Abordagem	**Quantidade**	**Porcentagem**
ABC	54	63,53%
FlexMap-TS	16	18,82%
Equivalente	15	17,65%

Cenário 1 *(K=5)*		
Abordagem	**Quantidade**	**Porcentagem**
ABC	40	47,06%
FlexMap-TS	22	25,88%
Equivalente	23	27,06%

6 CONCLUSÕES

Este trabalho apresentou uma nova abordagem para o problema de cobertura do MT. A abordagem proposta é baseada no algoritmo de Têmpera Simulada e foi incorporada à ferramenta de Mapeamento Tecnológico FlexMap.

Para a adaptação do TS para o Mapeamento Tecnológico foram desenvolvidas várias sub-rotinas, como a geração de uma cobertura inicial aleatória, a execução do processo de limpeza e a geração de estados vizinhos à cobertura atual. Com essas sub-rotinas foi possível a adaptação do algoritmo, nomeado FlexMap-TS, ao problema. Como o foco do trabalho foi o mapeamento para FPGAs com o objetivo de redução de área, o FlexMap-TS buscou a minimização do número de *cortes-K* necessários para a cobertura do circuito.

O primeiro Cenário desenvolvido foi baseado no algoritmo clássico de Têmpera Simulada. A abordagem começou com temperaturas elevadas e realizou otimizações até que a variável temperatura atingisse um valor pré-determinado, próximo a zero. Apesar deste Cenário não conseguir superar os resultados obtidos pela ferramenta ABC (ABC, 2014), ele retornou resultados notáveis quando comparados à solução inicial aleatória em todos os *benchmarks* usados no experimento.

O Cenário 2 apresentado neste trabalho foi desenvolvido após uma análise dos resultados dos experimentos e do comportamento do TS no problema. Esta abordagem inicia sua execução com temperaturas menores, alterando a probabilidade de aceitação de soluções inferiores à atual. Neste Cenário foi também alterado a condição de parada do FlexMap-TS, sendo agora determinada por um numero máximo de ciclos de decaimento de temperatura sem que fosse encontrada uma uma cobertura que superasse a melhor obtida.

Os resultados obtidos neste Cenário foram superiores aos obtidos no Cenário 1, sendo alguns destes comparáveis e até melhores que a ferramenta ABC. Esses experimentos mostram que a técnica de Têmpera Simulada pode ser aplicada no Mapeamento Tecnológico e utilizada para melhorar ainda mais o desenvolvimento de circuitos integrados. O FlexMap-TS implementado no Cenário 2 conseguiu encontrar resultados tanto equivalentes como superiores ao ABC, superando a ferramenta em aproxi-

madamente 19% dos casos com *K=4* e 26% dos casos com *K=5*. Esses resultados mostram que a abordagem implementada consegue retornar resultados promissores e possui potencial para maiores estudos.

6.1 Trabalhos Futuros

Futuramente pretende-se expandir os experimentos para avaliar todos os benchmarks dos pacotes 'ISCAS85' (BRYAN, 1985; HANSEN; YALCIN; HAYES, 1999) e 'MCNC91' (YANG, 1991) para garantir uma melhor avaliação da técnica implementada. Pretende-se também realizar a implementação dos balanceamentos e associações existentes no ABC que fizeram com que a mesma obtivesse a diferença significativa nos resultados comparados ao FlexMap-TS, para que os resultados obtidos nesse trabalho também possam ser melhorados.

Um trabalho futuro que já está em desenvolvimento é a modificação do estado inicial do circuito. Ao invés da inicialização em um estado inicial aleatório, pretende-se utilizar a cobertura final proposta pelo ABC como ponto de partida do FlexMap-TS, buscando uma otimização ainda mais profunda. Para isso é necessário a modificação da estrutura de dados do FlexMap, que atualmente utiliza grafos AIG, para o padrão de saída do ABC, que é o padrão *EQN*, onde cada nodo representa uma equação equivalente a uma sub-função do circuito.

Por fim, pretende-se analisar outras técnicas de Inteligência Artificial, como por exemplo algoritmos de enxame, para verificar se é possível a adaptação destas para o problema.

REFERÊNCIAS

AARTS, E.; KORST, J. Simulated annealing and Boltzmann machines: a stochastic approach to combinatorial optimization and neural computing. **Wiley**, [S.l.], 1988.

ABC. A system for sequential synthesis and verification. **Berkeley Logic Synthesis and Verification Group**, [S.l.], 2014. http://www.eecs.berkeley.edu/˜alanmi/abc.

AIGER, f. **The AIGER And-Inverter Graph (AIG) Format**. Disponível em: http://fmv.jku.at/aiger/.

BERKELAAR, M.; JESS, J. Technology mapping for standard-cell generators. In: COMPUTER-AIDED DESIGN, 1988. ICCAD-88. DIGEST OF TECHNICAL PAPERS., IEEE INTERNATIONAL CONFERENCE ON, 1988. **Anais...** [S.l.: s.n.], 1988. p.470–473.

BERTSIMAS, D.; TSITSIKLIS, J. Simulated annealing. **Statistical Science**, [S.l.], p.10–15, 1993.

BRYAN, D. The ISCAS'85 benchmark circuits and netlist format. **North-Carolina State University**, [S.l.], 1985.

CHANG, K.-H.; BERTACCO, V.; MARKOV, I. L.; MISHCHENKO, A. Logic synthesis and circuit customization using extensive external don't-cares. **ACM Transactions on Design Automation of Electronic Systems (TODAES)**, [S.l.], v.15, n.3, p.26, 2010.

CHATTERJEE, S.; MISHCHENKO, A.; BRAYTON, R. Factor cuts. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 2006., 2006, New York, NY, USA. **Proceedings...** ACM, 2006. p.143–150. (ICCAD '06).

CHEN, D.; CONG, J. DAOmap: A depth-optimal area optimization mapping algorithm for FPGA designs. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 2004., 2004. **Proceedings...** [S.l.: s.n.], 2004. p.752–759.

CONG, J.; C., W.; Y., D. Cut Ranking and Pruning: Enabling A General and Efficient FPGA Mapping Solution. **INT'L SYMP. ON FPGA**, [S.l.], 1999.

CONG, J.; DING, Y. FlowMap: An optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.13, n.1, p.1–12, 1994.

CORREIA, V. R. A. Advanced technology mapping for standard-cell generators. **SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEM DESIGN, SBCCI**, [S.l.], v.17, p.254–259, 2004.

DAVIS, L. (Ed.). **Handbook of Genetic Algorithms**. [S.l.]: Van Nostrand Reingold, 1991.

DORIGO, M.; GAMBARDELLA, L. Ant colony system: A cooperative learning approach to the traveling salesman problem. **Evolutionary Computation, IEEE Transactions on**, [S.l.], v.1, n.1, p.53–66, 1997.

GOLDBERG, D. E. **Genetic Algorithms in Search, Optimization and Machine Learning**. [S.l.]: Addison-Wesley Publishing Company, 1989.

GOODSELL, D. S.; OLSON, A. J. Automated docking of substrates to proteins by simulated annealing. **Proteins: Structure, Function, and Bioinformatics**, [S.l.], v.8, n.3, p.195–202, 1990.

GREFENSTETTE, J.; GOPAL, R.; ROSMAITA, B.; VAN GUCHT, D. Genetic algorithms for the traveling salesman problem. In: INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS AND THEIR APPLICATIONS, 1985. **Proceedings...** [S.l.: s.n.], 1985. p.160–168.

HANSEN, M. C.; YALCIN, H.; HAYES, J. P. Unveiling the ISCAS-85 benchmarks: A case study in reverse engineering. **Design & Test of Computers, IEEE**, [S.l.], v.16, n.3, p.72–80, 1999.

HENTSCHKE, R. F. **Algoritmos para o Posicionamento de Células em Circuitos VLSI**. 2002. Tese (Doutorado em Ciência da Computação) — UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL.

KAO, C.-C.; LAI, Y.-T. An efficient algorithm for finding the minimal-area FPGA technology mapping. **ACM Transactions on Design Automation of Electronic Systems (TODAES)**, [S.l.], v.10, n.1, p.168–186, 2005.

KEUTZER, K. DAGON: Technology Binding and Local Optimization by DAG Matching. In: DESIGN AUTOMATION, 1987. 24TH CONFERENCE ON, 1987. **Anais...** [S.l.: s.n.], 1987. p.341 – 347.

KOMMU, V.; POMERANZ, I. GAFPGA: Genetic algorithm for FPGA technology mapping. In: DESIGN AUTOMATION CONFERENCE, 1993, WITH EURO-VHDL'93. PROCEEDINGS EURO-DAC'93. EUROPEAN, 1993. **Anais...** [S.l.: s.n.], 1993. p.300–305.

KUKIMOTO, Y.; BRAYTON, R. K.; SAWKAR, P. Delay-optimal technology mapping by DAG covering. In: DESIGN AUTOMATION CONFERENCE, 35., 1998. **Proceedings...** [S.l.: s.n.], 1998. p.348–351.

LIU, Y.-H. Different initial solution generators in genetic algorithms for solving the probabilistic traveling salesman problem. **Applied mathematics and computation**, [S.l.], v.216, n.1, p.125–137, 2010.

MANOHARARAJAH, V.; BROWN, S. D.; VRANESIC, Z. G. Heuristics for area minimization in LUT-based FPGA technology mapping. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.25, n.11, p.2331–2340, 2006.

MARQUES, F.; ROSA JR, L.; RIBAS, R.; SAPATNEKAR, S.; REIS, A. DAG based library-free technology mapping. In: ACM GREAT LAKES SYMPOSIUM ON VLSI, 17., 2007. **Proceedings...** [S.l.: s.n.], 2007. p.293–298.

MARQUES, F. S. **Technology mapping for virtual libraries based on cells with minimal transistor stacks**. 2008. Dissertação (Mestrado em Ciência da Computação) — Programa de Pós-Graduação em Ciência da Computação, Universidade Federal do Rio Grande do Sul.

MARQUES, F. S.; RIBAS, R. P.; SAPATNEKAR, S.; REIS, A. I. A new approach to the use of satisfiability in false path detection. In: ACM GREAT LAKES SYMPOSIUM ON VLSI, 15., 2005. **Proceedings...** [S.l.: s.n.], 2005. p.308–311.

MELANIE, M. An introduction to genetic algorithms. **Cambridge, Massachusetts London, England, Fifth printing**, [S.l.], 1999.

MINTZ, A.; GOLUMBIC, M. C. Factoring Boolean functions using graph partitioning. **Discrete Applied Mathematics**, [S.l.], v.149, n.1, p.131–153, 2005.

MISHCHENKO, A.; CHATTERJEE, S.; BRAYTON, R. K. Improvements to technology mapping for LUT-based FPGAs. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.26, n.2, p.240–253, 2007.

MISHCHENKO, A.; CHATTERJEE, S.; BRAYTON, R.; WANG, X.; KAM, T. Technology mapping with Boolean matching, supergates and choices. **Berkeley Logic Synthesis and Verification Group**, [S.l.], 2005.

PANDEY, R.; CHATTOPADHYAY, S. Low power technology mapping for LUT based FPGA-a genetic algorithm approach. In: VLSI DESIGN, 2003. PROCEEDINGS. 16TH INTERNATIONAL CONFERENCE ON, 2003. **Anais...** [S.l.: s.n.], 2003. p.79–84.

PERTTUNEN, J. On the significance of the initial solution in travelling salesman heuristics. **Journal of the Operational Research Society**, [S.l.], p.1131–1140, 1994.

POND, S. L. K.; POSADA, D.; GRAVENOR, M. B.; WOELK, C. H.; FROST, S. D. GARD: a genetic algorithm for recombination detection. **Bioinformatics**, [S.l.], v.22, n.24, p.3096–3098, 2006.

RAHMAT-SAMII, Y.; MICHIELSSEN, E. **Electromagnetic optimization by genetic algorithms**. [S.l.]: John Wiley & Sons, Inc., 1999.

SCHNEIDER, F. R.; RIBAS, R. P.; SAPATNEKAR, S. S.; REIS, A. I. Exact lower bound for the number of switches in series to implement a combinational logic cell. In: COMPUTER DESIGN: VLSI IN COMPUTERS AND PROCESSORS, 2005. ICCD 2005. PROCEEDINGS. 2005 IEEE INTERNATIONAL CONFERENCE ON, 2005. **Anais...** [S.l.: s.n.], 2005. p.357–362.

SENTOVICH, E. M.; SINGH, K. J.; LAVAGNO, L.; MOON, C.; MURGAI, R.; SALDANHA, A.; SAVOJ, H.; STEPHAN, P. R.; BRAYTON, R. K.; SANGIOVANNI-VINCENTELLI, A. **SIS**: A system for sequential circuit synthesis. [S.l.]: Citeseer, 1992.

SOUZA, V.; SILVA-FILHO, A. MogaMap: An application of multi-objective genetic algorithm for LUT-based FPGA technology mapping. In: ELECTRONICS, CIRCUITS, AND SYSTEMS (ICECS), 2013 IEEE 20TH INTERNATIONAL CONFERENCE ON, 2013. **Anais...** [S.l.: s.n.], 2013. p.485–488.

STOK, L.; IYER, M.; SULLIVAN, A. Wavefront technology mapping. In: DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE AND EXHIBITION 1999. PROCEEDINGS, 1999. **Anais...** [S.l.: s.n.], 1999. p.531–536.

SVERGUN, D. Restoring low resolution structure of biological macromolecules from solution scattering using simulated annealing. **Biophysical journal**, [S.l.], v.76, n.6, p.2879–2886, 1999.

TESLENKO, M.; DUBROVA, E. Hermes: LUT FPGA technology mapping algorithm for area minimization with optimum depth. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 2004., 2004. **Proceedings...** [S.l.: s.n.], 2004. p.748–751.

TJIANG, S. W. **Twig reference manual**. [S.l.]: AT and T Bell Laboratories. Computing Science, 1986.

VAN LAARHOVEN, P. J.; AARTS, E. H. **Simulated annealing**. [S.l.]: Springer, 1987.

YANG, S. **Logic synthesis and optimization benchmarks user guide**: version 3.0. [S.l.]: Microelectronics Center of North Carolina (MCNC), 1991.

ANEXO A TABELAS COM OS VALORES DOS EXPERIMENTOS

Tabela 4: Cenário 1 (***K=4***)

Benchmark	**ABC**	**FlexMap-TS (Média)**	**Desvio Padrão**	**Diferença (%)**	**Normalização**
C432	96	79,13	1,17	-16,87 (-17,57%)	-0,18
b1	2	2	0	0 (0%)	0,00
C17	2	2	0	0 (0%)	0,00
cm150a	13	13	0	0 (0%)	0,00
cm151a	8	8	0	0 (0%)	0,00
cm42a	11	11	0	0 (0%)	0,00
subfrg1	2	2	0	0 (0%)	0,00
my_adder	42	42,88	2,54	0,88 (2,05%)	0,02
C1355	79	82,03	2,71	3,03 (3,69%)	0,04
C499	79	82,1	3,32	3,09 (3,77%)	0,04
unreg	33	34,88	2,66	1,88 (5,38%)	0,06
C880	115	123,5	2,71	8,5 (6,88%)	0,07
tcon	8	8,64	2,13	0,64 (7,4%)	0,08
9symml	80	88,06	2,59	8,06 (9,15%)	0,10
cm85a	15	16,74	2,32	1,74 (10,41%)	0,12
pcler8	31	35,2	1,24	4,2 (11,93%)	0,14
cm162a	15	17,22	1,79	2,22 (12,89%)	0,15
9sym	145	166,62	3,69	21,62 (12,97%)	0,15
parity	5	5,85	1,06	0,85 (14,52%)	0,17
C1908	114	134,47	3,48	20,47 (15,22%)	0,18
count	37	44,77	1,4	7,77 (17,35%)	0,21
cm82a	4	4,88	1,51	0,88 (18,03%)	0,22
cm152a	9	11	1,46	2 (18,18%)	0,22
b9	41	50,62	3,07	9,62 (19,01%)	0,23
cu	19	23,52	3,75	4,52 (19,21%)	0,24
sqrt8ml	16	19,97	1,68	3,96 (19,87%)	0,25
decod	18	22,5	1,92	4,5 (20%)	0,25
C3540	355	454,13	3,7	99,13 (21,82%)	0,28
f51m	72	92,45	3,56	20,45 (22,12%)	0,28
subcmb	4	5,24	1,26	1,24 (23,66%)	0,31
apex6	243	321,38	1,66	78,38 (24,38%)	0,32
majority	4	5,32	1,72	1,32 (24,81%)	0,33
C2670	176	236,7	1,83	60,7 (25,64%)	0,34
misex1	20	26,93	2,95	6,93 (25,74%)	0,35
x2	18	24,32	2,47	6,32 (25,98%)	0,35
comp	38	51,95	2,44	13,94 (26,84%)	0,37
cm163a	12	16,44	3,61	4,44 (27%)	0,37
cm138a	9	12,51	3,64	3,51 (28,05%)	0,39
pcle	21	29,21	3,84	8,21 (28,1%)	0,39
sqrt8	33	46,6	1,13	13,59 (29,17%)	0,41
i1	16	23,12	1,53	7,12 (30,79%)	0,45
misex2	42	60,77	1,15	18,76 (30,88%)	0,45

Tabela 5: Cenário 1 (***K=4***) (Cont.)

Benchmark	ABC	FlexMap-TS (Média)	Desvio Padrão	Diferença (%)	Normalização
cmb	17	25,54	2,28	8,53 (33,42%)	0,50
alu2	181	281,69	3,15	100,69 (35,74%)	0,56
con1	5	7,81	1,83	2,8 (35,95%)	0,56
C5315	432	677,97	11,53	245,97 (36,28%)	0,57
apex7	79	124,03	2,19	45,02 (36,3%)	0,57
ldd	36	57,28	3,88	21,27 (37,14%)	0,59
o64	44	71,68	1,57	27,68 (38,61%)	0,63
table5	400	661,45	2,99	261,44 (39,52%)	0,65
i9	247	410,94	2,9	163,94 (39,89%)	0,66
duke2	208	347,59	2,24	139,59 (40,15%)	0,67
vda	328	559,05	3,14	231,04 (41,32%)	0,70
cc	17	29,9	2,61	12,9 (43,14%)	0,76
pm1	17	29,9	3,7	12,9 (43,14%)	0,76
table3	435	804,65	3,72	369,64 (45,93%)	0,85
C6288	506	938,17	21,01	432,17 (46,06%)	0,85
z4ml	48	90,23	1,75	42,22 (46,8%)	0,88
rd84	211	401,03	3,68	190,03 (47,38%)	0,90
5xp1	51	98,54	1,93	47,54 (48,24%)	0,93
cht	38	73,73	1,81	35,72 (48,46%)	0,94
sao2	79	156,12	2,78	77,11 (49,39%)	0,98
lal	29	61,41	1,76	32,41 (52,77%)	1,12
rot	245	522,32	2,36	277,31 (53,09%)	1,13
b12	49	108,14	3,31	59,13 (54,68%)	1,21
struct1	28	62,19	2,96	34,19 (54,97%)	1,22
x4	149	333,22	1,27	184,22 (55,28%)	1,24
vg2	139	338,31	3,52	199,3 (58,91%)	1,43
c8	40	99,79	2,38	59,79 (59,91%)	1,49
x3	216	541,45	3,53	325,45 (60,1%)	1,51
e64	266	709,37	3,9	443,36 (62,5%)	1,67
x1	228	623,1	1,43	395,1 (63,4%)	1,73
rd73	82	243,86	3,67	161,85 (66,37%)	1,97
sct	22	68,33	1,42	46,33 (67,8%)	2,11
frg1	158	508,33	1,59	350,32 (68,91%)	2,22
squar5	18	60,31	1,95	42,31 (70,15%)	2,35
clip	190	675,58	3,57	485,58 (71,87%)	2,56
rd53	15	55,34	2,42	40,33 (72,89%)	2,69
mux	16	62,29	3,15	46,28 (74,31%)	2,89
ttt2	66	266,33	3,45	200,32 (75,21%)	3,04
term1	54	285,41	2,99	231,41 (81,07%)	4,29
t481	123	860,73	2,13	737,72 (85,7%)	6,00
xor5	2	28,27	2,34	26,27 (92,92%)	13,14

Tabela 6: Cenário 2 (***K=4***)

Benchmark	**ABC**	**FlexMap-TS (Média)**	**Desvio Padrão**	**Diferença (%)**	**Normalização**
C432	96	67,60	0,72	-28,4(-29,58%)	-0,29
my_adder	42	32,00	0,00	-10 (-23,80%)	-0,24
cm85a	15	12,13	0,35	-2,86 (-19,13%)	-0,19
cm151a	8	7,00	0,00	-1 (-12,50%)	-0,13
cm42a	11	10,00	0,00	-1 (-9,09%)	-0,09
cm162a	15	14,00	0,00	-1 (-6,66%)	-0,07
C499	79	76,06	1,11	-2,93 (-3,72%)	-0,04
C1355	79	76,13	1,31	-2,86 (-3,63%)	-0,04
C880	115	111,46	1,31	-3,53 (-3,07%)	-0,03
unreg	33	32,00	0,00	-1 (-3,03%)	-0,03
9symml	80	77,93	0,78	-2,06 (-2,58%)	-0,03
C2670	176	174,76	2,51	-1,23 (-0,7%)	-0,01
f51m	72	71,66	0,80	-0,33 (-0,46%)	-0,01
b1	2	2,00	0,00	0 (0%)	0,00
C17	2	2,00	0,00	0 (0%)	0,00
cm138a	9	9,00	0,00	0 (0%)	0,00
cm163a	12	12,00	0,00	0 (0%)	0,00
cm82a	4	4,00	0,00	0 (0%)	0,00
count	37	37,00	0,00	0 (0%)	0,00
decod	18	18,00	0,00	0 (0%)	0,00
majority	4	4,00	0,00	0 (0%)	0,00
parity	5	5,00	0,00	0 (0%)	0,00
subcmb	4	4,00	0,00	0 (0%)	0,00
subfrg1	2	2,00	0,00	0 (0%)	0,00
tcon	8	8,00	0,00	0 (0%)	0,00
misex1	20	20,10	0,31	0,1 (0,49%)	0,01
C6288	506	509,14	3,68	3,13 (0,61%)	0,01
apex6	243	249,13	1,48	6,13 (2,46%)	0,03
pcler8	31	32,00	0,00	1 (3,12%)	0,03
misex2	42	44,03	0,18	2,03 (4,61%)	0,05
9sym	145	152,86	1,28	7,86 (5,14%)	0,05
x2	18	19,00	0,00	1 (5,26%)	0,06
i1	16	17,00	0,00	1 (5,88%)	0,06
sqrt8ml	16	17,06	0,25	1,06 (6,25%)	0,07
sqrt8	33	35,30	0,65	2,3 (6,51%)	0,07
C3540	355	381,36	2,28	26,36 (6,91%)	0,07
C1908	114	123,00	1,29	9 (7,31%)	0,08
pcle	21	23,00	0,00	2 (8,69%)	0,10
b9	41	45,20	0,41	4,2 (9,29%)	0,10
cu	19	21,00	0,00	2 (9,52%)	0,11

Tabela 7: Cenário 2 (***K=4***) (Cont.)

Benchmark	ABC	FlexMap-TS (Média)	Desvio Padrão	Diferença (%)	Normalização
cm152a	9	10,00	0,00	1 (10%)	0,11
comp	38	42,23	0,50	4,23 (10,02%)	0,11
C5315	432	487,53	11,12	55,53 (11,39%)	0,13
cmb	17	19,20	0,41	2,2 (11,45%)	0,13
cm150a	13	15,00	0,00	2 (13,33%)	0,15
ldd	36	43,06	0,25	7,06 (16,4%)	0,20
alu2	181	218,36	1,45	37,36 (17,11%)	0,21
vda	328	417,20	3,41	89,2 (21,38%)	0,27
apex7	79	102,50	0,86	23,5 (22,92%)	0,30
duke2	208	275,86	1,98	67,86 (24,6%)	0,33
pm1	17	23,00	0,00	6 (26,08%)	0,35
z4ml	48	67,33	0,88	19,33 (28,71%)	0,40
con1	5	7,03	0,18	2,03 (28,9%)	0,41
o64	44	64,00	0,00	20 (31,25%)	0,45
5xp1	51	75,80	0,92	24,8 (32,71%)	0,49
table5	400	595,90	3,01	195,9 (32,87%)	0,49
cc	17	26,00	0,00	9 (34,61%)	0,53
i9	247	380,50	2,45	133,5 (35,08%)	0,54
rot	245	386,90	3,79	141,9 (36,67%)	0,58
table3	435	687,73	3,13	252,73 (36,74%)	0,58
sao2	79	125,90	1,09	46,9 (37,25%)	0,59
cht	38	60,93	0,37	22,93 (37,63%)	0,60
struct1	28	46,06	0,83	18,06 (39,21%)	0,65
b12	49	80,70	1,26	31,7 (39,28%)	0,65
rd84	211	358,06	3,69	147,06 (41,07%)	0,70
x4	149	254,36	1,97	105,36 (41,42%)	0,71
vg2	139	254,36	1,22	115,36 (45,35%)	0,83
lal	29	53,40	0,67	24,4 (45,69%)	0,84
x3	216	416,50	3,18	200,5 (48,13%)	0,93
e64	266	514,03	1,22	248,03 (48,25%)	0,93
x1	228	465,00	2,59	237 (50,96%)	1,04
c8	40	89,10	1,67	49,1 (55,1%)	1,23
rd73	82	201,53	1,91	119,53 (59,31%)	1,46
sct	22	54,23	0,73	32,23 (59,43%)	1,47
clip	190	504,16	2,55	314,16 (62,31%)	1,65
rd53	15	40,10	0,88	25,1 (62,59%)	1,67
frg1	158	434,46	2,01	276,46 (63,63%)	1,75
mux	16	45,46	1,20	29,46 (64,8%)	1,84
ttt2	66	190,23	2,34	124,23 (65,3%)	1,88
squar5	18	55,33	1,15	37,33 (67,46%)	2,07
term1	54	224,73	2,73	170,73 (75,97%)	3,16
t481	123	723,30	0,75	600,3 (82,99%)	4,88
xor5	2	23,36	0,49	21,36 (91,44%)	10,68

Tabela 8: Cenário 2 (***K=5***)

Benchmark	**ABC**	**FlexMap-TS (Média)**	**Desvio Padrão**	**Diferença(%)**	**Normalização**
cm85a	14	10	0,00	-4 (-28,57%)	-0,29
subcmb	4	3	0,00	-1 (-25,00%)	-0,25
C432	81	61	1,58	-20 (-24,69%)	-0,25
C6288	693	529	1,22	-164 (-22,74%)	-0,24
f51m	51	39,4	0,55	-11,6 (-15,38%)	-0,23
cm150a	13	11	0,00	-2 (-13,33%)	-0,15
cm151a	6	5,2	0,45	-0,8 (-13,15%)	-0,13
o64	38	33	0,00	-5 (-12,85%)	-0,13
cmb	14	12,2	0,45	-1,8 (-12,50%)	-0,13
i1	16	14	0,00	-2 (-9,20%)	-0,13
misex1	17	15	0,00	-2 (-8,68%)	-0,12
C880	100	90,8	1,30	-9,2 (-7,78%)	-0,09
C2670	145	132,4	1,67	-12,6 (0%)	-0,09
C3540	280	258,2	4,66	-21,8 (-23,66%)	-0,08
C5315	346	338,8	3,03	-7,1 (-2,08%)	-0,02
i9	237	233,6	5,90	-3,4 (-1,43%)	-0,01
C1908	106	104,6	2,07	-1,4 (-1,32%)	-0,01
majority	1	1	0,00	0 (0%)	0,00
subfrg1	1	1	0,00	0 (0%)	0,00
xor5	1	1	0,00	0 (0%)	0,00
b1	2	2	0,00	0 (0%)	0,00
C17	2	2	0,00	0 (0%)	0,00
cm82a	3	3	0,00	0 (0%)	0,00
con1	3	3	0,00	0 (0%)	0,00
rd53	3	3	0,00	0 (0%)	0,00
parity	5	5	0,00	0 (0%)	0,00
squar5	8	8	0,00	0 (0%)	0,00
tcon	8	8	0,00	0 (0%)	0,00
cm138a	9	9	0,00	0 (0%)	0,00
cm163a	9	9	0,00	0 (0%)	0,00
cm42a	10	10	0,00	0 (0%)	0,00
cm162a	11	11	0,00	0 (0%)	0,00
decod	16	16	0,00	0 (0%)	0,00
pcle	16	16	0,00	0 (0%)	0,00
cc	21	21	0,00	0 (0%)	0,00
pcler8	27	27	0,00	0 (0%)	0,00
count	31	31	0,00	0 (0%)	0,00
unreg	32	32	0,00	0 (0%)	0,00
misex2	35	35	0,00	0 (0%)	0,00
9sym	116	116	0,00	0 (0%)	0,00
comp	33	33,2	1,10	0,2 (0,6%)	0,01

Tabela 9: Cenário 2 (***K=5***) (Cont.)

Benchmark	ABC	FlexMap-TS (Média)	Desvio Padrão	Diferença (%)	Normalização
sqrt8	24	24,2	0,45	0,1 (0,82%)	0,01
9symml	57	57,6	0,89	0,6 (1,04%)	0,01
C499	66	67,4	1,67	1,4 (2,07%)	0,02
my_adder	24	24,8	0,45	0,8 (3,22%)	0,03
pm1	17	17,6	0,55	0,6 (3,4%)	0,04
C1355	66	68,4	0,89	2,4 (3,5%)	0,04
cu	15	16	0,00	1 (6,25%)	0,07
apex6	192	205,5	3,11	13,5 (6,56%)	0,07
x2	14	15	0,00	1 (6,66%)	0,07
b9	36	39	0,00	3 (7,69%)	0,08
cht	39	42,6	0,55	3,6 (8,45%)	0,09
frg1	140	161,8	1,64	21,8 (13,47%)	0,16
alu2	147	172,4	1,14	25,4 (14,73%)	0,17
ldd	27	33	0,00	6 (18,18%)	0,22
clip	159	198,4	1,14	39,4 (19,85%)	0,25
apex7	64	80	1,00	16 (20%)	0,25
duke2	180	226,2	2,17	46,2 (20,42%)	0,26
vda	277	355,8	2,95	78,8 (22,14%)	0,28
rot	229	302,6	2,41	73,6 (24,32%)	0,32
sqrt8ml	9	12	0,00	3 (25%)	0,33
table5	352	475,2	1,10	123,2 (25,92%)	0,35
5xp1	34	47,4	0,55	13,4 (28,27%)	0,39
cm152a	5	7	0,00	2 (28,57%)	0,40
sao2	71	100,2	1,10	29,2 (29,14%)	0,41
z4ml	33	46,6	0,55	13,6 (29,18%)	0,41
x3	181	260	3,08	79 (30,38%)	0,44
b12	38	55,8	0,45	17,8 (31,89%)	0,47
lal	28	41,8	0,45	13,8 (33,01%)	0,49
struct1	23	35	1,22	12 (34,28%)	0,52
table3	353	545	2,00	192 (35,22%)	0,54
vg2	121	188	1,41	67 (35,63%)	0,55
x4	111	191,2	1,48	80,2 (41,94%)	0,72
e64	225	396,4	0,89	171,4 (43,23%)	0,76
x1	204	360	3,32	156 (43,33%)	0,76
mux	11	21	0,00	10 (47,61%)	0,91
rd84	168	321,2	15,85	153,2 (47,69%)	0,91
c8	32	64,8	1,30	32,8 (50,61%)	1,03
sct	21	42,8	0,84	21,8 (50,93%)	1,04
rd73	53	136,8	1,48	83,8 (61,25%)	1,58
ttt2	50	131	2,92	81 (61,83%)	1,62
term1	48	149	1,00	101 (67,78%)	2,10
t481	108	592	3,16	484 (81,75%)	4,48