UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

MODELO DE DESENVOLVIMENTO INTELECTUAL PARA AGENTES ROBÓTICOS

Rosiery da Silva Maia

Orientador: Prof. Dr. Luiz Marcos Garcia Gonçalves

Tese apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e de Computação da UFRN (área de concentração: Engenharia de Computação) como parte dos requisitos para obtenção do título de Doutor em Ciências

Número de ordem PPgEEC: D086
Natal/RN, Dezembro de 2012

Catalogação da publicação na fonte.

M217m Maia, Rosiery da Silva

Modelo de Desenvolvimento Intelectual para Agentes Robóticos / Rosiery da Silva Maia. Natal: UFRN, 2015.

113 f.

Orientador(a): Prof. Dr. Luiz Marcos Garcia Gonçalves

Tese (Doutorado). Universidade Federal do Rio Grande do Norte. Centro de Tecnologia. Programa de Pós-Graduação em Engenharia Elétrica e de Computação.

1.Sistema Multirrobô. 2.Modelo Social de Aprendizagem. 3.Cooperação. I. Gonçalves, Luiz Marcos Garcia. II. Universidade Federal do Rio Grande do Norte. III. Título.

FDHS/BC CDU: 004.896

CDD: 004.6

Bibliotecário: Sebastião Lopes Galvão Neto - CRB 15/486

MODELO DE DESENVOLVIMENTO INTELECTUAL PARA AGENTES ROBÓTICOS

Rosiery da Silva Maia

Tese de Doutorado aprovada em 18 de dezembro de 2012 pela banca examinadora composta pelos seguintes membros:

Prof. Dr. Luiz Marcos Garcia Gonçalves
(Orientador, UFRN)

Profª Drª Silvia Silva da Costa Botelho
(Examinadora Externa, UFRGS)

Prof. Dr. Anderson Abner de S. Souza
(Examinador Externo, UERN)

Prof. Dr. Daniel Aloise
(Examinador Interno, UFRN)

Ao meu esposo, Marcos Swami,
às minhas filhas, Lara e Bruna,
e aos meus pais, Oswaldo e Telma.

Agradecimentos

Meus sinceros agradecimentos:

- ao professor Luiz Marcos, pela orientação durante todo o período da pesquisa;
- ao professor Daniel Aloise pelas contribuições durante os experimentos;
- aos amigos Lourena Rocha, Anderson Abner e André Macedo, pelas importantes sugestões durante o produtivo horário do café;
- aos colegas do Laboratório NatalNet pelas ajudas oferecidas;
- a UERN, pela liberação em tempo integral durante o período solicitado.

Resumo

IDeM-MRS é um modelo com regras formais para gerenciar a cooperação de um grupo de robôs, quando aplicados em resolução de tarefas. Esse formalismo é totalmente baseado em modelos sociais de aprendizagem de indivíduos reais e coordena eficientemente o sistema multirrobô, propiciando a Assimilação e Acomodação do conhecimento através de trocas de experiências entre os componentes do grupo. Algumas questões foram particulamente tratadas, como uma representação realista do ambiente multirrobô (que envolve a missão global, as tarefas que pertencem a essa missão e os robôs atuantes) e uma forma de seleção de tarefas, baseada em teorias de abordagens socias de aprendizagem, como as pregadas por Lev Vygotsky, Jean Piaget e John Watson. Essas teorias, já consolidadas para indivíduos reais, permitem a execução cooperativa eficiente pelos robôs. IDeM-MRS pode ser usado em diferentes tipos de missões, desde as mais simples às mais complexas. Os experimentos e resultados validam a eficiência dessa proposta em comparação a um modelo empírico tradicional.

Palavras-chave: Sistema Multirrobô, Modelo Social de Aprendizagem, Cooperação.

Abstract

IDeM-MRS is a model with formal rules to manage the cooperation of multi-robot systems, if it is applied in tasks execution. This formalism is completely based on learning social models of real individuals and efficiently coordinates the group allowing knowledge assimilation and accommodation through experiences exchange among the group components. Some issues are particularly investigated, such as the realist multi-robot environment representation (involving the global mission, the tasks that belong to this mission and the active robots) and a task selection form based on theories and social learning approaches, such as theories of Lev Vygotsky, Jean Piaget and John Watson. Its theories already consolidated to real individuos, allows the cooperative execution efficient by robots. IDeM-MRS can be used for different types of missions, from the simplest to more complex ones. The experiments and results validate the effectiveness of this formalism compared to the traditional empirical model.

Keywords: Multi-robot Systems, Learning Social Models, Cooperation.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Abreviaturas

CETP	*Cooperative Execution Task Problem*
CMAP	*Cooperative Mission Achievement Problem*
CTSP	*Coordinated Task Selection Problem*
IDeM-MRS	*Intellectual Development Model for MultiRobot Systems*
LPD	*Level of Potential Development*
LRD	*Level of Real Development*
LSM	*Learning Social Models*
MRS	*Multi-Robot Systems*
NP-Hard	*Non-deterministic Polynomial-time Hard*
OR	*Operational Research*
RCPSP	*Resource Constrained Project Scheduling Problem*
ZPD	*Zone of Proximal Development*

Lista de Definições

LPD_i	Nível de Desenvolvimento Potencial do indivíduo i
LRD_i	Nível de Desenvolvimento Real do indivíduo i
ZPD_i	Zona de Desenvolvimento Proximal do indivíduo i
α_i	Agente robótico i
τ_j	Tarefa j
y	Valor numérico $y = \{0,1\}$
w	Valor numérico $w = \{-1,1\}$
$C(\alpha_i)$	Conhecimento do robô i
$C(\tau_j)$	Conhecimento requerido pela tarefa j
$F(\alpha_i)$	Características físicas do robô i
$F(\tau_j)$	Características físicas requeridas pela tarefa j
c'	Conhecimento novo específico
c	Conhecimento atual específico
h	Valor numérico $h = \{0,1\}$
K	Conjunto específico de conhecimento $K = \{c_1, c_2, ..., c_k\}$
k	Valor numérico $k = \{0,1\}$
A'	Conjunto de robôs do ambiente
m	Quantidade de robôs no ambiente
n	Quantidade de tarefas requisitadas no ambiente
n_1	Quantidade de tarefas resolvidas da missão

Lista de Equações

$ZPD_i = LPD_i - LRD_i$	ZPD do indivíduo i
$f\_lrd(\alpha_i, \tau_j) = y$	LRD do robô i em relação à tarefa j
$f\_lpd(\alpha_i, \tau_j) = -w$	LPD do robô i em relação à tarefa j
$f\_zpd(\alpha_i, \tau_j)$	ZPD do robô i em relação à tarefa j
$C(\alpha_i) = C(\alpha_i) \cup \{c'\}$	Assimilação unitária do robô i
$f\_assimilation(\alpha_i, \alpha_x, \tau_j) = h$	Assimilação do robô i, sobre j, através de x
$C(\alpha_i) = \{C(\alpha_i) - \{c\}\} \cup \{c'\}$	Acomodação unitária do robô i
$f\_accommodation(\alpha_i, c, c') = s$	Acomodação do robô i,acerca de c e c'
$f\_\varepsilon_c(\tau_j, \alpha_i) = K$	Estímulo-conhecimento do robô i sobre j
$f\_\varepsilon_r(\tau_j, \alpha_i) = k$	Estímulo-Resposta do robô i sobre a tarefa j
$f\_\varepsilon_{gr}(\tau_j, A') = k$	Estímulo-Grupo dos robôs A' sobre a tarefa j
$A = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$	Conjunto de robôs do ambiente
$T = \{\tau_1, \tau_2, \ldots, \tau_n\}$	Conjunto de tarefas requisitadas no ambiente
$M = \{\tau_i, \tau_k, \tau_j\}$	Missão do ambiente
$fmission\_time = \sum_{k=1}^{n} time(\tau_k)$	Tempo de execução de uma missão
$C(\tau_j) = \{c_1, c_2, \ldots, c_k\}$	Conhecimento requerido pela tarefa j
$F(\tau_j) = \{f_1, f_2, \ldots, f_w\}$	Característica física requerida pela tafefa j
$C(\alpha_i) = \{c_1, c_2, \ldots, c_x\}$	Conhecimento do robô i
$F(\alpha_i) = \{f_1, f_2, \ldots, f_y\}$	Característica física do robô i
$ftask_{exec} = \frac{m_1.100}{m}$	Percentual de tarefas executadas pelos robôs

Capítulo 1

Introdução

Esta tese propõe um modelo de desenvolvimento cognitivo artificial para melhorar o aprendizado de agentes robóticos, em grupo, visando a execução de tarefas de forma cooperativa. Para tal, esta proposta formaliza um modelo matemático de desenvolvimento intelectual, denominado IDeM-MRS (***I**ntellectual **De**velopment **M**odel for **M**ulti**R**obot **S**ystems*) que permite o aprendizado dos indivíduos baseado em interações com o grupo.

O modelo proposto é inspirado em teorias de aprendizagem (social) de indivíduos reais de teóricos tradicionais da literatura, tais como Lev Semenovich Vygotsky [Vygotsky 1967, Vygotsky 1978, Vygotsky 1991, Vygotsky 1993], Jean William Fritz Piaget [Piaget 1975, Piaget & Inhelder 1982, J.Piaget 1985] e John Broadus Watson [Watson 1913, Wenger 2000]. Essas teorias são analisados neste trabalho com a finalidade de mapear os principais conceitos e suas possíveis aplicações em sistemas multirrobôs. Para validar e testar o formalismo do IDeM-MRS, ele é implementado em um ambiente constituído por vários agentes robóticos que devem cooperar em função de um objetivo global.

O problema global tratado nesta tese pode ser relatado como um problema de execução cooperativa de tarefas por um grupo de agentes robóticos, sendo este problema maior particionado em partes distribuídas de problemas de seleção coordenada de tarefas para cada robô componente do grupo. Assim, este trabalho envolve investigações de questões importantes relacionadas ao problema de execução e alocação de tarefas por sistemas multirrobôs autônomos, propondo uma abordagem que influencie positivamente na cooperação do grupo, agregando benefícios ao ambiente.

Aplicações com robôs móveis e autônomos exigem de alto grau de flexibilidade, adaptabilidade e eficiência em seus resultados, especialmente em tarefas que envolvem cooperação entre o grupo. Os sistemas multirrobôs (*Multi-Robot Systems* - MRS), assim como são chamados o grupo de robôs cuja tarefa global é comum a todos que interagem no ambiente [Serment et al. 2002], possuem como característica principal a cooperação em grupo para alcançar um objetivo global e específico.

Esses sistemas são adequados para domínios em que a realização da missão não é possível com um único robô, ou para situações em que a coordenação de mais de um robô contribui para a eficiência do sistema global. Outra razão para usar MRS são as exigências da distribuição de tarefas da solução. A estrutura do domínio do problema pode possuir partes inerentemente distribuídas no espaço, no tempo, ou de acordo com a funcionalidade. Alguns exemplos usuais de aplicações (ou missões) nesses domínios, são:

futebol de robôs; pesquisa cooperativa multi-objetivo; operações perigosas de limpeza, limpeza de resíduos, limpeza de minas (em solo e em mar); e exploração planetária.

Algumas aplicações contém mais de uma dessas missões. Os domínios também podem ser adversos, tal como o futebol de robôs, que um time de robôs compete com outro. O foco deste trabalho está em sistemas multirrobôs cooperativos. A proposta, no entanto, pode ser adequada em sistemas adversos, interpretando a competição como restrições do ambiente em um domínio cooperativo.

Segundo Talay et al. (2011), embora muitas arquiteturas tenham sido propostas para dar suporte à cooperação (e também à coordenação) multirrobótica, o campo ainda continua, desde seu estado inicial, com uma grande quantidade de questões em aberto. É verdade que essas questões ocorrem atualmente, muito embora alguns desses problemas tenham sido solucionados em pesquisas que envolvem um único robô. Algumas questões de cooperação multirrobótica que este trabalho destaca são:

- Como as tarefas devem ser representadas para uma cooperação efetiva? Como a prioridade de cada uma é definida?
- Qual robô deve executar qual ordem de tarefa para, cooperativamente, encontrar a solução global?
- Qual a função de cada robô do grupo, para adequar uma cooperação mais eficiente?
- Qual o tipo de ligação que cada componente do grupo deve ter? Qual a forma mais eficiente de comunicação para o ambiente?
- Quais informações devem ser compartilhadas entre o grupo, quando há limitações no ambiente em que atuam, seja na comunicação ou no *hardware* de cada componente?
- Pode existir algum tipo de contingência na aplicação? Se sim, como detectá-la e escapar dela?
- A cooperação multirrobótica depende da coordenação definida para o grupo? A primeira característica existe sem a segunda? O contrário é verdade? O estudo de cooperação remete a investigações de coordenação?
- Pensando na questão anterior, como efetivar uma cooperação, sem que haja algum robô com função de coordenação ou líder? Quais características o robô, em condição de liderança, deve ter?
- Caso haja um líder no grupo, como substituí-lo quando necessário? Qual outro robô poderia desempenhar essa função de forma que não haja perdas na eficiência da cooperação do grupo?

Especificamente no contexto de sistemas multirrobôs direcionados para resolução de tarefas, através de uma cooperação adequada pode-se obter uma boa relação de trocas de conhecimento e experiência entre o grupo, e com isso, possibilitar a inclusão de conhecimento individualmente, acerca de cada tarefa específica.

A dinâmica da execução de tarefas por um grupo de robôs é baseada, mas não limitada, no planejamento de ações a serem executadas prioritariamente. Porém, o problema de alocação de tarefas não é simples de solucionar, já que dependem de fatores como: a ordem de operações básicas que devem ser executadas, as características de *hardware* que o robô deve possuir para executar específica operação e a definição de qual robô é o mais indicado para a resolução de determinada operação.

Em particular, sistemas multirrobôs tratam de dificuldades intrínsecas ao grupo, que podem tornar o retorno de algumas informações imprecisas. Essas dificuldades advém de fatores que podem afetar a consistência da solução global do sistema. Alguns fatores são: dados incoerentes oriundos de ruídos na informação do sensor; resultados inesperados de ações dos componentes de *hardware*, como por exemplo, falhas; e limitações na comunicação do ambiente.

1.1 Uma breve visão do modelo desenvolvido

O objetivo principal é apresentar uma nova abordagem com estratégias para planejamento de tarefas, alocação e execução de tarefas por grupos de entidades (robôs ou agentes), independente do comportamento da arquitetura de baixo nível que dá suporte ao sistema. Assim, não são tratadas questões de baixo nível acerca dos elementos essenciais da robótica, como localização, mapeamento e dados advindos de sensores.

Esta pesquisa aborda situações de execução em tempo real, quando diante de um grupo de robôs destinados à resolução de uma determinada tarefa global, sem a presença de um robô com autoridade prioritária sobre os demais robôs do ambiente. Portanto, cada robô individual está apto a encontrar sua própria maneira de resolver o problema global, de uma perspectiva local e de forma descentralizada. Dessa forma, todos os robôs do ambiente possuem possibilidades de operações igualitárias.

A abordagem proposta envolve questões de seleção e alocação de tarefas, além de mecanismos que permitem a aquisição de conhecimento dos robôs (acerca da execução de tarefas) com os demais componentes do grupo. Experimentos para analisar a performance da proposta são conduzidos para aplicações em que uma missão necessita ser resolvida cooperativamente, mesmo quando em simulação. Os resultados das simulações demonstram que a proposta influencia positivamente a cooperação do grupo, agregando benefícios ao ambiente em que os robôs atuam.

Algumas teorias relatadas para indivíduos são analisadas neste trabalho com finalidade de mapear seus principais conceitos para os robôs. O problema global é relatado como um Problema de Execução Cooperativa de Tarefas para um grupo de robôs (*Cooperative Execution Task Problem* - CETP), em que cada robô tem autonomia para selecionar suas tarefas, alocar seus horários (filas de prioridades) e agendar as execuções. Para resolver esse problema global, é proposto um modelo de desenvolvimento intelectual para gerenciar a cooperação do grupo, sendo (o modelo) norteado por processos de aprendi-

zagem de indivíduos reais. Esse grupo apresenta a cooperação sem algum componente responsável pela coordenação.

Este trabalho levanta questões de execução em tempo real quando diante de um grupo de robôs destinados à resolução de uma determinada tarefa global, sem a presença de um robô líder (com autoridade prioritária sobre os demais robôs do ambiente). Portanto, cada robô individual pode encontrar sua própria maneira para resolver o problema global de uma perspectiva local e de forma descentralizada. Essa abordagem também envolve mecanismos que permitem a aquisição de conhecimento dos robôs (acerca da execução de tarefas) pelos demais componentes do grupo.

A figura 1.1 mostra exatamente a visão geral acerca do IDeM-MRS: um modelo formulado através de teorias advindas de pesquisadores, como Lev Semenovich Vygotsky e Jean William Fritz Piaget, de teorias originadas de abordagens de aprendizagens, como a Histórico/Social e o Behaviorismo Clássico, pregado por John Broadus Watson. Esse modelo deve ser mapeado para sistemas multirrobôs (simulando indivíduos em agentes robóticos), quando aplicados em resolução cooperativa de tarefas.

Experimentos para analisar a performance da nova abordagem são conduzidos para aplicações simuladas em que uma missão necessita ser resolvida cooperativamente entre um grupo de robôs. As análises são realizadas para validar a proposta, além de verificar a melhoria adquirida pelo ambiente. Algumas questões são particularmente pontuadas:

- Qual o percentual de tarefas que podem ser executadas pelo grupo, quando exposto ao IDeM-MRS?

- Qual o ganho efetivo de conhecimento do grupo, após a cooperação induzida pelo modelo, ao término da missão global?

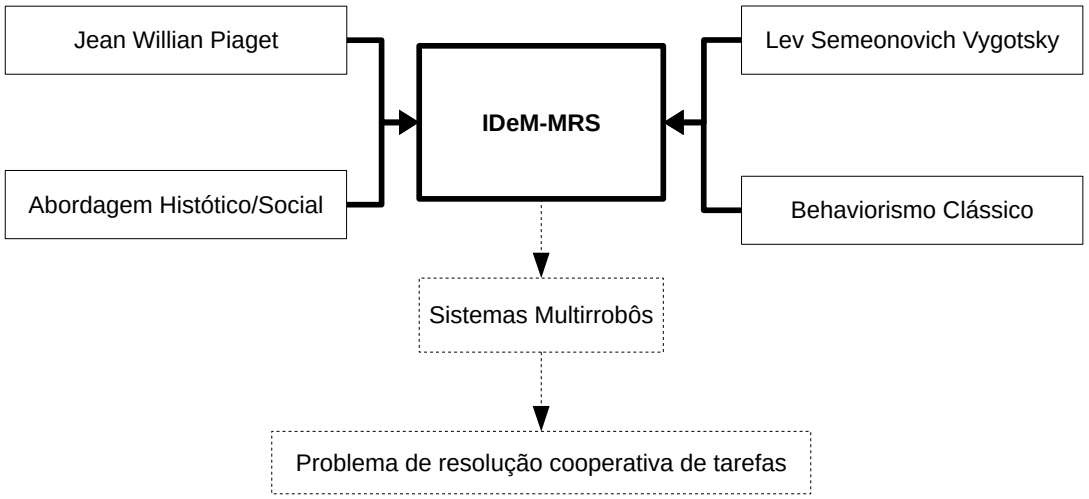

Figura 1.1: Visão geral do modelo IDeM-MRS desenvolvido.

1.2 Contribuições da Tese

A contribuição primária desta tese é a formulação matemática de teorias sociais de aprendizagem, mapeadas para agentes robóticos. A segunda, e mais importante contribuição, é a construção do modelo matemático de desenvolvimento intelectual, o IDeM-MRS, baseado no formalismo de aprendizado desenvolvido, para ser usado em um ambiente constituído por um grupo de robôs que devem cooperar em função de um objetivo global. A terceira contribuição é a consolidação das ações esperadas para os robôs, quando em resolução cooperativa de tarefas. Essas contribuições são descritas de forma sucinta a seguir.

1.2.1 Formulação matemática de teorias sociais de aprendizagem

Este trabalho parte do pressuposto de que robôs têm capacidade de aprender através da interação com outros do mesmo ambiente, comunicando-se e administrando suas diferenças de habilidades. Também pressume-se que cada robô mude suas próprias regras (alterando seu comportamento individual), sempre levando em consideração o objetivo ou trabalho comum. Sendo assim, caracteriza-se aprendizado quando um robô possui capacidade de execução da tarefa com maior eficiência do que em execuções anteriores, independente do mecanismo que o fez receber esse conhecimento acerca da tarefa. Sem esta capacidade básica, ele reagirá sempre da mesma maneira para um mesmo ambiente e em uma mesma situação, já que não ocorrerá alteração de conhecimento.

Investigando teorias de processos de aprendizagem para indivíduos reais [Vygotsky 1991, Piaget & Inhelder 1982, Watson 1913, Wenger 2000], essa tese evoluiu um estudo nas características do robô, quando em grupo, que poderiam ser mapeadas com as de um indivíduo real, quando em sociedade. Essas características permitiram analisar, previamente, a possibilidade de adequar as teorias em ambiente multirrobóticos.

- **Abordagem segundo Lev Vygotsky**

  O robô pode deter o conhecimento acerca de uma determinada tarefa, ou não. Além disso, pode ter toda capacidade de resolvê-la, autonomamente ou não. Nesse caso, é possível definir níveis de desenvolvimento dele em relação a essa específica tarefa, como os de desenvolvimento Real e Potencial, além da Zona de Desenvolvimento Proximal.

  - O Nível de Desenolvimento Real informa as tarefas que o robô é capaz de executar autonomamente.

  - O Nível de Desenvolvimento Potencial informa as tarefas que o robô pode executar (ou aprender, caso ainda não saiba).

  - A Zona de Desenvolvimento Proximal informa as tarefas que o robô é capaz de aprender.

- **Abordagem Behaviorista**

  Diante de suas regras bem definidas e implementadas, o robô pode produzir uma resposta para o grupo, ao receber um estímulo de uma tarefa específica, que necessariamente será seu conhecimento acerca da execução dessa tarefa, ou o esclarecimento da ausência de conhecimento (caso ele não saiba executá-la).

- **Abordagem Humanista**

  O robô tem total liberdade para agir, considerando suas características intrínsecas e o que está implementado em seu sistema. Isso resulta na conclusão de que as respostas às situações apresentadas ao robô são reflexos de seus *hardware* e *software*. Sendo assim, a liberdade de ação também pode ser oferecida como uma espécie de escolha de ações possíveis dentro do contexto em que o robô se encontra.

- **Abordagem Cognitiva**

  Como regra geral, o robô deve processar a informação recebida através de elementos físicos e cognitivos do ambiente em que está atuando. O robô define suas ações de acordo com o que lhe é oferecido (pelo ambiente), tais como vias de acesso, locais de obstáculos e presença de outro robô próximo.

- **Abordagem Social**

  Há uma interação dos robôs com outros do mesmo ambiente, que é chamado de contexto social. As tomadas de decisão desse robôs também leva em consideração os elementos sociais em que ele está inserido, como espaço, característica de *hardware* e *software* do ambiente.

- **Abordagem segundo Piaget**

  O robô tem capacidade de assimilar e acomodar informações em sua base de dados, sabendo que assimilar é adquirir novos conhecimento, e acomodar é atualizar conhecimento acerca de alguma tarefa.

Este trabalho construiu a base da fundamentação de que modelos de processos de aprendizagem, já consolidados para indivíduos reais, podem ser aplicados no âmbito dos sistema multirrobôs, trazendo benefícios ao grupo em termos de aquisição de conhecimento (aprendizagem). No entanto, essa aprendizagem é realizada durante a execução da missão global, oferecendo maior possibilidade de todas as tarefas serem executadas pelos robôs, que a cada interação estão com mais conhecimento em sua base de dados.

Com isso, este trabalho também concluiu que os modelos sociais de aprendizagem possuem teorias que, se aplicadas a sistemas multirrobôs, oferecem uma probabilidade maior da missão global ser concluída em sua totalidade (ou seja, todas as tarefas serem executadas com sucesso).

1.2.2 IDeM-MRS

IDeM-MRS é um modelo adequado para grupos de robôs que visam alcançar uma missão global em uma variedade de domínio de aplicação. Os membros cooperam para completar a missão através da divisão da execução das tarefas e de decisões individuais que coordenam suas ações, contribuindo para a realização do objetivo de uma maneira distribuída e colaborativa.

O comportamento dos robôs expostos ao IDeM-MRS é definido baseado em Modelos Sociais de Aprendizagem, consolidados para indivíduos reais. Essencialmente, cada robô mantém suas próprias crenças acerca dos estados correntes das tarefas e dos recursos disponíveis do ambiente. Suas ações são direcionadas por seus desejos e suas intenções para seleção e execução de uma tarefa particular da missão global (entendendo que uma missão é um conjunto de tarefas básicas). As questões seguintes indicam as propriedades que IDeM-MRS oferece:

- **Ubiquidade**

  IDeM-MRS integra seleção, alocação e execução de tarefas por robôs, em um simples modelo para resolução de tarefas de forma cooperativa.

- **Eficiência**

  IDeM-MRS oferece uma estratégia de planejamento baseada em teorias oriundas de processos de aprendizagem de indivíduos reais. A eficiência desse modelo é validado através de sua performance em diferentes simulações em ambientes experimentais com robôs heterogêneos.

- **Flexibilidade**

  IDeM-MRS é um modelo flexível que responde muito bem às mudanças que possam ocorrer no ambiente, se adaptando facilmente às novas configurações. Tarefas que possam surgir durante a execução da missão podem ser facilmente integradas na instância do problema e as mudanças são refletidas imediatamente nas regras de seleção de tarefas.

- **Aplicabilidade**

  IDeM-MRS pode ser utilizado por qualquer grupo de robôs, seja ele heterogêneo ou homogêneo. Tanto por robôs robustos, com alta performance computacional, quanto por robôs muito pequenos e com capacidade computacional limitada. Além disso, esse modelo pode ser utilizado em qualquer sistema multientidades, ou por processos paralelos em um programa.

1.2.3 Consolidação das ações esperadas para os robôs

O robô que está inserido em um ambiente multirrobô, que coopera em função de um objetivo global, deve ser capaz de planejar e executar ações. Quando se trata de um sistema para resolução de tarefas, esse robô também deve alocar horários para que sejam coordenados com os demais robôs do ambiente. Contudo, os robôs determinam suas listas de prioridades de tarefas para execuções (planejamento), constroem seus próprios horários para as resoluções (alocação) e anunciam suas intenções na execução das tarefas (execução). Todas essas ações são realizadas de forma distribuída entre os robôs do ambiente e os resultados são tratáveis computacionalmente.

A figura 1.2 informa essas ações esperadas que o robô deve ser passível de realizar, quando atuante em grupos para resolução cooperativa de tarefas. IDeM-MRS oferece regras que auxiliam aos robôs nas construções dessas ações, em que cada robô altera seu próprio estado e o estado do ambiente.

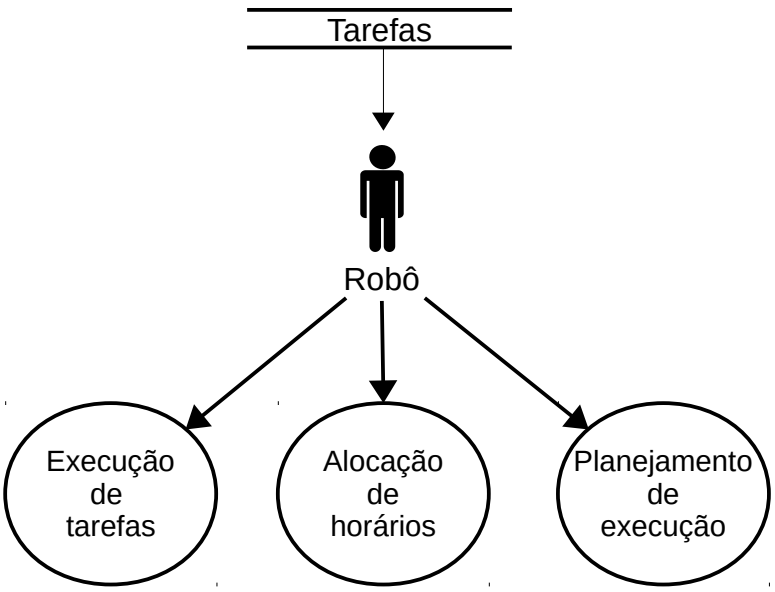

Figura 1.2: Ações esperadas pelos robôs, quando em resolução cooperativa de tarefas.

Em geral, um sistema multirrobô, quando destinado à execução cooperativa de tarefas, abrange o problema de execução cooperativa da missão global, e cada robô individualmente contribui para resolver o problema de seleção coordenada das tarefas. Esses problemas foram declarados e formalizados por Talay [Talay et al. 2007, Talay et al. 2011] como sendo, respectivamente:

- Problema de resolução cooperativa de Missão (*Cooperative Mission Achievement Problem* - CMAP);
- Problema de seleção coordenada de tarefas (*Coordinated Task Selection Problem* - CTSP).

Através do formalismo IDeM-MRS, é possível um grupo resolver cooperativamente uma missão, concomitantemente cada robô cooperativamente resolve a seleção coordenada das tarefas que compõem a missão. Talay também demonstrou que é capaz de solucionar esses problemas, porém não oferece aquisição de conhecimento aos robôs, já que a base de dados deles não sofrem atualizações. Contudo, para esta pesquisa a aquisição de conhecimento é um requisito buscado e obtido coletivamente, como forma de acúmulo de benefício ao ambiente.

IDeM-MRS garante uma forma eficiente de solucionar a seleção, alocação e execução de tarefas por grupos de entidades (robôs ou agentes) independentes, através de políticas de ações em que cada robô (ou agente) está exposto no momento. Essas políticas de ação visam resolver tanto a missão, quanto a seleção de cada tarefa específica, de uma forma eficiente.

A eficácia do modelo matemático construído foi observada analisando os resultados obtidos experimentalmente através de simulações de aplicações com domínios de execução de tarefas. Foi possível concluir que esse formalismo pode ser aplicado tanto para grupos de robôs, quanto para grupos de entidades, que estejam envolvidos cooperativamente em função de um objetivo comum.

No geral, IDeM-MRS está focado em missões complexas envolvendo tarefas com restrições de recursos e de conhecimentos próprios. Além disso, cada robô tem seus próprios recursos de *hardware* e uma base de dados composta pelos comandos básicos, que estão diretamente relacionados com as execuções das tarefas que compõem a missão do ambiente. Portanto, ambientes compostos por um time de entidades independentes (robôs) que necessitam realizar uma meta global são domínios perfeitos para aplicação do modelo de desenvolvimento intelectual.

O objetivo principal da aplicação desse formalismo vai desde a minimização do tempo gasto com a resolução completa da missão, até a atualização da base de conhecimento de cada robô do ambiente, devido à aquisição individual de conhecimento. Para isso, IDeM-MRS estabelece as regras de cooperação, que são aplicadas ao ambiente de execução em tempo real.

As regras de cooperação são baseadas em modelos sociais de aprendizagem de indivíduos reais e são mapeadas para o ambiente através de ações básicas de cada robô. São investigadas, principalmente:

- **A teoria da Zona de Desenvolvimento Proximal**
  Por Lev Semenovich Vygotsky;

- **A teoria de equilíbrio entre a acomodação e a assimilação de conhecimento**
  Por Jean William Fritz Piaget;

- **As abordagens para aprendizagem de indivíduos**
  Como a Abordagem Humanista e a Abordagem Social;

- **As orientações do Behaviorismo Clássico**
  Por John Broadus Watson.

1.3 Investigação de processos de aprendizagem social

Este trabalho realizou investigações em alguns processos de aprendizagem de indivíduos reais, o que possibilitou criar uma formulação matemática capaz de direcionar a cooperação de um grupo de robôs. Essa cooperação baseia-se na troca de experiências entre os robôs do grupo, sem a interferência de um agente externo (ou humano).

O modelo de desenvolvimento intelectual para sistemas multirrobôs desenvolvido neste trabalho, permite que haja aquisição de conhecimento de cada robô individualmente, através dos conceitos das abordagens LSM analisadas.

A figura 1.3 mostra a metodologia elaborada para construção do IDeM-MRS. Inicialmente foi feito o mapeamento das similaridades indivíduo/robô, depois a aplicação dos modelos sociais de aprendizagem e, por fim, a construção e validação do modelo. Cada etapa está descrita adiante como forma de apresentar a responsabilidade de cada uma.

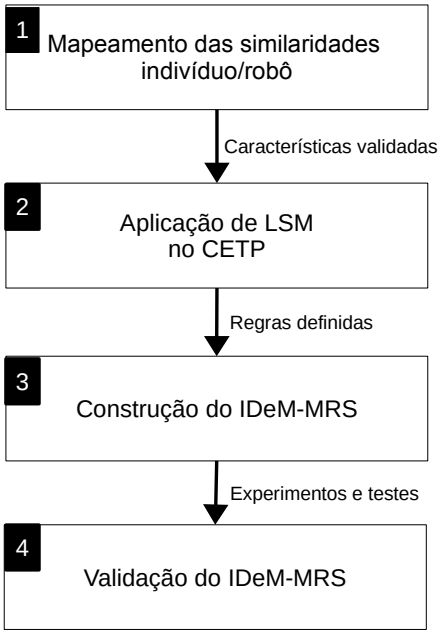

Figura 1.3: Sequência de operações para obtenção do IDeM-MRS.

1.3.1 Etapa 1: Mapeamento das Similaridades indivíduo/robô

Etapa responsável por mapear as características dos robôs que podem ser consideradas similares às identificadas nos indivíduos reais. Para isso, tanto o indivíduo quanto o robô estão expostos a contextos de grupos ou sociedade.

Com o término dessa operação, concluiu-se positivamente sobre a possibilidade de utilizar os modelos de aprendizagem social mapeados para os robôs, já que todas as características observadas nos indivíduos, quando em sociedade, podem ser mapeadas. No entanto, todas essas características foram arbitrárias, não sendo excludentes na decisão de se achar necessário a aglomeração futura de outras.

1.3.2 Etapa 2: Aplicação de LSM no CETP

Nessa etapa, foram investigados os modelos de aprendizagem Social e Humanista, além das abordagens baseadas na teoria da Zona de Desenvolvimento Proximal de Lev Semionovich Vygotsky, na teoria do Equilíbrio de Jean William Fritz Piaget e nas orientações Behavioristas Clássicas de John Broadus Watson (ver figura 1.4). Com isso, foi possível obter informações relevantes para o contexto de grupos de robôs executando, cooperativamente, tarefas. A partir desse ponto, foi projetado a formulação para um modelo matemático, com regras inferidas desses modelos.

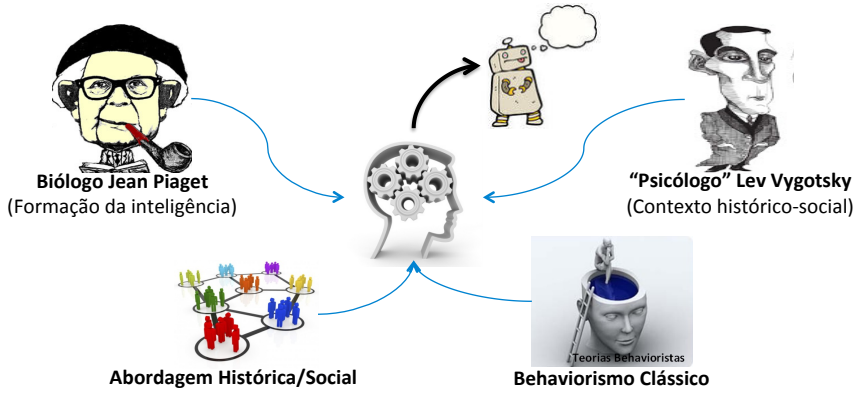

Figura 1.4: Aplicação de Modelos de Desenvolvimento Social no IDeM-MRS.

1.3.3 Etapa 3: Concepção do IDeM-MRS

Essa é a fase da construção do modelo de desenvolvimento intelectual para um grupo de robôs, quando expostos ao Problema de Execução Cooperativa de Tarefas. Esse modelo é baseado nas regras inferidas dos Modelos Sociais de Aprendizagem analisados. O capítulo 4 mostra essa fase de desenvolvimento através das seguintes fases:

- definição e formulação do problema de execução cooperativa de tarefas;
- formalização do IDeM-MRS segundo as necessidades do ambiente multirrobô;
- formalização de definições importantes para que seja possível a aplicação dos modelos sociais;
- a definição das regras de cooperação, elaboradas de acordo com a análise das teorias dos processos de aprendizagem abordados.

1.3.4 Etapa 4: Validação do IDeM-MRS

Etapa responsável pelos experimentos para verificação da eficiência do modelo. O capítulo 5 apresentam os resultados dos testes realizados em várias instâncias de grupos de robôs, diante da necessidade de resolução de diferentes tarefas.

1.4 Estrutura do documento

Este documento visa demonstrar um modelo de desenvolvimento cognitivo artificial para o aprendizado em grupo, objetivando a execução de tarefas de forma cooperativa por agentes robóticos: o modelo matemático de desenvolvimento intelectual, IDeM-MRS. Esse modelo permite o aprendizado dos indivíduos baseado em interações com o grupo, e é inspirado em teorias de aprendizagem (social) de indivíduos reais. Toda seção desse texto está organizada de forma a delinear o conhecimento do leitor acerca do assunto.

O capítulo 2 explica os conceitos que envolvem o contexto do problema de execução cooperativa de tarefas: sistemas multirrobôs, homogêneos e heterogêneos, e os mecanismos de cooperação e colaboração que esses tipos de sistemas necessitam para completar a resolução do problema global. Esse capítulo também informa a necessidade de alternativas para gerenciar a cooperação do grupo de robôs como a principal motivação para o desenvolvimento do referido trabalho. Ainda nesse capítulo são elencados os principais trabalhos relacionados com ambiente multientidades (robóticos ou agentes inteligentes) que lidam com formas de manipulação de cooperação do grupo.

O Capítulo 3 apresenta os modelos sociais de processos de aprendizagem, conforme os teóricos consolidaram (para indivíduos reais) e conforme este trabalho mapeia em robôs. As contribuições pontuais, como a Zona de Desenvolvimento Proximal gerada por Lev Semenovich Vygotsky, o equilíbrio entre a assimilação e a acomodação de conhecimento pregado pelo Jean William Fritz Piaget, a resposta ao estímulo defendida por John Broadus Watson e os conceitos definidos pelas abordagens Social e humanista, são mapeados para sistemas multirrobôs. Cada conceito que foi possível mapear do indivíduo real para o robô, está devidamente formalizado em funções, como parte das regras que regem o IDeM-MRS.

O Capítulo 4 contém o formalismo IDeM-MRS. Essa seção informa a definição formal do problema de execução cooperativa de tarefas - CETP (dentro do contexto multirrobô), contém os módulos funcionais do modelo proposto, especifica as regras que regem o IDeM-MRS, bem como os algoritmos utilizados para a implementação do modelo. Esse capítulo demonstra a possibilidade de resolver o CETP através do modelo proposto.

O Capítulo 5 mostra a investigação da performance dos módulos funcionais do IDeM-MRS para casos simulados de execução cooperativa de tarefas por um grupo de robôs. Nessa seção são mostrados os experimentos de ambientes multirrobôs simulados (em que o modelo está inserido), frente às execuções de ambientes, também simulados, mas que não possuem as regras expostas no modelo para direcionar a cooperação do grupo. Esses experimentos serviram para concretizar a eficiência do IDeM-MRS.

Finalmente, o Capítulo 6 analisa os resultados obtidos nos experimentos e conclui acerca da eficiência do modelo. Ainda nesse capítulo, são mencionados os trabalhos que possam vir a contribuir para a continuação e melhoria dessa proposta.

Capítulo 2

Sistemas Multiagentes Robóticos

Trabalho em equipe pode significar algumas palavras, como grupo de pessoas, esforço coletivo, colaboração, resolução em conjunto de uma tarefa, dentre outras similares. Juntando esses conceitos, pode-se definir que trabalho em equipe é observado quando um grupo de pessoas se dedica a realizar uma tarefa ou determinado trabalho, possibilitando a troca de conhecimento e agilidade, no cumprimento de metas e objetivos compartilhados. Exemplo de uma atuação de um trabalho em equipe são os esportes coletivos, em que times jogam uns contra outros e os jogadores do mesmo time possuem um objetivo único comum: a vitória.

O valor de um grupo de indivíduos cooperando para um mesmo ideal tem sido comprovado em vários domínios e em diferentes situações do nosso cotidiano. Dessa forma é possível perceber facilmente as vantagens de se organizar em grupos visando um objetivo em comum. É com esse intuito que se estende essa configuração a um grupo de robôs, que trabalham como um time, dividindo informações e recursos.

Em aplicações de robótica, trabalhos que envolvem robôs móveis autônomos para resolução de tarefas geralmente possuem um alto nível de flexibilidade e adaptabilidade. Como uma característica comum, esses tipos de trabalhos envolvem apenas um único robô, sendo gerenciado para tarefas previamente conhecidas e específicas, como por exemplo:

- um robô capaz de entregar bebidas em um hotel, simulando um garçom [Graf & Weckesser 1998];
- um robô que executa tarefas conforme um entregador de correspondências, em uma estação central de correios [Vestli & Tschichold-Gurman 1996];
- um robô reponsável por transportar materiais dentro de um hospital [Engelberger 1993].

Este Capítulo explora os conceitos que envolvem o contexto do problema de execução cooperativa de tarefas por agentes robóticos, bem como informa a necessidade de alternativas para gerenciar a cooperação do grupo de robôs como a principal motivação para o desenvolvimento do trabalho. Ao final, está disponível o estado da arte estritamente relacionado com esta pesquisa, sendo contextualizada a relação de cada trabalho com este tema.

2.1 Sistemas Multirrobôs

Para ilustrar a ideia prática de um sistema multirrobô operando numa experiência de trabalho em grupo na área da robótica, muito atual, pode-se usar como exemplo o futebol de robôs, em que duas equipes de robôs com rodas enfrentam uns aos outros, disputando a posse de uma bola em um campo do tamanho de uma mesa de pingue-pongue. Dependendo da categoria, as dimensões do campo e a configuração dos robôs podem ser diferentes. No caso, cada equipe se esforça para empurrar a bola para o gol do outro time localizado no extremo oposto do campo. Esse é um exemplo de sistema adverso [Cass 2001].

A ideia de cooperação entre robôs está se tornando cada vez mais popular em pesquisas no mundo. Nas últimas décadas, os sistemas multirrobôs – MRS têm recebido destaque significativo pela comunidade dos roboticistas, devido ao sucesso de suas aplicações em vários domínios. Tarefas em cenários de risco como, limpeza de resíduos tóxicos, descontaminação de ambientes nucleares, exploração planetária, combate a incêndios, procura e resgate de vítimas, segurança e vigilância, são exemplos de situações nas quais os sistemas multirrobôs podem ser aplicados. Nesses casos, os MRS apresentam maior eficiência frente à sistemas com um único robô.

Quando comparados a sistemas com um único robô, os MRS apresentam importantes vantagens quanto à distribuição de espaço e tempo, decomposição de problemas complexos, confiabilidade, robustez e custo, conforme afirma Rocha (2006). Essas vantagens são detalhadas a seguir:

- **Espaço e Tempo**

  Um time de robôs pode explorar uma determinada área mais rapidamente do que um único robô. Além disso, para o mesmo tempo, um grupo de robôs explora, em conjunto, espaços maiores de busca.

- **Decomposição de problemas complexos**

  Uma tarefa complexa pode ser decomposta em sub-tarefas menores, mais simples de ser resolvida. Nesse caso, essas sub-tarefas podem ser alocadas aos vários membros do time.

- **Confiabilidade**

  A redundância de dados sensoriais capturados pelos membros do time pode ser útil para se obter uma descrição mais precisa e confiável do ambiente dos robôs. Além disso,

- **Robustez**

  Se um membro do time falhar ou sofrer algum dano, os demais membros poderão finalizar a tarefa com autonomia e confiabilidade.

2.1.1 Sistemas homogêneos e heterogêneos

Em missões executadas por sistemas multirrobôs, a perda de um ou mais robôs é tolerável. Porém, caso essas missões fossem executadas por humanos, acidentes certamente não são elencados como aceitáveis. Isso reforça a motivação e a necessidade de usar sistemas multirrobôs em diversas situações, especialmente as que envolvem perigo aos indivíduos. Alguns exemplos de aplicações que envolvem um time de agentes ou de robôs estão listados abaixo.

- Modelagem de problemas de comércio eletrônico e criação de um framework multiagente [Chen et al. 2008];
- Envio de cartas por robôs, validando uma proposta criada para sistemas multiagentes [Carrascosa et al. 2008];
- Agendamento de rotas de transportes em tempo real [Mes et al. 2007];
- Modelagem da demanda de energia em uma região específica usando otimização por colônia de formigas [Toksari 2007];
- Missões de resgate de vítimas de desastres [Rooker & Birk 2005];
- Um sistema para programação de uma frota de robôs em ambiente fechado, cujo objetivo é executar as ordens de entregas de encomendas, realizando escalonamentos e planejamento de caminhos para os robôs envolvidos [Surmann & Morales 2002].

Diante de um grupo de robôs, quanto ao grau de similaridade entre os membros, um sistema multirrobô pode assumir classificação como homogêneo ou heterogêneo. Em um time homogêneo, os membros são idênticos em relação às partes mecânicas e dispositivos de hardware e softwares, o que simplifica a construção e programação do sistema como um todo. Esses times podem ser associados a modelos biológicos como colônias de formigas, enxames de abelhas, cardumes de peixes ou em outros grupos de insetos que possuem grande número de membros idênticos e que, em grupo, se coordenam de maneira a cooperarem até chegarem ao objetivo final [Schmickl et al. 2009].

Já os sistemas heterogêneos, pelo menos dois dos indivíduos do time diferem em capacidade de processamento, de software ou de construção mecânica. Uma configuração comum desses times é ter um de seus membros com maior poder computacional. Esse indivíduo normalmente passa a funcionar como um líder (ou coordenador) do time, podendo direcionar as ações dos demais indivíduos com menor poder computacional, além de ser utilizado em situações especiais, como por exemplo, em atividades complexas para os demais membros. A desvantagem é que se o robô coordenador falhar ou sofrer algum dano, a missão do time ficará comprometida. Porém, a heterogeneidade dos membros de um time de robôs reflete em especialidades diferentes, o que é interessante do ponto de vista de compartilhamento de recursos, conforme afirmou Murphy (2000).

2.2 Cooperação, coordenação e colaboração

Cooperação, coordenação e colaboração são três termos utilizados em sistemas multiagentes. Botelho & Alami (2000) afirma que a coordenação e colaboração são duas formas de cooperação e considera os conceitos de coordenação e colaboração descritos adiante.

- **Coordenação:**

  A coordenação acontece quando os agentes coordenam suas ações um com o outro, ou seja, sincronizam suas ações com as dos outros agentes. Dessa forma, há uma troca de informação, de sinais e de conhecimento.

- **Colaboração:**

  A colaboração, por outro lado, ocorre quando os agentes dividem a tarefa global em sub-tarefas menores e mais simples. Assim, cada sub-tarefa deve ser executada por um agente específico, em um determinado tempo.

Noreils (1993) afirma que a cooperação e a coordenação são duas instâncias distintas, e que a cooperação acontece quando dois ou mais agentes interagem com o intuito de executar uma tarefa global (ou missão). A Figura 2.1 mostra essa situação, trazendo esses conceitos para sistemas multirrobôs. Na ilustração, dois robôs (*Robô 1* e *Robô 2*) interagem em função de uma missão, ocasionando a cooperação.

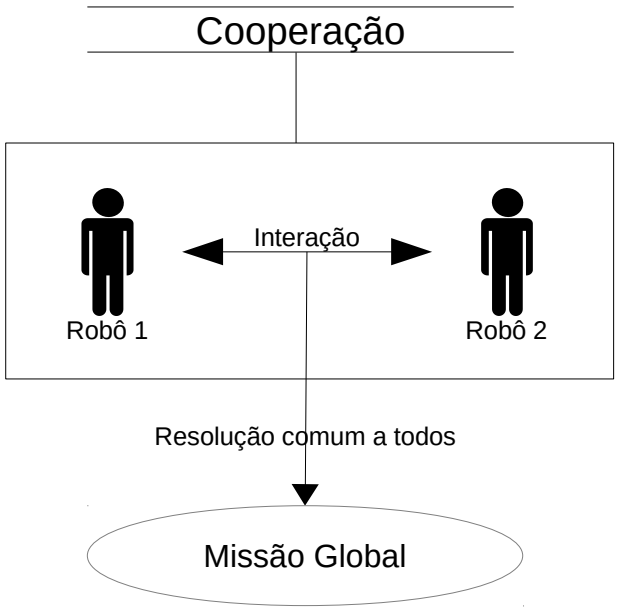

Figura 2.1: A cooperação ocorre com a interação de agentes para a execução de uma tarefa, segundo Noreils (1993).

Agregando a definição de Noreils (1993) aos conceitos de cooperação pregados por Botelho & Alami (2000), percebe-se que colaboração e cooperação podem se confundir e se tornar iguais. Porém, essa fusão não ocorre com o conceito de coordenação (ver Figura 2.2).

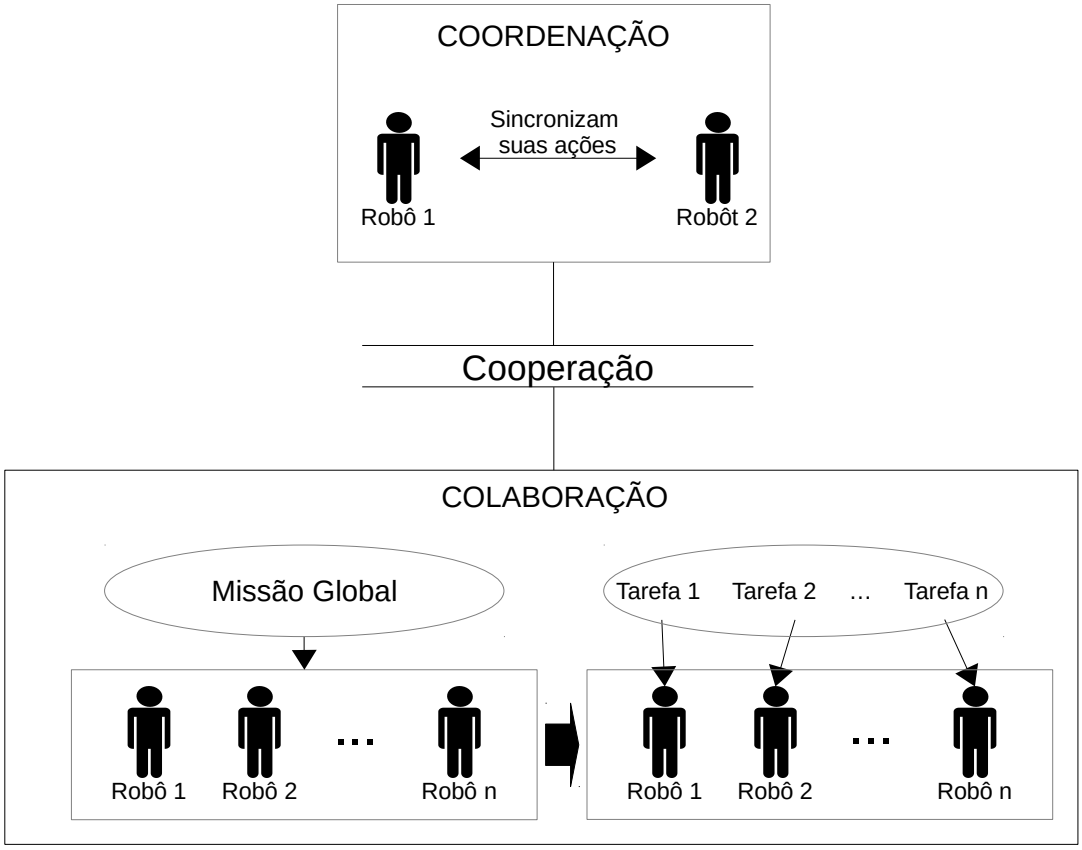

Figura 2.2: Definição de coordenação e cooperação, segundo Botelho & Alami (2000).

Para que o ambiente corresponda às expectativas de eficiência na resolução de um dado problema, os componentes do grupo devem ter ações bem definidas e pré-estabelecidas quanto ao seu comportamento cooperativo (ou colaborativo) juntamente com o mecanismo de coordenação do grupo. As interações são de natureza colaborativa, posto que todos os indivíduos colaboram com um mesmo objetivo, subdividindo as tarefas em outras independentes e menores (ou menos complexas).

O modelo formal proposto por essa pesquisa admite que a cooperação e a colaboração são conceitos muito ligados e podem ser entendidos como similares. Basicamente, ambos são formas de atuação no ambiente multiagente para que se tenha o objetivo alcançado de maneira eficiente. É importante observar que, dentro destes sistemas, também existem interações que exigem coordenação, porém a colaboração se dá em maior intensidade. No entanto, o escopo do problema alvo dessa pesquisa aborda esses conceitos em sistemas multirrobôs, em que a colaboração é atuante, desconsiderando questões inerentes à coordenação.

2.2.1 Mecanismo de Cooperação e de Coordenação

Os mecanismos de cooperação são definidos pelas regras de interação entre os agentes que devem contribuir para que suas ações individuais sejam coerentes para alcançar o objetivo final. Nesse caso, o desempenho do sistema deve ser aumentado com a existência de uma forma de cooperação eficiente.

Um mecanismo de cooperação atua sobre a tarefa, que deve ser um objetivo comum ao grupo, cujo cumprimento pode gerar benefícios a todos os componentes. A execução de uma tarefa pode requerer que ela seja dividida em outras menores e que seja utilizada uma série de requisitos básicos, como alguma forma específica de comunicação, conforme experiências no gerenciamento de organizações humanas, [Fox 1981].

Malone & Crowston (1994) definem que coordenação é um gerenciamento de dependência entre atividades, normalmente sendo realizado por um componente do grupo que tem a função de repassar as informações relevantes aos demais. Durfee (2001*b*) afirma que se trata de uma capacidade fundamental do agente para decidir acerca de suas próprias ações no contexto das atividades de outros agentes ao seu redor. Seguindo essa ideia, todos agentes do grupo teriam o poder de coordenação.

Segundo Santos & Sichman (1997), a necessidade de organização de uma comunidade de agentes inteligentes, vem da demanda por coordenação das atividades desses agentes. Essa coordenação é um pré-requisito para a obtenção de um comportamento coerente do sistema como um todo. Caso haja um mecanismo que conduza o comportamento cooperativo dos robôs, Cao et al. (1997) acredita que há necessariamente um aumento considerado na eficiência do sistema.

O aumento na eficiência da realização de tarefas é a principal razão para se desenvolver sistemas multiagentes (da mesma forma, multirrobôs). Entretanto, à medida que aumenta a quantidade de agentes atuando no ambiente, mais complexo é definir um mecanismo de cooperação e de coordenação para as ações. Por isso, em sistemas multiagentes heterogêneos, a coordenação é um tópico que deve ser cuidadosamente analisado.

No caso específico de sistemas multirrobôs, um exemplo de boa coordenação seria um ambiente em que o robô líder tivesse conhecimento de sua localização e dos demais robôs, a cada momento em que fosse requisitada essa informação. Naturalmente, uma coordenação eficiente poderia ser alcançada através de um mapa consistente que definisse a estrutura do ambiente físico, pois a disponibilidade de um mapa comum aos robôs facilitaria a coordenação, conforme afirmou Souza et al. (2010).

IDeM-MRS, no futuro, poderá dispor de mecanismos de coordenação. Uma solução para um robô conhecer a localização dos demais robôs do grupo seria utilizar mapas de ocupação, sendo obtidos através de uma rotina de mapeamento [Souza et al. 2010]. Problemas que exigem coordenação são amplamente usados para testar algoritmos de aprendizado em sistemas multiagentes. Porém nos que exigem colaboração, são aplicados algoritmos sem capacidade de aprendizado, principalmente os inspirados biologicamente, como os algoritmos de enxames inteligentes (*Swarm Intelligence*). Esses, se baseiam na ideia de que uma inteligência coletiva pode surgir das interações de um grande número de entidades simples e homogêneas, como afirmam Beni (2005) e He et al. (2006).

2.3 Trabalhos correlatos

A robótica cooperativa tem sido alvo de várias pesquisas. O uso de um grupo de robôs trabalhando juntos para a execução de diferentes tipos de tarefas, pode trazer vantagens sobre soluções com um único robô, tais como melhor performance na resolução do problema, maior tolerância a falhas do sistema geral e possibilidade de sensoriamento distribuído.

A principal característica de ambientes com vários robôs é a cooperação do grupo, o que força buscar alternativas para gerenciar essa cooperação de forma mais eficiente possível. Alguns trabalhos se destacam nesse contexto:

- Chaimowicz et al. (2001) apresentam uma arquitetura para a coordenação de grupos de robôs fortemente acoplada, adequada para a manipulação cooperatica de tarefas. Ele define um framework de cooperação multirrobô distribuído que pode ser usado para resolver problemas em uma ampla variedade de domínios de aplicação, conforme mostrado. Porém, a técnica proposta não utiliza mecanismos de aprendizagem, tornando o conhecimento dos robôs estático, sem possibilidade de aquisição no conhecimento.

- Kambayashi et al. (2009) utilizam uma forma de otimização por Colônia de Formigas em agentes móveis para simular um sistema multirrobô voltado para buscas cooperativas. No entanto, os agentes não são capazes de aprender sozinhos nem com a ajuda de outros agentes externos ao ambiente. Eles dependem do conhecimento que o usuário oferece, de acordo com a necessidade e durante a colaboração com o ambiente.

- Kalmar et al. (1998) propiciam uma aprendizagem de tarefas, partindo de um específico ambiente em que cada robô tem o mesmo conjunto de conhecimento prévio e características intrínsecas. Contudo, como esse ambiente não foi restringido a um subespaço apropriado, houve um aumento de complexidade no problema de decisão. Além disso, ele considera induzir a aprendizagem em cada robô separadamente, de forma que a meta final é determinar o quanto de conhecimento é acrescido no robô mais evoluído, não levando em consideração o ganho coletivo.

Todos esses trabalhos buscam melhorar a cooperação de sistemas multirrobôs, mas não oferecem a aprendizagem como um ganho positivo para o ambiente. O modelo de desenvolvimento intelectual proposto nesta tese (IDeM-MRS) objetiva oferecer a possibilidade de aquisição de conhecimento ao ambiente, através de regras de colaboração baseadas em modelos sociais de aprendizagem. Definindo que uma missão é um conjunto de tarefas a serem executadas, a ideia é que um sistema multirrobô, após a execução dessa missão, possua mais conhecimento do que antes da execução. Essa alteração (aquisição) de conhecimento, analisada após a execução da missão, caracteriza aprendizagem do grupo.

2.3.1 Cooperação multiagente/multirrobô

Muitos trabalhos de pesquisa atuam na cooperação entre os agentes, como:

- Schwartz & Kraus (1997) propõem um modelo de estratégia, que oferece uma negociação entre os agentes para resolver o problema de alocação de dados em um ambiente, sem nenhum coordenador central;

- Jennings (1993) utiliza uma arquitetura de alto nível para resolver problemas de execução cooperativa;

- Durfee (2001*a*) aborda o problema de planejamento, assumindo a particularidade de que os planos de um agente levam em consideração os planos de outros;

- Carver et al. (1991) implementam uma cooperação que permite obter o estado das ações que estão sendo tomadas no momento, por todos componentes do grupo.

Para tornar a cooperação mais eficientes, algumas pesquisas propõem abordagens baseadas em antecipação e raciocínio do comportamento dos outros agentes do ambiente:

- Nawarecki et al. (2003) utilizam o conceito de ontologia para facilitar a compreensão acerca da resolução das sub-tarefas pelos agentes.Com base nesse conhecimento, os agentes podem inferir a respeito da cooperação entre eles;

- Zhang & Volz (2005) observam o ambiente e as ações do grupo de agentes para estimar as crenças de cada um, no ambiente;

- Liu & Wang (2008) propõem um método de cooperação baseada na intenção de reconhecimento para prever possíveis crenças dos demais agentes do ambiente;

- Veloso et al. (1999) estabelecem regras que podem obter as próximas ações de agentes, de acordo com seus comportamentos no ambiente.

Devido à capacidade de auto-organização, descentralização e adaptação às mudanças do ambiente, algumas técnicas inspiradas em sistemas biológicos também são aplicadas em sistemas multiagentes. A exemplo disso: as técnicas de enxames inteligentes [Martinoli et al. 2004, Oliveira & Bazzan 2006], de sistemas imunológicos [Lau 2007] e de algoritmos por colônia de formigas [Bonabeau et al. 2001, Borzello & Merkle 2005].

Wenpin (2010) utiliza uma abordagem através do princípio do pensamento transposicional. Esse princípio permite que cada agente reflita acerca de possíveis ações baseadas nos comportamentos dos demais agentes do ambiente. Porém, como essa abordagem faz com que os agentes construam, gradualmente, seus planos cooperativos para execução do objetivo global, também permitem que eles obtenham comportamentos incoerentes de algum outro agente. Neste caso, o agente que errou necessita replanejar toda sua meta cooperativa, aumentando consideravelmente o custo computacional do sistema.

2.4 Contextualizando a proposta

Todos os trabalhos citados acima possuem o intuito de melhorar a cooperação do sistema multiagente. Mesmo sendo possível refletir as propriedades desses sistemas em ambientes multirrobôs, quase todos os trabalhos citados não incluem aprendizagem nos agentes (ou robôs). Os que incluem, não permitem que um robô melhore autonomamente sua habilidade na resolução de problemas.

O modelo de desenvolvimento intelectual proposto trata a cooperação de um grupo de robôs, permitindo uma efetiva aquisição de conhecimento pelos agentes. A ideia principal do IDeM-MRS está baseada na exploração da capacidade de aprendizado de um robô, dando a ele habilidades, ou melhorando as habilidades existentes, para resolução de problemas ou tarefas.

Na prática, as habilidades encontram-se no computador de bordo (hardware capaz de guardar informações). Ou seja, se um determinado robô sabe executar uma tarefa, ele possui algum código executável (uma série de passos em um programa) para isso. Este código representa a tarefa em questão e é passado ao computador de bordo de alguma maneira, seja por codificação manual de programa feita por um programador, ou usando alguma técnica de aprendizado de máquina, de forma autônoma [Song et al. 2010, Smart & Kaelbling 2002] ou por imitação [León et al. 2011, Billard & Mataric 2001, Barrios-Aranibar & Alsina 2007].

É pressuposto, neste trabalho, que este conhecimento adquirido pode ser autônomo ou passado a outros robôs, em última análise, por transferência de código executável ou de bibliotecas. Ou seja, mais geral ainda é que um robô possa aprender com um ser humano, seja por imitação ou por programação, caso os seus colegas não saibam executar a tarefa em questão. Esta é a ideia chave que permite a um grupo de robôs aprender a realizar tarefas para as quais ele possui a capacidade (recursos), mas que ainda não possui a habilidade de realizá-las. Neste trabalho, é mostrado o formalismo matemático que permite implementar esta ideia e também é verificado, através de experimentos simulados, a sua factibilidade.

Portanto, a hipótese principal desta tese é a de que um robô consegue melhorar o seu nível intelectual (conhecimento acerca da execução de tarefas), se houver interação com o grupo ou com seres humanos. Mais ainda, que um grupo de agentes consegue melhorar o seu nível intelectual, caso haja interação. Para demonstrar esta hipótese, IDeM-MRS oferece o formalismo matemático necessário, determinando um conjunto de tarefas que devem ser executadas por vários robôs, que devem cooperar para atingir um objetivo global.

Foram realizados experimentos em que o time de robôs executa as tarefas separadamente, sem interação (comunicação), e depois havendo a comunicação entre eles, regida pelas regras do IDeM-MRS. Como consequência das análises realizadas é possível mostrar que esta hipótese é válida, uma vez que ocorre melhoria no nível intelectual, concluindo que o time consegue aprender.

Capítulo 3

Modelos sociais de aprendizagem

O processo de aprendizagem pode ser compreendido como uma transformação na estrutura mental de quem aprende, ou um processo que causa uma alteração na conduta do indivíduo. Os motivos que podem fazer com que o indivíduo adquira elementos que incentivam essa transformação ou alteração, são:

- Experiências passadas com situações semelhantes;
- Observações de situações semelhantes acontecidas com outras pessoas;
- Motivação interna para agregar mais conhecimento;
- Aptidão devido à disposição genética.

Compreende-se que o ser humano nasce potencialmente inclinado a aprender, mas necessita de estímulos externos (do ambiente em que vive) e internos (como uma espécie de motivação). Ele já nasce com uma vontade de aprender e com capacidades criativas e dinâmicas para tal processo.

A vontade de aprender é uma característica dinâmica e essencial do psíquico humano, pois está sempre em mutação e procurando novas informações. Além disso, essa vontade também é criativa, pois busca novos métodos para melhorar o entendimento acerca de algo, como por exemplo, a metodologia de aquisição de conhecimento por tentativa e erro.

Segundo Wertsch (1997), o indivíduo possui o que se chama de *conhecimento nato*, que são, dentre outros: andar, falar, correr e brincar. Eles são temporais e são introduzidos no meio social em que vivem. Porém, esses conhecimentos necessitam de maturidade física, psicológica e social, mesmo sendo oferecidos de forma certa e sem dispender grandes esforços.

A partir das necessidades do indivíduo, novos conhecimentos deverão surgir (os não natos), configurando uma situação de aprendizagem. Muitos estudiosos definem teorias sobre como essa aprendizagem se dá ao longo do tempo. Algumas delas estão elencadas sucintamente nas seções seguintes.

3.1 Questões fundamentais sobre a aprendizagem social

O processo de aprender de um indivíduo é bastante complexo e envolve vários fatores, como variáveis cognitivas, afetivas, sociais, econômicas e até políticas. Sabendo disso, essa pesquisa investigou os modelos sociais de aprendizagem (*Learning Social Models* - LSM) que tratam as abordagens social e humanista, além das idéias e teorias pregadas pelos pesquisadores Jean William Piaget, Lev Semionovich Vygotsky e John Broadus Watson.

Os robôs têm capacidade de adquirir conhecimento através da interação com outros do mesmo ambiente, se comunicando e administrando suas diferenças de habilidades. Também é possível que cada robô mude suas próprias alternativas, alterando com isso, seu comportamento individual. Em uma situação própria, é possível também que um robô deixe de atuar no ambiente ou seja substituído por outro, como uma espécie de morte e nascimento de um indivíduo.

As relações entre os robôs são mediadas por um objetivo ou trabalho comum, que é a realização de uma determinada tarefa, conhecida previamente ou não. Contudo, a aprendizagem se dá quando um robô possui mais conhecimento acerca da execução de uma tarefa do que em execuções anteriores (passadas). Sem esta capacidade básica, ele reagirá sempre da mesma maneira, para um mesmo ambiente e diante de uma mesma situação.

Inicialmente, algumas informações são fundamentais para compreender como se dá o processo de aprendizagem nos indivíduos, são elas: conhecimentos natos e anteriores, cooperação entre mais de um indivíduo, memória dispendida, motivação necessária e quantidade de informação passada.

- **Conhecimento nato:**

  São conhecimentos que o indivíduo poderá adquirir, de acordo com o seu tempo de maturidade física, psicológica e social, sem qualquer estímulo obrigatório. São capacidades inerentes à condição do ser humano. Dentre outros, são exemplos de conhecimentos natos:

  - andar e correr;
  - falar e sorrir;
  - sentar e levantar.

- **Conhecimento anterior:**

  A aprendizagem é caracterizada pela incorporação de novos conhecimentos a alguma informação armazenada, de modo que, após essa incorporação, o indivíduo passará a ter mais conhecimento do que antes. Sendo assim, o indivíduo pode possuir algum conhecimento, chamado de conhecimento anterior, que já estava previamente guardado em sua memória, seja por alguma experiência vivida ou por algum ensinamento.

- **Memória dispendida:**

  O indivíduo possui lembranças que são permanentes, como imagem, odores e sons. Elas podem ser entendidas como blocos desconexos que, quando ativados, montam a lembrança que é novamente sentida pelo indivíduo, formando o denominado conhecimento anterior. Esse conhecimento, por sua vez, está situado na memória de longo prazo. Pode-se perceber, portanto, que o conhecimento nato está situado nessa memória. Porém, existe também a memória de curto prazo, que é esquecida completamente com o decorrer de um determinado tempo. Sendo assim, a memória pode ser:

  - memória de curto prazo, que é liberada com o decorrer de um determinado tempo pequeno;
  - memória de longo prazo, que possui um tempo grande para sua total liberação de espaço.

- **Motivação necessária:**

  A aprendizagem é obtida mais rapidamente e mais prazerosamente quando há algum interesse por parte de quem deseja aprender. Nesse caso, a motivação é algo como um impulso interno que traz benefícios a quem quer aprender como uma espécie de gratificação individual.

- **Quantidade de informação:**

  Mesmo o indivíduo possuindo a capacidade de aprender muito grande (podendo ser ilimitada), não pode integrar grande quantidade de informação ao mesmo tempo. Dessa forma, a quantidade de informação oferecida ao indivíduo em um dado momento deve ter um limite ideal (dependendo de cada um), para que não comprometa a aprendizagem. Esse comprometimento pode ocorrer devido a dois fatores relacionados às informações recebidas:

  - desmotivação pessoal diante da complexidade dessas informações;
  - confusão no momento de gerenciar a ordem de aquisição dessas informações.

- **Cooperação de indivíduos:**

  Se um dado problema for oferecido a um grupo de pessoas que se ajudam, elas terão formas diferentes de resolução, já que cada indivíduo responde a uma situação específica de forma diferente. Alguns tipos de problemas apresentam melhores resultados se solucionados de forma cooperativa, e portanto, a aprendizagem é mais eficaz. Notoriamente, a aprendizagem cooperativa, quando observada em interações ou em ajuda mútua de grupo, permite a resolução de problemas mais complexos de forma mais simples e eficaz.

3.2 Similaridades entre robôs e indivíduos em processo de aprendizagem

Para identificar a possibilidade de inserir processos de aprendizagem em um ambiente multirrobô (seguindo as teorias consolidadas para indivíduos reais), foi necessário investigar as características do robô que poderiam ser diagnosticadas como similares as dos indivíduos. A definição dessas características cruciais (dos indivíduos nos robôs), oferecem a possibilidade de aplicar os modelos de processos sociais, com algumas alterações.

A análise comparativa é feita em robôs, quando inseridos em um mesmo ambiente, e em indivíduos reais, quando participante de atividades sociais em um mesmo contexto. Nada mais lógico do que analisar as questões fundamentais e questionar a possibilidade de mapeá-las nos robôs. A tabela 3.1 mostra a comparação realizada entre o indivíduo e o robô, levando em consideração os itens: sociedade, conhecimento nato, memória dispendida, formas de motivação e aprendizagem por estímulos externos. Mais adiante, cada seção explica o mapeamento obtido em cada item.

Características	**Indivíduo**	**Robô**
Sociedade	Ser humano	Entidade Computacional
	Possui diferentes características (físicas/mentais)	Possui diferentes características (*hardware*/*software*)
	Podem conviver em grupos (sociedade)	Pode conviver em grupos (sistemas multirrobôs)
Conhecimento	Pode ser compreendido como conhecimento necessário	Pode ser compreendido como condição nata
	Nasce com conhecimentos considerados natos, mas precisa de maturidade para desenvolvê-lo	Pode ser projetado com conhecimentos básicos, dependendo da aplicação
	São conhecimentos natos: andar, correr, sorrir, falar, dentre outros	São conhecimentos básicos: movimentação e comunicação
Memória	Possui memória de curto e longo prazos	Pode assumir base de dados fixas ou voláteis
	A memória de curto prazo pode ser esquecida mais rapidamente	A base de dados, dependendo da necessidade, pode ser apagada
	O esquecimento de ambas as memórias é uma ação involuntária	A alteração da base de dados pode ocorrer em qualquer tempo
Motivação	Motivação interna ou pessoal	Recompensa
	Motivação externa	Bonificação
Estímulos	Oriundos de outro indivíduo mais experiente	Oriundos de outro robô com mais conhecimento

Tabela 3.1: Características do indivíduo mapeadas para robôs móveis no contexto de aprendizagem.

3.2.1 Mapeamento da sociedade

O indivíduo possui características físicas e capacidades mentais diferentes entre si. O robô pode ser considerado como uma entidade computacional com capacidade de *software* e característica de *hardware* distintas individualmente. Traçando o mapeamento, a capacidade de *software* do robô se equivale à capacidade mental do indivíduo. Já a característica de *hardware* do robô, se equivale à característica física do indivíduo.

O indivíduo nasce com pré-disposição para conviver com outros em sociedade. Em contra-partida, o robô pode ser projetado para ter sua vida útil em um ambiente com outros robôs, reagindo a ele (ambiente) com certa autonomia. Nesses termos, o tempo de vida que o indivíduo pode atuar na sociedade, pode ser mapeado para o tempo útil que o robô dispende no ambiente. Logo, ambos podem conviver em grupos.

- **Indivíduos:** individualmente são pessoas com diferentes características físicas e mentais, que podem conviver em grupos. Esses grupos são denominados de sociedade.

- **Robôs:** individualmente são entidades computacionais com *software* e *hardware*. Tanto a capacidade de *software*, quanto as característica de *hardware* podem ser distintas (ou não) entre si. São capazes de conviver em grupos, que são denominados de sistemas multirrobôs.

3.2.2 Mapeamento do conhecimento nato

O indivíduo nasce com expectativas de adquirir todo conhecimento nato possível, como andar, falar, correr, sorrir, dentre muito outros. No entanto, esses conhecimentos vão sendo inseridos com o passar do tempo e com o decorrer da maturidade individual.

O robô pode ser projetado com conhecimento básicos para sua autonomia. Ele pode se movimentar tanto de um lugar para outro, quanto no mesmo lugar. Além disso, ele também pode ser projetado para se comunicar com outros do mesmo ambiente. Essa comunicação deve ser conhecida por todos do grupo, para haver compreensão. O que ocorre também com o indivíduo, que para poder compreender o que outro fala, precisa conhecer a mesma língua de comunicação, em se tratando da comunicação oral.

Portanto, os conhecimentos natos que o robô necessita ter, sem qualquer estímulo externo, são: movimentação (uma vez que são móveis) e comunicação (uma vez que precisam interagir com outros robôs no ambiente). No entanto, eles não passam por um período de maturidade, pois já são construídos com essas características.

- **Indivíduos:** nascem com conhecimentos natos, mas precisa de maturidade para desenvolvê-lo, como: andar, correr, sorrir, falar, dentre outros.

- **Robôs:** podem ser projetado com conhecimentos básicos, como: movimentação e comunicação.

3.2.3 Mapeamento da memória dispendida

O indivíduo possui memória de longo e de curto prazo. A primeira passa mais tempo para desaparecer. A segunda pode ser esquecida em tempo substancialmente rápido, ou até mesmo, devido a alguma circunstância específica, como por exemplo, trauma, doença, ou qualquer outro fator relevante e atípico. Porém, ambas possuem tempos de esquecimento involuntários. Não é possível ao indivíduo, manipular esse tempo.

O robô pode ser projetado com memória, porém, diferindo da memória do indivíduo, ela pode ser manipulada. Ou seja, essa memória pode ser alterada ou apagada em qualquer tempo, dependendo do projeto inicial do robô, voltado para o fim específico.

- **Indivíduos:** possuem memória cujo esquecimento é uma ação involuntária.
- **Robôs:** possuem base de dados, capazes de sofrerem alterações em qualquer tempo.

3.2.4 Mapeamento das formas de motivação

Diante de qualquer situação, o indivíduo pode receber alguma motivação como estímulo para desenvolver algo. No entanto, ele pode ter sua própria motivação (motivação interna) ou receber de algo externo (motivação externa).

Para o robô, pode ser inserido algum tipo de motivação, como recompensa ou bonificação para o desenvolvimento de algo. Especificamente no contexto de execução de tarefas, o robô pode receber alguma recompensa por cada tarefa executada. Dessa forma, essa bonificação funciona como um mecanismo para incrementar a busca desse robô por mais execução de tarefas. Essa alternativa de inserção de motivação no robô, pode auxiliar ao grupo na execução da tarefa global, a missão, estabelecendo uma espécie de competição entre os membros do grupo para captação de mais recompensas. Mesmo com essa configuração de competição, não pode haver comprometimento da execução da tarefa global.

- **Indivíduos:** possuem motivação interna (ou pessoal) e motivação externa.
- **Robôs:** possuem a recompensa e a bonificação como principais formas de simular a motivação.

3.2.5 Mapeamento da aprendizagem por estímulos externos

O indivíduo pode aprender algo novo, caso um outro indivíduo lhe ensine. Dessa forma, o estímulo externo se dá através de outro indivíduo mais experiente.No caso do robô, ele também é capaz de atualizar ou aumentar sua base de dados com informações passadas por outro robô do ambiente, sendo esse mais experiente do que aquele no referido assunto.

- **Indivíduos:** recebem estímulos através de outro indivíduo mais experiente.
- **Robôs:** recebem estímulos através de outro robô com mais conhecimento.

3.3 Teorias de Vygotsky aplicadas em robôs

Lev Semenovich Vygotsky foi um psicólogo bielo-russo e pesquisador do desenvolvimento intelectual das crianças. Ele afirmou que esse desenvolvimento ocorre em função das interações sociais e condições de vida que a criança está inserida. Ele defendeu em [Vygotsky 1993] que o processo de aprendizagem se dá de maneira histórico-cultural, a partir das interações com outros indivíduos. Toda sua investigação levou à elaboração de um modelo em psicologia, alicerçado na teoria marxista do funcionamento intelectual humano.

De acordo com o teórico, o desenvolvimento cognitivo ocorre pelo processo de internalização das interações sociais. Oliveira (1997) afirmou que a interação social de membros de um ambiente culturalmente estruturado, fornece a matéria prima para o desenvolvimento intelectual do indivíduo. Vygotsky afirmava que a criança nasce apenas dotada de funções cognitivas elementares, como os reflexos e a atenção involuntária. Então, com o aprendizado cultural mediado pela linguagem, parte dessas funções básicas transforma-se em funções cognitivas superiores, que abrangem conhecimentos mais complexos.

Segundo Frawley (2000), para Vygotsky, o desenvolvimento é teleológico. A aprendizagem se dá quando, ao ocorrer uma situação inesperada, o indivíduo solicita apoio ao grupo, que responde com informações de acordo com as condições sociais. Esse processo está envolvido com três conceitos elementares: a internalização, a mediação e o controle. Através da mediação e da internalização, o pensamento superior do indivíduo busca o controle. A saber:

- **Internalização**: refere-se ao processo através do qual sugestões ou conteúdos externos ao indivíduo (apresentados por um outro) são trazidos para o domínio intrapsicológico (do pensar e do sentir subjetivos), passando a incorporar-se à subjetividade do indivíduo.

- **Mediação**: é o processo de intervenção de um elemento intermediário em uma relação, que deixa de ser direta e passa a ser mediada por esse elemento.

- **Controle**: possui três características importantes que possibilitam ao indivíduo adquirir de novos conhecimentos. São eles: o planejamento, a inibição e o local de controle.

  - **Planejamento**: permite que o indivíduo antecipe e regule as ações, no presente e no futuro.

  - **Inibição**: constitui um filtro cognitivo que limita, para o indivíduo, as opções feitas durante o planejamento.

  - **Local de controle**: especifica onde o indivíduo obtém as informações para regular o pensamento.

Vygotsky (1991) identificou dois níveis de desenvolvimento cognitivo: o Real e o Potencial. O nível Real (*Level of Real Development* - LRD), também chamado de adquirido ou formado, é o desenvolvimento obtido através do ambiente ou de outro indivíduo, ou ainda, formado através de ações próprias. Ele determina o que uma pessoa é capaz de fazer sozinha, sem a ajuda de outras pessoas.

O nível Potencial (*Level of Potential Development* - LPD) define a capacidade que o indivíduo tem de aprender e fazer, embora com a ajuda de outra pessoa mais experiente. Esses conceitos originaram o que Vygotsky definiu como sendo a Zona de Desenvolvimento Proximal (*Zone of Proximal Development* - ZPD), que é a diferença entre os níveis de desenvolvimento Real e Potencial do indivíduo (Figura 3.1).

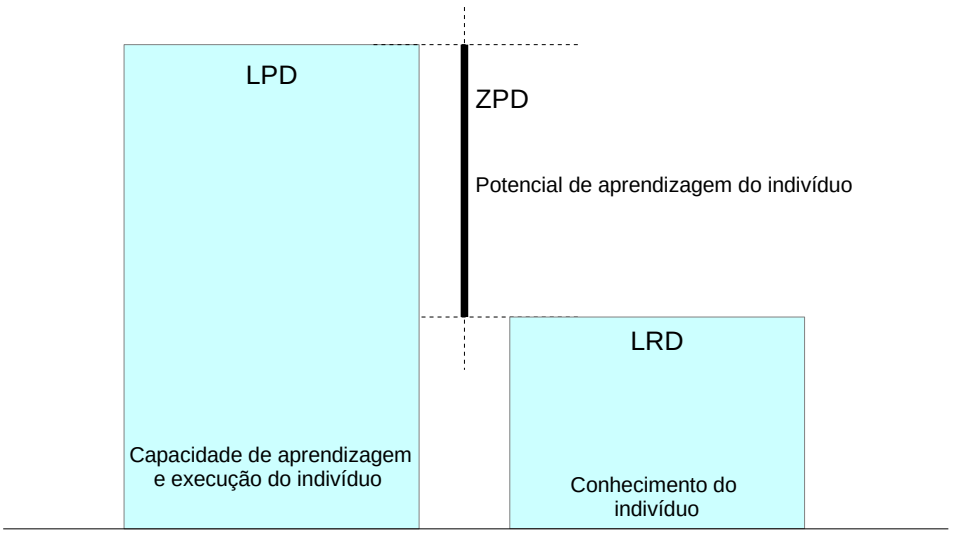

Figura 3.1: A ZPD do indivíduo, segundo a teoria de Vygotsky.

Outra teoria de Vygotsky é a afirmação de que a aprendizagem evolui com o desenvolvimento, o que produz abertura na ZPD de cada indivíduo [Vygotsky 1978]. Através disso, constata-se que o processo de aprendizagem está relacionado com o desenvolvimento individual. Mais ainda, que a evolução do coletivo é resultado da evolução individual, quando em grupo.

Contudo, Vygotsky também mostrou interesse nas áreas de psicologia do desenvolvimento, desenvolvimento infantil e educação [Vygotsky 1967]. Porém, formalizando a teoria sócio-interativista desse pesquisador, este trabalho elaborou a Equação (3.1), que define a ZPD de um indivíduo qualquer *i* como a diferença entre seus níveis de desenvolvimento Potencial (LPD_i) e Real (LRD_i).

$$ZPD_i = LPD_i - LRD_i \tag{3.1}$$

3.3.1 Nível de Desenvolvimento Real

O Nível de Desenvolvimento Real é o conhecimento de determinadas situações já consolidado pelo indivíduo, de forma a torná-lo capaz de resolver essas situações autonomamente. Em outras palavras, informa a capacidade de uma pessoa de resolver um problema sem ajuda. Esse nível de desenvolvimento é dinâmico e aumenta com o decorrer do processo de aprendizagem.

Para o robô, o LRD informa a capacidade que o robô tem de resolver problemas utilizando apenas seu conhecimento autonomamente. A Figura 3.2 informa graficamente esse conceito e a Equação (3.2) mostra como esse conceito pode ser obtido, sabendo que α_i é o agente robótico, τ_j é a tarefa a ser executada e $y \in \mathbb{N}$, $y = \{0,1\}$.

$$f\_lrd(\alpha_i, \tau_j) = y \tag{3.2}$$

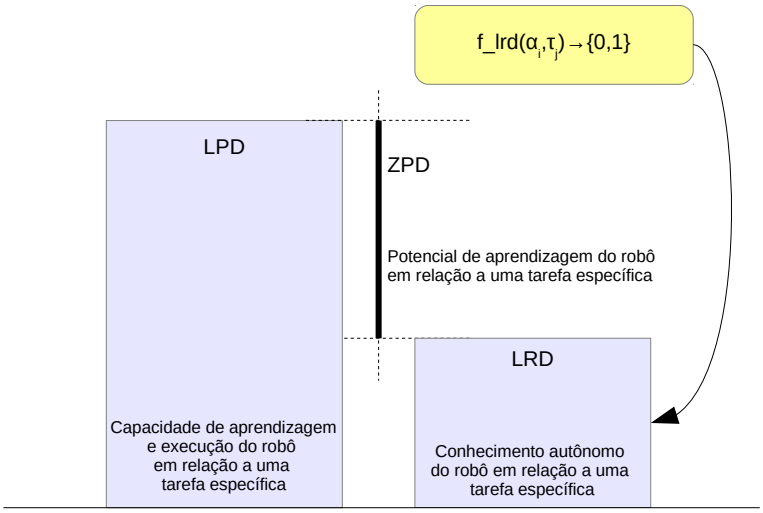

Figura 3.2: O Nível de Desenvolvimento Real, de Vygotsky, mapeado para robôs.

Nesse contexto, algumas alternativas devem ser analisadas:

- **Questão 1. O robô não é capaz de resolver a tarefa sozinho.**
  Se $f\_lrd(\alpha_i, \tau_j) = 0$, então o robô α_i não é capaz de resolver a tarefa τ_j sozinho, seja por um dos motivos:

  (i) não possuir todo conhecimento requisitado pela tarefa;
  (ii) não possuir toda capacidade física necessária.

- **Questão 2. O robô é capaz de resolver a tarefa sozinho.**
  Se $f\_lrd(\alpha_i, \tau_j) = 1$, então o robô α_i é capaz de resolver τ_j sozinho. Nesse caso, o robô α_i possui conhecimento e característica física necessários para a resolução da tarefa τ_j.

3.3.2 Nível de Desenvolvimento Potencial

O Nível de Desenvolvimento Potencial é determinado pelas habilidades que o indivíduo poderá construir, podendo ser inferido com base no que o indivíduo consegue resolver, com ajuda ou não.

Para o robô, o LPD define a capacidade de aprendizagem e execução que o robô poderá desenvolver quando necessário. Esse nível é calculado através da Equação (3.3), sabendo que α_i é o agente robótico, τ_j é a tarefa a ser executada e $w \in \mathbb{N}$, $w = \{-1, 1\}$. A Figura 3.3 expõe esse conceito.

$$f\_lpd(\alpha_i, \tau_j) = -w \tag{3.3}$$

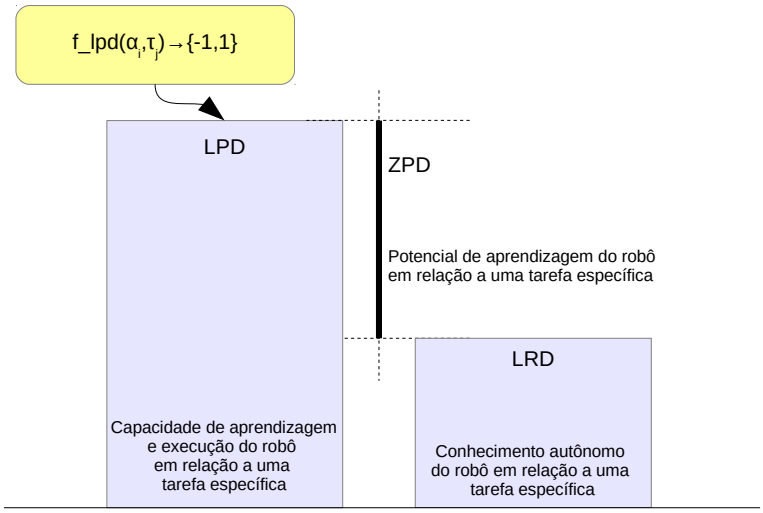

Figura 3.3: O Nível de Desenvolvimento Potencial, de Vygotsky, mapeado para robôs móveis.

Através das ideias que envolvem o Nível de Desenvolvimento Potencial, é possível inferir acerca das seguintes questões:

- **Questão 1. O robô é capaz de aprender a executar a tarefa.**
  Se $f\_lpd(\alpha_i, \tau_j) = -1$, então $w = 1$. Isto quer dizer que o robô α_i tem capacidade para aprender e executar a tarefa τ_j. Conclui-se, portanto, que o robô α_i tem a característica física necessária para executar τ_j.

- **Questão 2. O robô não é capaz de aprender a executar a tarefa.**
  Se $f\_lpd(\alpha_i, \tau_j) = 1$, então $w = -1$. Sendo assim, o robô α_i não tem capacidade para aprender a tarefa τ_j. Isto deve-se ao fato de ocorrer pelo menos uma das seguintes alternativas:

  (i) α_i não possuir a capacidade física necessária para executar τ_j;
  (ii) no ambiente, não há um grupo que, cooperativamente, detenha todo conhecimento necessário para repassá-lo a α_i.

3.3.3 Zona de Desenvolvimento Proximal

A Zona de Desenvolvimento Proximal fornece os indícios do potencial da aprendizagem, isto é, as informações que o indivíduo tem potencialidade de aprender, desde que ele ainda não tenha finalizado seu processo de aprendizagem.

Por outro lado, a ZPD de cada robô informa sua capacidade de aprender uma determinada tarefa, desconhecida por ele até o momento. A Equação (3.4) define a ZPD de um robô α_i em relação a uma tarefa específica τ_j. A Figura 3.4 mostra o conceito da Zona de Desenvolvimento Proximal mapeada para os robôs.

$$f\_zpd(\alpha_i, \tau_j) = f\_lpd(\alpha_i, \tau_j) - f\_lrd(\alpha_i, \tau_j) \tag{3.4}$$

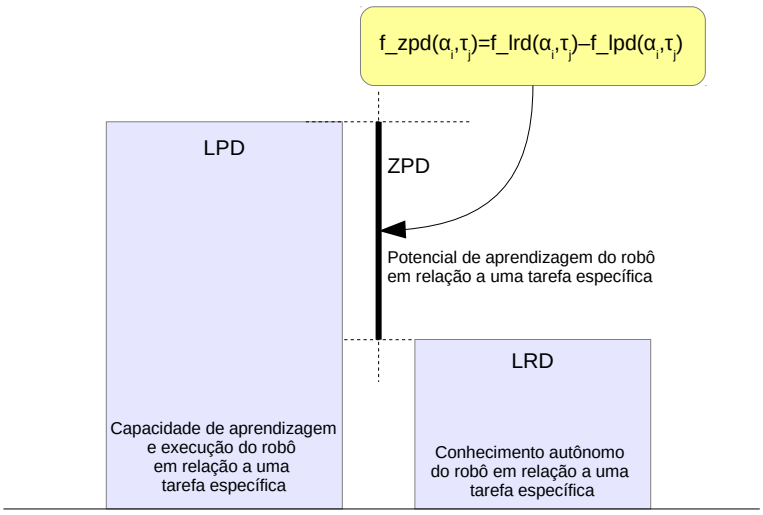

Figura 3.4: A Zona de Desenvolvimento Proximal, de Vygotsky, mapeada para robôs.

Facilitando a inferência dos resultados a serem obtidos para possibilitar a identificação das características reais e potenciais, a análise dessa equação permite as seguintes conclusões para as questões abaixo:

- **Questão 1. O robô não sabe resolver a tarefa, mas não é capaz de aprendê-la.**
  Se $f\_zpd(\alpha_i, \tau_j) = -1$, então $f\_lpd(\alpha_i, \tau_j) = -1$ e $f\_lrd(\alpha_i, \tau_j) = 0$, ou seja, α_i não sabe resolver τ_i mas não é capaz de aprendê-la.

- **Questão 2. O robô sabe resolver a tarefa, mas não é capaz de aprimorar seu conhecimento.**
  Se $f\_zpd(\alpha_i, \tau_j) = 0$, então $f\_lrd(\alpha_i, \tau_j) = 1$ e $f\_lpd(\alpha_i, \tau_j) = 1$, ou seja, o robô α_i já sabe resolver a tarefa τ_i, e não é capaz de aprimorar o seu conhecimento a cerca da resolução. Esse é um exemplo de robôs que estão programados com uma execução de tarefa e que não podem ser reprogramados, seja por limitações nas especificações de *hardware* ou de *software* (programador ou recursos insuficientes).

- **Questão 3. O robô não sabe resolver a tarefa e não é capaz de aprendê-la.**
  Se $f\_zpd(\alpha_i, \tau_j) = 1$, então $f\_lrd(\alpha_i, \tau_j) = 0$ e $f\_lpd(\alpha_i, \tau_j) = 1$, ou seja, α_i não sabe resolver τ_i, e não é capaz de aprendê-la.

- **Questão 4. O robô sabe resolver a tarefa e ainda é capaz de aprimorar seu conhecimento.**
  Se $f\_zpd(\alpha_i, \tau_j) = -2$, então $f\_lrd(\alpha_i, \tau_j) = 1$ e $f\_lpd(\alpha_i, \tau_j) = -1$, ou seja, o robô α_i já sabe resolver τ_i e ainda é capaz de aprimorar seu conhecimento acerca da resolução.

Finalmente, conforme conceitos analisados de Vygotsky para indivíduos reais, a saber: níveis de desenvolvimento Potencial e Real e a Zona de Desenvolvimento Proximal (rever Figura 3.1), as equações (3.2), (3.3) e (3.4) estão mostradas em conjunto na Figura 3.5.

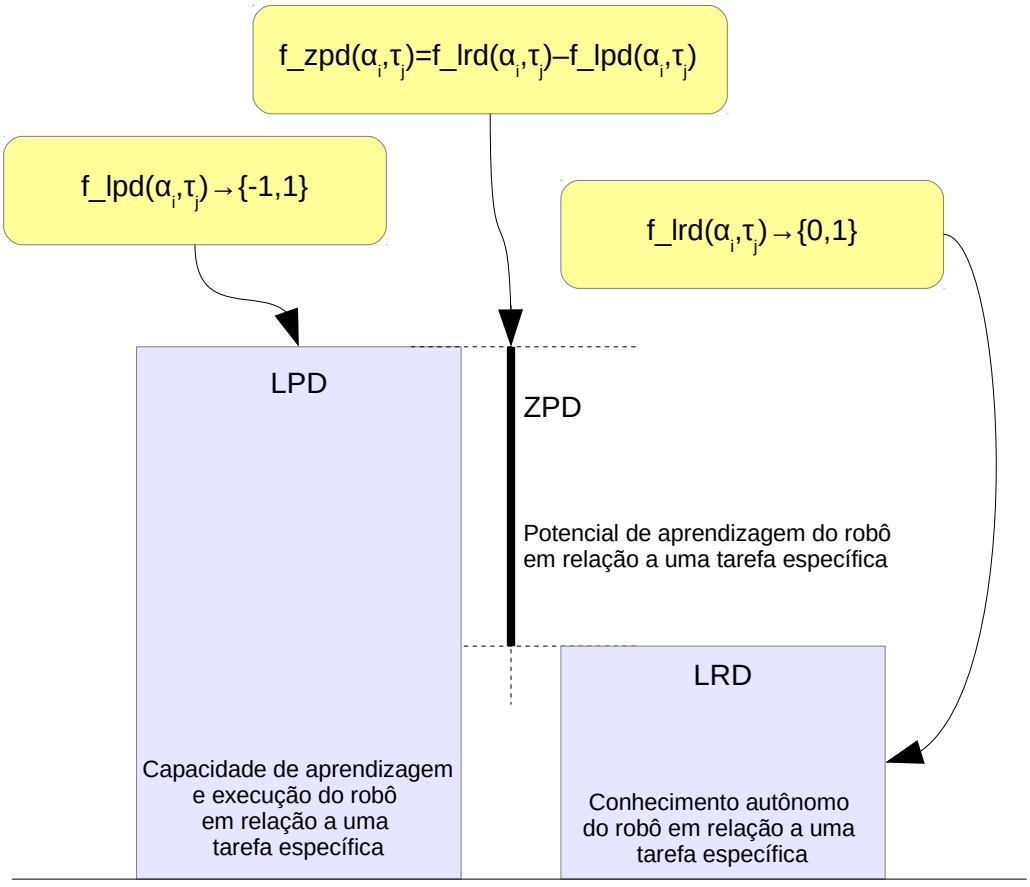

Figura 3.5: Aplicação da Teoria da Zona de Desenvolvimento Proximal, de Vygotsky, para os robôs.

3.4 Teorias de Jean Piaget aplicadas em robôs

Jean William Fritz Piaget é considerado um grande expoente do estudo do desenvolvimento cognitivo. Ele partiu do postulado que o desenvolvimento da personalidade humana, sob os aspectos intelectuais, é uma consequência das relações afetivas, sociais e morais constituintes da vida educativa institucional. Piaget & Inhelder (1982) afirmaram que esses elementos eram indissociáveis para a aprendizagem eficaz de um indivíduo, enquanto estudante de uma instituição.

A base dos trabalhos desse pesquisador suíço é a Teoria do Equilíbrio. Ela é um mecanismo auto-regulador necessário para assegurar que uma criança tenha uma interação eficiente com o meio ambiente em que vive, indicando que ela deve ter um ponto de equilíbrio entre a Assimilação (integração) e a Acomodação (diferenciação) do conhecimento adquirido.

Segundo Wadsworth (1996), a Assimilação é a incorporação de elementos externos e compatíveis com a natureza do indivíduo em sua base de conhecimento. Já Acomodação é a modificação de algum esquema armazenado. Se um indivíduo, por exemplo, não assimilasse um estímulo, ele tentaria fazer uma Acomodação dessa informação (conhecimento). Caso contrário, esse indivíduo criaria um esquema novo em sua base de conhecimento. Esse processo é definido como Assimilação e, nesse momento, o equilíbrio entre a Assimilação e a Acomodação é alcançado.

Em um caso extremo, se o indivíduo somente assimilasse (e não acomodasse) o estímulo, segundo Piaget (1975), seriam desenvolvidos poucos esquemas cognitivos. Como esses esquemas seriam muito amplos, poderia comprometer a capacidade de diferenciação. Porém, se fosse o contrário e o indivíduo não assimilasse qualquer estímulo (somente acomodasse), os esquemas cognitivos desenvolvidos seriam bem pequenos e em grande número. Isso comprometeria este esquema de generalização de tal forma que a maioria das coisas seriam vistas sempre como diferentes, mesmo sendo muito similares. Portanto, é possível identificar algumas conclusões:

- Assimilação sem Acomodação: indivíduo sem capacidade de diferenciação;
- Acomodação sem Assimilação: indivíduo sem capacidade de generalização.

Contudo, Piaget & Inhelder (1982) explicaram como se desenvolve a inteligência nos seres humanos com comprovação em bases científicas. Segundo eles, os fatores que influenciam no desenvolvimento cognitivo de indivíduos reais são processos resultantes de características comportamentais, culturais, sociais e biológicas, às quais o indivíduo está inserido.

O psicólogo, portanto, pregou a necessidade da aplicação da Teoria do Equilíbrio, impondo que o indivíduo seja capaz de, cognitivamente, assimilar experiências e/ou acomodá-las, ficando assim em harmonia com seus conhecimentos. Nesses termos, entende-se que válidos os seguintes conceitos:

- **Assimilação:** é a incorporação de um conhecimento novo;
- **Acomodação:** é a atualização de um conhecimento prévio.

3.4.1 Equilíbrio entre a Assimilação e a Acomodação

O processo de desenvolvimento cognitivo do indivíduo pregado por Piaget possui, como fontes de influência, os fatores comportamentais, culturais, sociais e biológicos. Além disso, Piaget acreditava que necessariamente deveria haver um equilíbrio entre a Assimilação de um novo conhecimento e a Acomodação do conhecimento prévio.

Fazendo o mapeamento das ideias de Piaget para um grupo de robôs aplicados ao Problema de execução cooperativa de tarefas, o desenvolvimento do conhecimento de cada robô é um processo resultante de características intrínsecas de *hardware* e *software*. Esses são fatores que podem influenciar no processo de atualização de conhecimento na base de dados, como recursos limitadores.

Para o robô, a Assimilação compreende na aquisição de um conhecimento novo. Já a Acomodação é o aprimoramento de uma determinada tarefa, conhecida previamente. O equilíbrio entre Assimilação e Acomodação é induzido, diferente do equilíbrio pregado por Piaget, que é necessário. A Figura 3.6 mostra esses conceitos mapeados do indivíduo para o robô.

Abordando os conceitos relacionados às teorias de Piaget (1975), Piaget & Inhelder (1982) e J.Piaget (1985), e obtendo os requisitos básicos para que essas teorias sejam bem estruturadas no contexto do CETP, algumas relações precisam ser bem definidas. Sabendo que o conjunto de conhecimento do robô α_i é especificado por $C(\alpha_i) = c_1, c_2, \ldots, c_x$, é necessário atentar para as seguintes definições sobre a Assimilação e a Acomodação de conhecimentos do robô, bem como sobre os resultados esperados dessas operações.

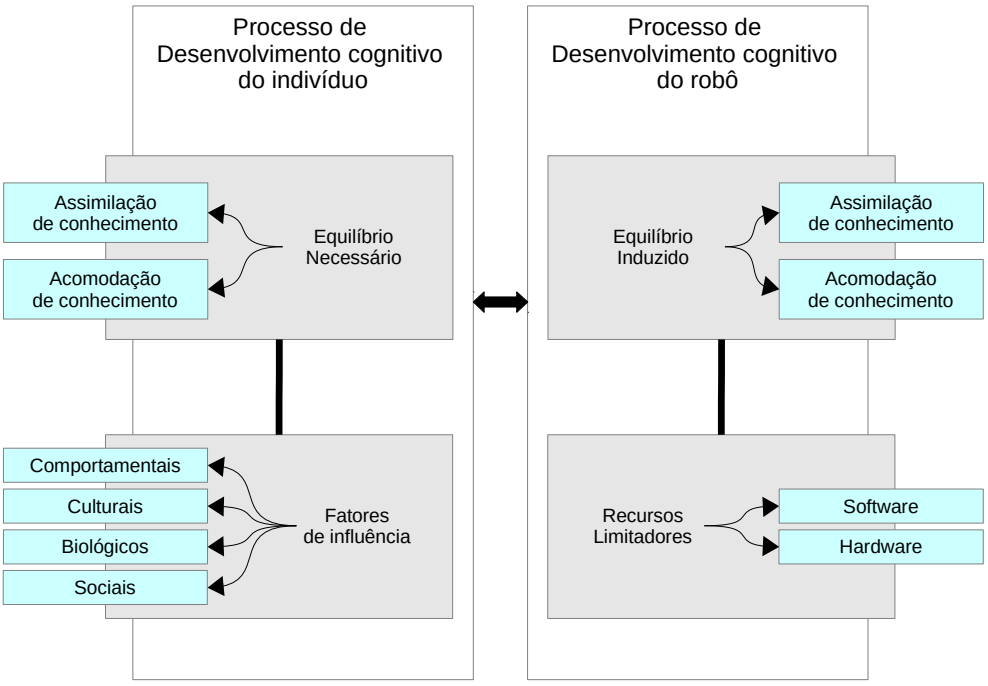

Figura 3.6: A Teoria do Equilíbrio, pregada por Piaget e mapeada para robôs.

3.4.2 Assimilação de conhecimento

Para que haja Assimilação na base de conhecimento de um robô, a função que define sua Zona de Desenvolvimento Proximal (ZPD) deve assumir valor -1. Essa é a afirmação de que o robô α_i não sabe resolver a tarefa τ_j, mas é capaz de aprendê-la, uma vez que ele possui a capacidade física necessária.

Com isso, a função do Nível de Desenvolvimento Potencial (LPD) terá valor -1, já que o robô α_i tem capacidade de aprender a tarefa τ_j. Já o LRD, por sua vez, terá valor 0, já que o robô α_i não pode resolver a tarefa τ_j sozinho, uma vez que não detém todo conhecimento necessário. Assim, resumindo essas informações e considerando um robô α_i e uma dada tarefa τ_j, a condição de Assimilação é:

- **Condição 1. O robô não sabe resolver a tarefa.**
  Se $f\_lrd(\alpha_i, \tau_j) = 0$, então α_i não sabe resolver τ_i.

- **Condição 2. O robô tem pontencial para aprender a tarefa.**
  Se $f\_lpd(\alpha_i, \tau_j) = -1$, então α_i pode aprender a resolver τ_i.

- **Condição 3. O robô não sabe executar a tarefa, mas é capaz de aprender.**
  Se $f\_zpd(\alpha_i, \tau_j) = -1$, então α_i não sabe executar τ_i, mas é capaz de aprender.

Logo, em um dado instante t, para que α_i possa adicionar o conhecimento c', é necessário satisfazer a seguinte restrição: $c' \notin C(\alpha_i)$. Então, caso essa condição seja verdade, no próximo instante $t+1$, tem-se a Equação (3.5).

$$C(\alpha_i) = C(\alpha_i) \cup \{c'\} \tag{3.5}$$

Finalmente, a função da Assimilação está definida pela Equação (3.6). Sabendo que o robô α_x é mais experiente que o robô α_i (pelo menos em se tratando da execução da tarefa τ_j), $h = 1$ implica em dizer que α_i pôde receber conhecimento de α_x sobre τ_j, ou seja, foi possível realizar $C(\alpha_i) = C(\alpha_i) \cup \{c'\}$, onde c' é o conhecimento novo acerca de τ_j. Caso contrário ($h = 0$), não foi possível aplicar a Assimilação, devido a α_i já possuir c'. Para saber se c' é um conhecimento novo, basta verificar a condição: $c' \notin C(\alpha_i)$. Concluindo acerca dessas afirmações, tem-se as questões a seguir.

$$f\_assimilation(\alpha_i, \alpha_x, \tau_j) = h \tag{3.6}$$

- **Questão 1. O robô assimilou a execução de uma tarefa com a ajuda de outro.**
  Se $f\_assimilation(\alpha_i, \alpha_x, \tau_j) = 1$, então α_i aprendeu a resolver τ_j com a ajuda de α_x.

- **Questão 2. O robô não assimilou a execução de uma tarefa, mesmo com a ajuda de outro.**
  Se $f\_assimilation(\alpha_i, \alpha_x, \tau_j) = 0$, então α_i não aprendeu a resolver τ_j com a ajuda de α_x.

A Figura 3.7 demonstra uma situação em que um robô recebe uma solicitação de execução de uma tarefa específica. Já a Figura 3.8 demonstra uma condição que caracteriza a necessidade de Assimilação, em que um robô *Robô i* está diante de uma tarefa *Tarefa j* e não sabe como executá-la. Essa situação se configura em tempo t. Após interação com outro robô que detém o conhecimento necessário (*Robô x*), o *Robô i* passa a assimilar o conhecimento necessário. Essa situação se configura o tempo $t+1$, exposta na Figura 3.9.

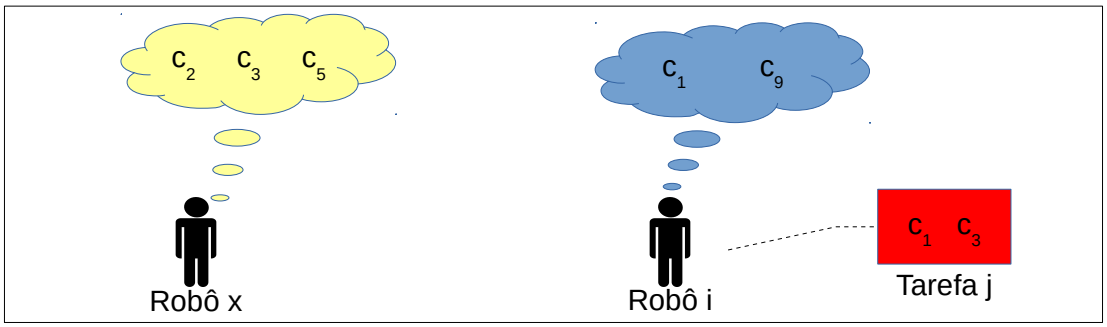

Figura 3.7: Situação em que um robô se depara com uma tarefa solicitando execução.

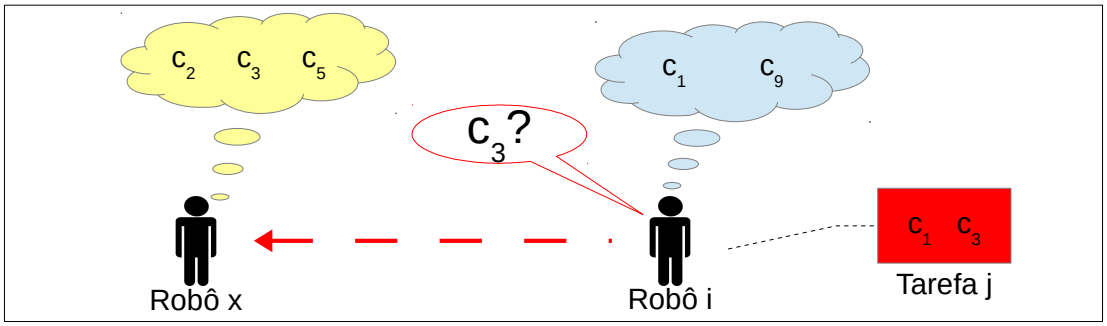

Figura 3.8: Situação t que caracteriza a necessidade de Assimilação de conhecimento.

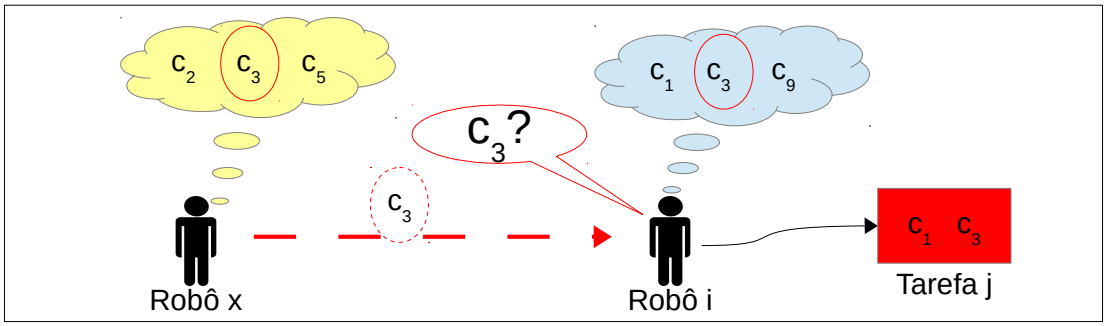

Figura 3.9: Situação $t+1$ após Assimilação de conhecimento.

3.4.3 Acomodação de conhecimento

O requisito principal para que seja possível acomodar conhecimento, é que a função que define a Zona de Desenvolvimento Proximal (ZPD) assuma valor -2. Ou seja, $f\_zpd(\alpha_i, \tau_j) = -2$, o que significa que o robô α_i já sabe resolver a tarefa τ_j e ainda é capaz de aprimorá-la. Com esse dado, sabe-se que a função do Nível de Desenvolvimento Potencial (LPD) terá valor -1 ($f\_lpd(\alpha_i, \tau_j) = -1$), já que o robô α_i tem capacidade de aprender a tarefa τ_j; e que o Nível de Desenvolvimento Real (LRD) terá valor 1 ($f\_lrd(\alpha_i, \tau_j) = 1$), já que o robô α_i pode executar a tarefa τ_j sozinho. Assim, para um robô α_i e uma dada tarefa τ_j, as condições para haver Acomodação é:

- **Condição 1. O robô sabe executar a tarefa autonomamente.**
  Se $f\_lrd(\alpha_i, \tau_j) = 1$, então α_i sabe resolver τ_i.

- **Condição 2. O robô pode aprender o conhecimento sobre a tarefa.**
  Se $f\_lpd(\alpha_i, \tau_j) = -1$, então α_i pode aprender a resolver τ_i.

- **Condição 3. O robô sabe resolver a tarefa e pode aprimorar sua forma de resolução.**
  Se $f\_zpd(\alpha_i, \tau_j) = -2$, então α_i sabe executar τ_i e ainda é capaz de aprimorar sua resolução.

Em um dado instante t, para que α_i possa alterar seu conhecimento c por um conhecimento novo c', que é informado por outro robô, é necessário satisfazer as seguintes restrições: $c \in C(\alpha_i)$ e $c' \notin C(\alpha_i)$. Então, caso essa condição seja verdade, no próximo instante $t+1$, tem-se a Equação (3.7), onde c' é o conhecimento novo e c é um conhecimento já instalado na base de dados do robô α_i.

$$C(\alpha_i) = \{C(\alpha_i) - \{c\}\} \cup \{c'\} \tag{3.7}$$

A função da Acomodação está definida pela Equação (3.8), tal que, se $s = 1$ (ou $f\_accommodation(\alpha_i, c, c') = 1$), então foi possível realizar $C(\alpha_i) = \{C(\alpha_i) - \{c\}\} \cup \{c'\}$. Caso não foi possível admitir o conhecimento novo, $s = 0$ ($f\_accommodation(\alpha_i, c, c') = 0$). Como conclusão, são informadas as questões a seguir.

$$f\_accommodation(\alpha_i, c, c') = s \tag{3.8}$$

- **Questão 1. O robô acomodou a execução de uma tarefa com a ajuda de outro.**
  Se $f\_accommodation(\alpha_i, c, c') = 1$, então α_i aprendeu a resolver τ_j com a ajuda de α_x e substituiu seu conhecimento anterior.

- **Questão 2. O robô não acomodou a execução de uma tarefa, mesmo com a ajuda de outro.**
  Se $f\_accommodation(\alpha_i, c, c') = 0$, então α_i não aprendeu a resolver τ_j com a ajuda de α_x, ficando com seu conhecimento anterior.

A Figura 3.10 mostra uma situação em que um robô pode alterar seu conhecimento acerca de uma determinada tarefa. Nesse caso, já que as restrições foram estabelecidas: $c_3 \in C(Robo_i)$ e $c'_3 \notin C(Robo_i)$, o robô $Robo_i$ pode acomodar c'_3 em sua base de dados. Sendo assim, $C(Robo_i) = \{C(Robo_i) - \{c_3\}\} \cup \{c'_3\}$.

Já a Figura 3.11 expõe a situação após a Acomodação do conhecimento (da Figura 3.10). A operação é realizada através do conhecimento passado por outro robô ($Robo_x$). A Acomodação se exemplifica bem utilizando o termo de substituição de conhecimento. Nesse exemplo, o sucesso da operação pode ser facilmente identificado através do resultado $f\_accommodation(Robo_i, c_3, c'_3) = 1$.

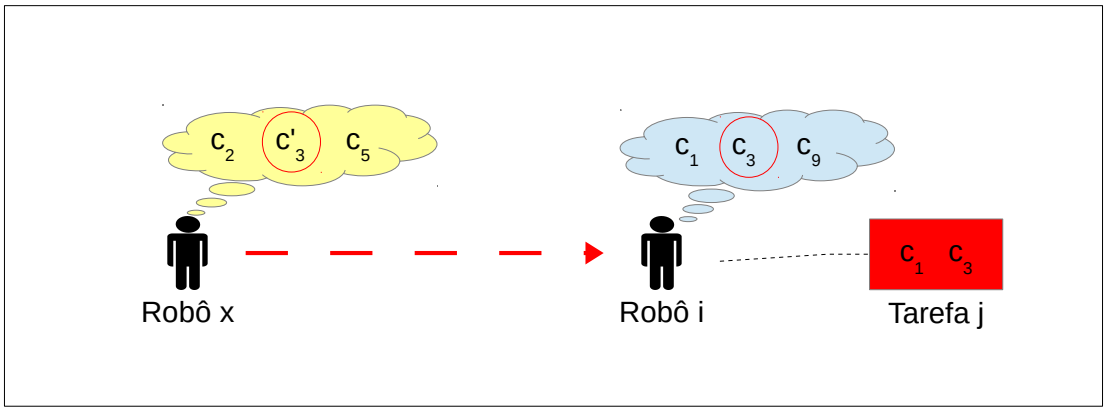

Figura 3.10: Situação t que caracteriza a possibilidade de Acomodação de conhecimento.

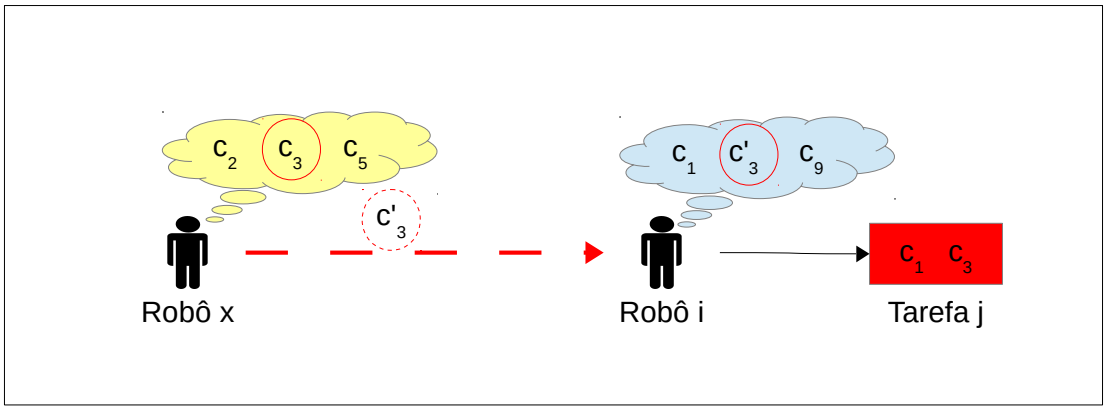

Figura 3.11: Situação $t+1$ após Acomodação de conhecimento.

3.5 Abordagem Behaviorista aplicada em robôs

A abordagem Behaviorista clássica pregada por John Broadus Watson defende que a psicologia não deve focar em processos internos da mente, mas sim no comportamento, pois este é visível e, portanto, passível de observação [Watson 1913]. Nesse caso, o fator gerador do comportamento humano é a resposta a um estímulo, sendo possível prever e controlar toda a conduta humana, com base no estudo do meio em que esse indivíduo vive. Assim, qualquer modificação resultante de um estímulo do meio ambiente pode provocar manifestações do comportamento humano.

No entanto, o indivíduo precisa receber um estímulo para produzir uma resposta. De fato, pode-se entender como estímulo uma solicitação de conhecimento, seja para resolver tarefas ou para passá-lo adiante como experiência.

O Behaviorismo Clássico, pregado por John Broadus Watson, evidencia que antes de qualquer informação disponibilizada ao meio em que vive, o indivíduo está submetido a uma condição de passividade. Após o entendimento dessa informação, esse indivíduo passará a assumir um papel ativo na sociedade, podendo reproduzir esse conhecimento quando lhe for conveniente.

A Figura 3.12 mostra a ideia principal dessa abordagem. O indivíduo que se encontra na condição passiva no meio social, ao receber algum estímulo de uma informação nova, passa a ser um indivíduo atuante no ambiente, podendo produzir conhecimento para a sociedade em que atua.

Fazendo o mapeamento para sistemas multirrobôs, é possível definir duas situações que configuram a perspectiva do estímulo definido por John Watson. São elas: Estímulo-Conhecimento e Estímulo-Resposta.

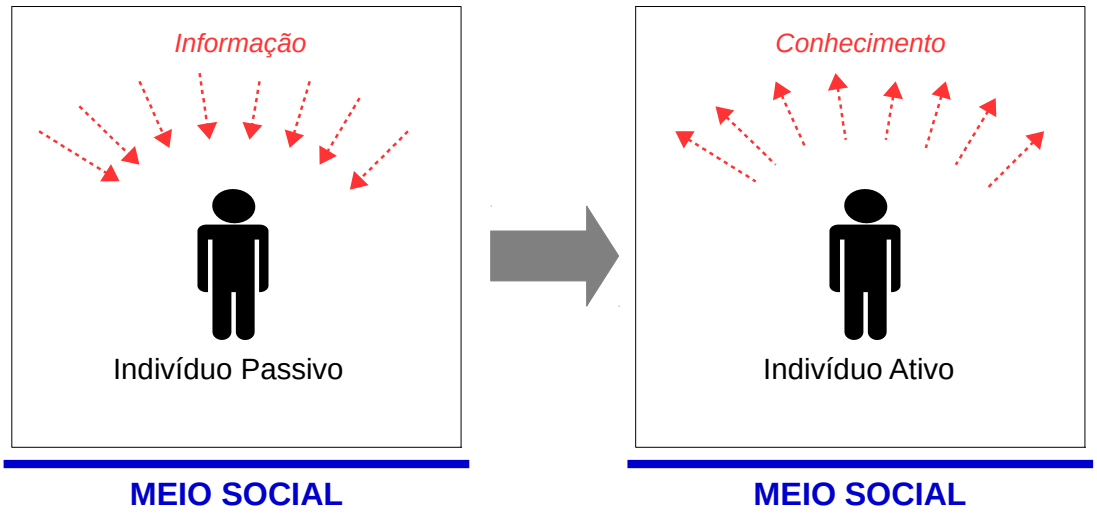

Figura 3.12: Ideia principal evidenciada pelo Behaviorismo Clássico.

3.5.1 Estímulo-Conhecimento

A função de Estímulo-Conhecimento está definida na Equação (3.9) como sendo a resposta do robô α_i ao estímulo da tarefa τ_j. K é definido como o conjunto de conhecimento que α_i possui acerca de τ_j, ou seja, $K = \{c_1, c_2, ..., c_k\}$, tal que k é a quantidade de conhecimento.

$$f\_\varepsilon_c(\tau_j, \alpha_i) = K \tag{3.9}$$

Algumas questões importantes podem ser elencadas e estão definidas mais abaixo, sabendo que $C(\alpha_i) \supseteq \{c_1, c_2, ..., c_k\}$ e que $C(\tau_j) \supseteq \{c_1, c_2, ..., c_k\}$, simultaneamente.

- **Questão 1. O robô não possui conhecimento acerda da tarefa.**
  Se $f\_\varepsilon_c(\tau_j, \alpha_i) = \emptyset$, então α_i não sabe como resolver τ_j.

- **Questão 2. O robô possui conhecimento acerda da tarefa.**
  Se $f\_\varepsilon_c(\tau_j, \alpha_i) \neq \emptyset$, então α_i sabe como resolver τ_j. Nesse caso, a função resulta no conjunto de conhecimento de α_i em relação a τ_j.

A Figura 3.13 mostra o mapeamento das ideias behavioristas de John Watson para sistemas multirrobôs, considerando a ideia do Estímulo-Conhecimento. O robô recebe uma solicitação de alguma tarefa e responde com o seu conhecimento a cerca da resolução, mesmo sem ter autonomia para a execução. Sendo assim, o retorno dessa operação é toda informação que o robô tem acerca da tarefa específica.

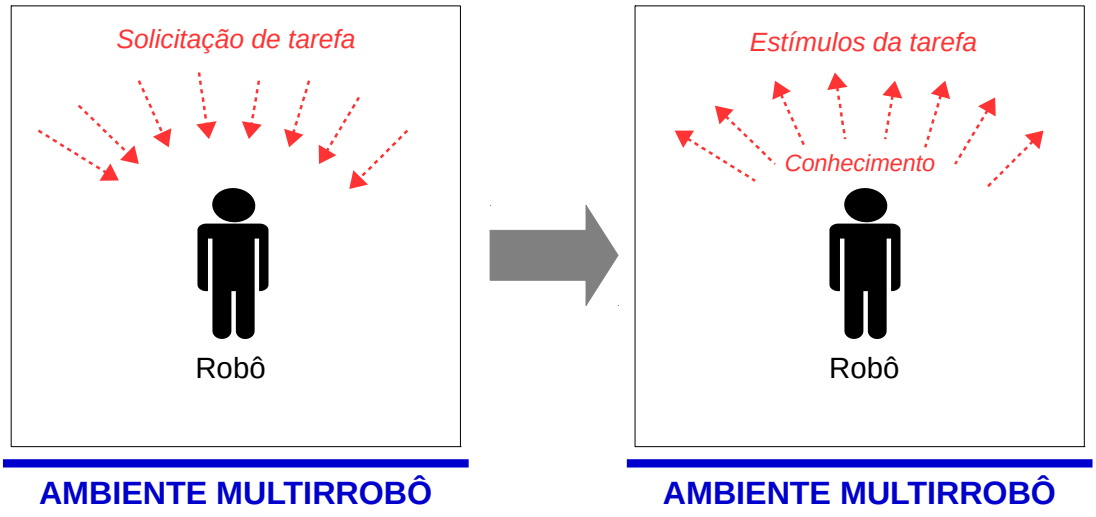

Figura 3.13: O Behaviorismo Clássico na forma do Estímulo-Conhecimento mapeado no ambiente multirrobô.

3.5.2 Estímulo-Resposta

O Behaviorismo Clássico, na forma do Estímulo-Resposta, ocorre quando o robô recebe um estímulo de uma tarefa específica e responde com o parecer de possuir, ou não, o conhecimento necessário para a execução, seja autonomamente ou não.

A função de Estímulo-Resposta está definida na Equação (3.10) como sendo a resposta do robô α_i ao estímulo da tarefa τ_j, sendo possível obter 1 ou 0 como respostas, já que $k \in \{0,1\}$. Algumas questões importantes podem ser elencadas acerca dessa equação e estão definidas mais abaixo.

$$f\_\varepsilon_r(\tau_j, \alpha_i) = k \tag{3.10}$$

- **Questão 1. O robô possui o conhecimento necessário para resolver a tarefa.** Se $f\_\varepsilon_r(\tau_j, \alpha_i) = 1$, então α_i sabe como resolver τ_j. Nesse caso, em se tratando de Estímulo-Conhecimento, ocorre $f\_\varepsilon_c(\tau_j, \alpha_i) \neq \emptyset$.

- **Questão 2. O robô não possui todo conhecimento para resolver a tarefa.** Se $f\_\varepsilon_r(\tau_j, \alpha_i) = 0$, então α_i não sabe como resolver τ_j. No Estímulo-Conhecimento, por sua vez, ocorre $f\_\varepsilon_c(\tau_j, \alpha_i) = \emptyset$.

A Figura 3.14 mostra o mapeamento das ideias behavioristas de John Watson para sistemas multirrobôs, considerando a ideia do Estímulo-Resposta. O robô recebe uma solicitação de alguma tarefa e responde se possui ou não conhecimento a cerca da resolução, desconsiderando a importância da autonomia para a execução. Sendo assim, o retorno dessa operação é uma resposta que significa "Sim, conheço essa tarefa"ou "Não, não conheço essa tarefa".

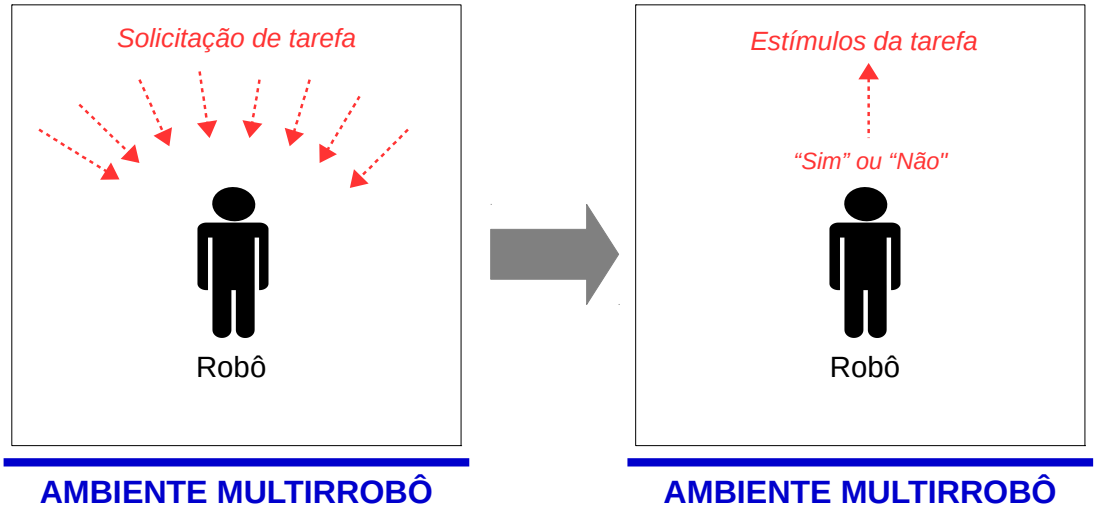

Figura 3.14: O Behaviorismo Clássico na forma do Estímulo-Resposta mapeado no ambiente multirrobô.

3.5.3 Comparação entre Estímulo-Resposta e Nível de Desenvolvimento Real

Em se tratando da função Estímulo-Resposta, da Equação (3.10), supondo que o robô α_i saiba executar a tarefa τ_j, então sabe-se que $f\_\varepsilon_r(\tau_j, \alpha_i) = 1$. Porém, não é possível afirmar se esse robô tem total autonomia para resolvê-la sozinho, sem a ajuda de outro robô mais experiente do ambiente. Dessa forma, é possível que ele necessite dividir a tarefa τ_i em outras menores, para receber, quando necessário, a intervenção na execução por parte de outros robôs.

Essa situação descrita difere da função Nível de Desenvolvimento Real, da Equação (3.2), quando $f\_lrd(\alpha_i, \tau_j) = 1$, pois esse resultado informa que α_i é capaz de resolver a tarefa τ_j autonomamente, ou seja, sem a ajuda de outro robô do ambiente.

Além dessas indagações, alguns resultados concebidos por uma função envolvida com um contexto, implica na resposta de outra, em outro contexto. Tanto a função Estímulo-Resposta por concluir sobre a função LRD, quanto vice-versa. Sendo assim, algumas questões importantes são elucidadas a seguir.

- **Questão 1.Se o robô tem autonomia para executar a tarefa, então ele produz uma resposta ao seu estímulo.**
  Se $f\_lrd(\alpha_i, \tau_j) = 1$, então $f\_\varepsilon_r(\tau_j, \alpha_i) = 1$, já que, se α_i tem todo conhecimento requerido por τ_j ($f\_lrd(\alpha_i, \tau_j) = 1$), ele necessariamente deve produzir um estímulo a essa tarefa ($f\_\varepsilon_r(\tau_j, \alpha_i) = 1$).

- **Questão 2. Se o robô não tem autonomia para executar a tarefa, ele pode produzir (ou não) uma resposta ao estímulo dessa tarefa.**
  O resultado $f\_lrd(\alpha_i, \tau_j) = 0$ não restringe qualquer valor para $f\_\varepsilon_r(\tau_j, \alpha_i)$, pois mesmo sem conseguir resolver τ_j sozinho ($f\_lrd(\alpha_i, \tau_j) = 0$), α_i pode, ou não, produzir uma resposta ao estímulo dessa tarefa. Nesse caso, α_i não pode resolver τ_j sozinho. Se ele soubesse resolvê-la, ele não possuiria a capacidade física para tal execução.

- **Questão 3. Se o robô não possui resposta ao estímulo da tarefa, então ele não tem autonomia para executá-la.**
  Se $f\_\varepsilon_r(\tau_j, \alpha_i) = 0$, então $f\_lrd(\alpha_i, \tau_j) = 0$, pois se α_i não pode informar uma resposta ao estímulo de τ_j ($f\_\varepsilon_r(\tau_j, \alpha_i) = 0$), então ele também não tem o conhecimento para sua execução autônoma ($f\_lrd(\alpha_i, \tau_j) = 0$).

- **Questão 4. Mesmo o robô possuindo resposta ao estímulo da tarefa, não se pode afirmar que ele tem autonomia sobre essa execução.**
  O resultado $f\_\varepsilon_r(\tau_j, \alpha_i) = 1$ não restringe qualquer valor para $f\_lrd(\alpha_i, \tau_j)$, já que, mesmo possuindo uma resposta para o estímulo da tarefa τ_j ($f\_\varepsilon_r(\tau_j, \alpha_i) = 1$), o robô α_i não necessariamente pode resolvê-la sozinho.

3.6 Abordagem Social aplicada em robôs

Segundo Wenger (2000), a aprendizagem Social é uma transformação diretamente relacionada às interações cotidianas entre as pessoas de um mesmo ambiente social. Essa abordagem enfatiza que o indivíduo aprende observando e interagindo com outras pessoas de seu contexto social. Não se enquadram questões biológicas, já que o indivíduo nasce com os conhecimentos básicos (iguais para todos os seres humanos) e vai se moldando de acordo com o contato estabelecido com a sociedade. A resposta de um indivíduo a um estímulo está em conformidade com o ambiente em que convive.

Mapeado para sistemas multirrobôs, entende-se que o contexto social dos robôs é um conjunto de fatores em que estão inseridos, sendo ele reconfigurável e dinâmico. Como todos os robôs estão inseridos em um mesmo ambiente, o contexto social é o mesmo para cada robô do sistema, sendo especificado por Φ. A resposta de um robô α_i a um estímulo oriundo do ambiente, está relacionado, portanto, com seu conhecimento acerca do assunto, levando em consideração o contexto social.

A tarefa necessita de uma série de conhecimentos básicos que o robô, ou um grupo de robôs, precisa ter em seu conjunto para executá-la. Ou seja, se $C(\tau_j) = \{c_1, c_2, \ldots, c_x\}$ e $F(\tau_j) = \{f_1, f_2, \ldots, f_y\}$, são os conjuntos de conhecimentos e capacidades físicas requisitadas pela tarefa, respectivamente, então o robô, ou um grupo, deve conter todos esses dados.

A Equação (3.11) verifica se a tarefa τ_j pode ou não ser executada por um ou mais robôs. Esses robôs compõem o conjunto A', sabendo que $k \in \{0,1\}$ e que $A \supseteq A'$. Algumas considerações importantes estão descritas abaixo.

$$f\_\varepsilon_{gr}(\tau_j, A') = k \tag{3.11}$$

- **Questão 1. A tarefa pode ser executada pelo ambiente multirrobô.**
  Se $k = 1$, ou seja $f\_\varepsilon_{gr}(\tau_j, A') = 1$, então τ_j pode ser executada pelo grupo de robôs A'. Com isso, são satisfeitas as duas condições: $C(\tau_j) \subseteq \{\bigcup_{\alpha_i \in A'} C(\alpha_i)\}$ e $F(\tau_j) \subseteq \{\bigcup_{\alpha_i \in A'} F(\alpha_i)\}$, simultaneamente.

- **Questão 2. A tarefa não pode ser executada pelo ambiente multirrobô.**
  Se $k = 0$, ou seja $f\_\varepsilon_{gr}(\tau_j, A') = 0$, então o subconjunto de robôs A' não é capaz de resolver τ_j. Nesse caso, é conclusivo que não existe algum grupo de robôs capaz de solucionar τ_j.

Diferentemente da Equação (3.10), essa função avalia um grupo de robôs (ou todos). Admitindo que, em sistemas multirrobôs, se um robô não possui conhecimento porém possui capacidade física necessária para executar uma determinada tarefa, ele pode executá-la, desde que ele seja ajudado por um outro robô que lhe disponibilizará o conhecimento necessário.

3.7 Abordagem Humanista aplicada em robôs

A abordagem Humanista (ou espontânea) parte do potencial humano de aprendizagem. Nessa abordagem o indivíduo decide o que quer aprender, controla o seu próprio destino, possui liberdade para agir e seu comportamento no decorrer da vida é consequência de suas escolhas. Esse processo de aprendizagem leva em consideração as características individuais, as experiências anteriores e a motivação que o indivíduo precisa para desenvolver um conhecimento acerca do assunto.

De acordo essa abordagem, o indivíduo controla suas ações e possui liberdade para agir, considerando seu potencial. No caso dos robôs, eles também têm liberdade para tomar decisões de acordo com a implementação de algumas regras do ambiente que sejam motivantes para uma cooperação eficiente. Essas regras estão embutidas nos algoritmos implementados para o Modelo de Desenvolvimento Inteligente IDeM-MRS, disponíveis no Capítulo 4.

Capítulo 4

IDeM-MRS

O foco deste trabalho está no gerenciamento da cooperação entre o grupo, buscando definir um modelo ideal que permite que os robôs planejem suas ações, aloquem suas listas de prioridades e executem as tarefas, autonomamente e eficientemente, e enfim, finalizem o objetivo global. Este problema, mesmo sem investigar sobre a escolha mais eficiente do robô para a execução, basicamente é o alvo da pergunta:

"Qual robô executa cada tarefa e em que momento?"

Sem perda de generalidade, este trabalho assume que um problema pode ser dividido em outro sub-problemas, já que, primeiramente a missão deve ser decomposta em tarefas menores, sendo que estas podem ser executadas por robôs diferentes. Depois desta quebra em tarefas menores, essas devem ser alocadas a cada robô para execução.

Durante uma execução real de um sistema multirrobô, muitos imprevistos podem surgir devido a fatores externos ao ambiente ou a fatores intrínsecos do sistema. Esta pesquisa não aborda formas de prevenção desses imprevistos, uma vez que se trata de um modelo teórico, validado através de experimentos com sistemas multirrobôs simulados. Além disso, permitir a aquisição de conhecimento de um robô por outro, sem que haja qualquer tipo de auxílio externo, é outra característica desejável, presumida e estudada neste trabalho.

O Problema de Execução Cooperativa de Tarefas por um grupo de robôs também pode ser associado ao Problema de Agendamento de Projetos com Recursos Limitados (*Resource Constrained Project Scheduling Problem* - RCPSP), conforme Brucker (2002) relacionou em seu trabalho. Segundo Weglarz et al. (2011), RCPSP pode ser classificado como um problema de difícil solução, classificado como NP-Hard em Pesquisa Operacional (*Operations Research* - OR).

Alguns fatores compõem o esquema das principais dificuldades do problema de alocação de tarefas por times de robôs, que são:

- a imprevisibilidade do exato tempo de processamento das tarefas;
- a ocorrência de resultados instáveis durante a execução;
- a existência de inconsistências devido a informações incertas do ambiente.

Pensando nesse esquema, Talay et al. (2011) define as principais situações que podem alterar a solução global de um problema de alocação de tarefas do mundo real. Mais tarde ele utiliza uma formulação do Problema de Execução Cooperativa da Missão (*Cooperative Mission Achievement Problem* - CMAP), implementada através de uma versão adaptada de RCPSP. Então, ele elabora um framework para a cooperação de um sistema multirrobô. Esse framework aborda os componentes do ambiente de forma distribuída, porém não utiliza a inclusão de conhecimentos nos robôs, seja individualmente ou em grupo.

O trabalho de Talay faz refletir sobre alguns pontos necessários a esta pesquisa. Quando se trata de ambientes multirrobôs simulados frente a ambientes multirrobôs reais, várias circunstâncias podem afetar a solução manipulada (simulada) no ambiente, quando comparada com a do mundo real. Algumas delas são:

1. Os robôs não detectam suas próprias falhas nem as de outros robôs.

2. O tempo estimado para execução de cada tarefa pode ser alterado devido a:

   - ambientes dinâmicos;
   - conhecimento duvidoso acerca da execução da tarefa;
   - problemas imprevistos de hardware;
   - incertezas nas informações, como por exemplo na localização.

3. A definição das tarefas pode ser alterada, como as dependências (de outras tarefas), as prioridades ou alguns objetivos específicos. Pode ocorrer, também, de alguma tarefa se tornar desnecessária durante a execução.

4. Novos robôs podem ser incluídos durante a execução ou retirados do ambiente, caso necessitem de algum tipo de recuperação de hardware.

5. Algum agente externo ao ambiente pode realizar intervenções sem qualquer pré-definição, alterando a configuração inicial.

Essas situações podem surgir no decorrer do problema, em qualquer momento. Com esses fatores, mesmo o resultado de uma abordagem capaz de encontrar a solução ótima, pode tornar-se ineficaz devido às incertezas das aplicações do mundo real. Além do mais, a busca pela otimalidade é uma questão difícil, também, para as aplicações do mundo real.

Assim, o Problema de Execução Cooperativa de Tarefas por um sistema multirrobô, denominado de *Cooperative Mission Achievement Problem* - CETP por esta pesquisa, pode ser compreendido como o Problema de Seleção Coordenada de Tarefas (CTSP), distribuído em cada robô do ambiente, em que todos os robôs cooperam para a resolução do Problema de Execução Cooperativa de Missão (CMAP). Ambos os problemas, tanto o CTSP quanto o CMAP, foram delineados por Talay et al. (2007) e Talay et al. (2011).

Por outro lado, o CETP também pode ser modelado baseando-se no Problema de Agendamento de Projetos com Recursos Limitados (RCPSP), abordado em Brucker (2002)

e Weglarz et al. (2011). A Figura 4.1 mostra uma modelagem, na qual, através da solicitação de uma dada missão, os robôs colaboram entre si para a completa resolução. Já a Figura 4.2 mostra a compreensão do CETP segundo a visão do CMAP. Nesse caso, além da colaboração do grupo, cada um robô individualmente realiza suas listas de prioridades para a resolução das tarefas específicas que compõem a missão global.

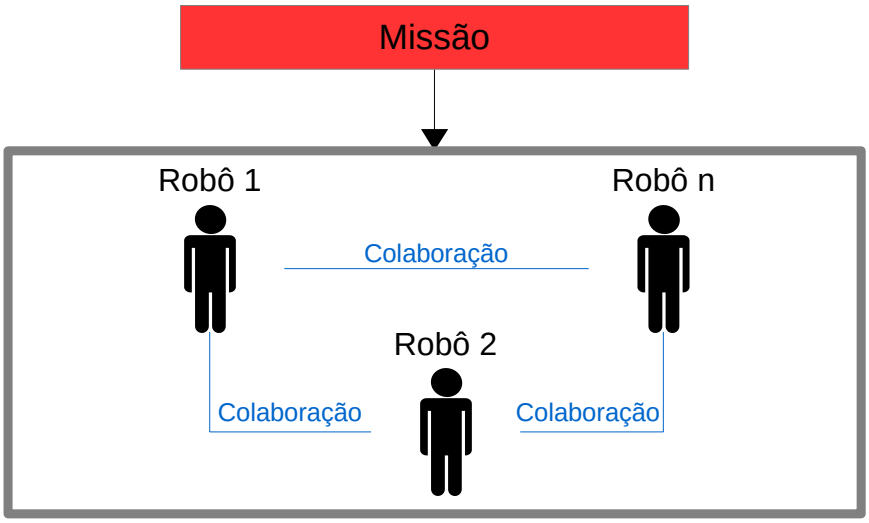

Figura 4.1: Definição do Problema de Execução Cooperativa de Tarefas segundo a visão do Problema de Agendamento de Projetos com Recursos Limitados.

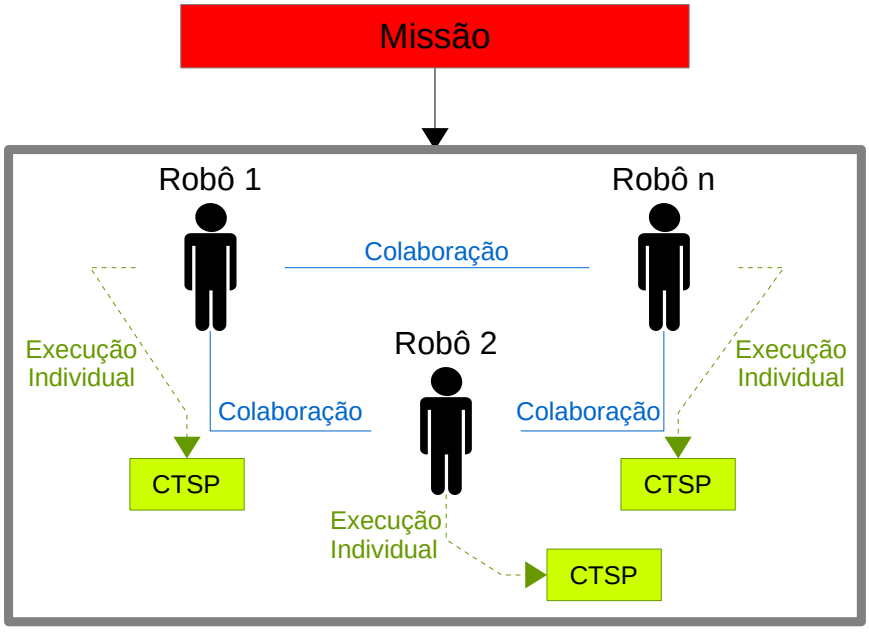

Figura 4.2: Definição do Problema de Execução Cooperativa de Tarefas segundo a visão do Problema de Execução Cooperativa de Missão.

4.1 Especificações do CETP

O Problema de Execução Cooperativa de Tarefas (CETP) é um problema NP-Hard, conforme Brucker (2004) informou para o RCPSP, e consiste em obter a execução de M pelos robôs A. No entanto, como se trata de um problema de difícil solução, uma otimização da cooperação dos robôs para resolução de tarefas é viável e pode resultar em melhoria para o ambiente, que em primeiro momento é simulado. A missão, as tarefas e os robôs são definidos com algumas condições (as próximas seções informam essas especificações).

No entanto, algumas observações devem ser levadas em consideração, quando se esta diante de um CETP:

- Um mesmo robô α_i não pode ser designado para executar mais de uma tarefa no mesmo instante t;

- A missão M é uma lista de tarefas para serem executadas em ordem pré-estabelecida. Portanto, uma tarefa só poderá ser executada caso a sua anterior na lista tenha sido finalizada, por completo, por algum robô. Barbosa et al. (2012) propôs uma forma que execução da missão em que algumas tarefas poderiam ser executadas paralelamente, diminuindo o tempo final de M. Contudo, esse trabalho não trata a especificação de tarefas que podem ser executadas em pararelo com outras;

- Uma tarefa só pode ser executada por um robô que possua todos elementos requeridos por ela (conhecimento e capacidade física).

- A cooperação deve ser uma maneira eficiente de transportar conhecimento entre os robôs do ambiente, de modo a aumentar a quantidade de robôs com possibilidades de execução das tarefas. Além disso, essa relação deve ser apenas de robô para robô, sem a atuação de algum agente externo ao ambiente multirrobô.

- Busca-se finalizar a execução de todas as tarefas da missão M no menor tempo possível e obter maior proveito do conhecimento que está inserido no ambiente (através de cada robô individualmente).

Considerando as especificações dos problemas de Resolução Cooperativa da Missão e Agendamento de Projetos com Recursos Limitados, o CETP pode ser formalizado, conforme Maia et al. (2011), através de dois conjuntos principais, descritos na Equação (4.1) e na Equação (4.2). Esses conjuntos indicam um time de robôs $A = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$, tal que m é a quantidade de robôs que atuam no ambiente ($m \in \mathbb{N}$) , e um conjunto de tarefas $T = \{\tau_1, \tau_2, \ldots, \tau_n\}$, tal que n é a quantidade de tarefas que compõem a missão M ($n \in \mathbb{N}$).

$$A = \{\alpha_1, \alpha_2, \ldots, \alpha_m\} \tag{4.1}$$

$$T = \{\tau_1, \tau_2, \ldots, \tau_n\} \tag{4.2}$$

4.1.1 Especificações das Tarefas

Sabe-se que o objetivo global é uma missão M composta por uma lista de tarefas. Para que M seja concluída, algumas características devem ser elencadas, como a ordem de precedência das execuções das tarefas e algumas definições específicas para cada uma, seja a capacidade física necessária ou o conhecimento intelectual requerido para sua resolução.

- **Ordem de precedência:**

  Deve ser definida uma restrição de precedência para as tarefas que compõem a missão. Sendo assim, $M = \{\tau_i, \tau_k, \tau_j\}$ é mesmo que afirmar que $\tau_i < \tau_k$, ou seja, τ_i tem prioridade maior em relação a τ_k. Nesse caso, τ_i precede τ_k, o que significa que τ_k não pode começar sua execução antes de τ_i ter sido finalizada. O mesmo se aplica a τ_k e τ_j. Igualmente, τ_j não deve iniciar antes da completa execução de τ_k.

  A Equação (4.3) informa o tempo total necessário para finalização da missão M, como sendo $fmission\_time$, considerando que m é a quantidade de tarefas da missão M e $time(\tau_k)$ é a função que retorna o tempo necessário para finalizar a tarefa τ_k.

$$fmission\_time = \sum_{k=1}^{n} time(\tau_k) \tag{4.3}$$

  A Figura 4.3 mostra um exemplo da ordem de precedência das tarefas. A missão $M = \{\tau_0, \tau_1, \ldots, \tau_k, \tau_{k+1}, \ldots, \tau_{n-1}, \tau_n\}$ possui como tempo final de execução, a soma de todos os tempos de execução de cada tarefa, porém, seguindo a ordem de execução.

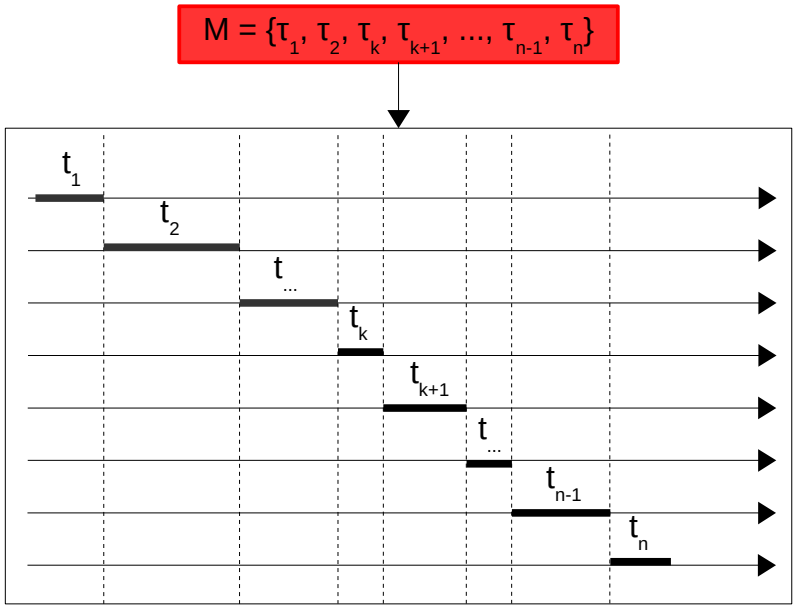

Figura 4.3: Ordem de execução das tarefas de uma missão M.

- **Conhecimentos necessários:**

  A tarefa deve especificar o conhecimento necessário para que um robô possa executá-la. Esse conhecimento é obtido pelo conjunto $C(\tau_j)$, especificado na Equação (4.4), onde k é a quantidade de conhecimentos necessários para execução da tarefa τ_j. Com isso, o robô precisa deter esses conhecimentos para executar a respectiva tarefa.

  Um exemplo de tarefa seria: "varrer uma sala de aula". Nesse caso, o robô precisa, pelo menos, saber varrer um espaço, conhecer uma vassoura e identificar um espaço sujo. Sem esses conhecimentos básicos, o robô não poderá executar a tarefa.

$$C(\tau_j) = \{c_1, c_2, \ldots, c_k\} \tag{4.4}$$

- **Capacidades físicas:**

  A tarefa também deve especificar quais as capacidades físicas que sua execução necessita. Contudo, além do conhecimento necessário, o robô também deve possuir a característica de hardware solicitada pela tarefa para efetuar sua execução.

  O conjunto $F(\tau_j)$ informa essas características requeridas e está especificado conforme a Equação (4.5), onde w é a quantidade de características físicas requeridas pela tarefa τ_j.

  Seguindo a mesma linha de raciocínio do exemplo anterior, o robô estaria capacitado para executar a tarefa "varrer uma sala de aula", se ele tiver algum dispositivo de *hardware* capaz de segurar uma vassoura e outro capaz de oferecer a sua movimentação no recinto.

$$F(\tau_j) = \{f_1, f_2, \ldots, f_w\} \tag{4.5}$$

4.1.2 Especificações dos Robôs

Diferentemente das tarefas, os robôs não necessitam de ordem de precedência para a resolução de uma determinada tarefa. Eles cooperam juntos pela resolução da missão, como objetivo global. No entanto, eles necessitam especificar os conjuntos de conhecimento e de características físicas, para identificar suas aptidões nas execuções da missão.

Cada robô possui um conjunto de conhecimentos básicos que são definidos por 4.6, onde x é a quantidade de conhecimento do robô α_i. O conjunto, especificado por 4.7, informa as características físicas (que podem ser associados a recursos de hardware). A quantidade de características físicas do robô α_i está representado por y.

$$C(\alpha_i) = \{c_1, c_2, \ldots, c_x\} \tag{4.6}$$

$$F(\alpha_i) = \{f_1, f_2, \ldots, f_y\} \tag{4.7}$$

4.1.3 Especificações do Ambiente para execução da missão

Conforme mostrado anteriormente, o ambiente multirrobô que atua no Problema de Execução Cooperativa de Tarefas (CETP), está definido através de algumas premissas, que são necessárias à definição do contexto (ver Figura 4.4):

- Seja $A = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ o conjunto de robôs do ambiente, onde m é a quantidade de robôs.
- Seja a missão $M = T$, onde $T = \{\tau_1, \tau_2, \ldots, \tau_n\}$ é o conjunto de tarefas a serem solucionadas no ambiente, tal que n é a quantidade de tarefas.
- Seja $C(\alpha_i) = \{c_1, c_2, \ldots, c_x\}$ o conjunto de conhecimento do robô α_i, tal que $x \in \mathbb{N}$. c_x representa um conhecimento específico.
- Seja $F(\alpha_i) = \{f_1, f_2, \ldots, f_y\}$ o conjunto de característica física do robô α_i, tal que $y \in \mathbb{N}$. f_y representa uma característica física específica.
- Seja $C(\tau_j) = \{c_1, c_2, \ldots, c_x\}$ o conjunto de conhecimento necessário para se executar a tarefa τ_j, tal que $x \in \mathbb{N}$;
- Seja $F(\tau_j) = \{f_1, f_2, \ldots, f_y\}$ o conjunto de capacidade física necessário para que um robô execute a tarefa τ_j, tal que $y \in \mathbb{N}$.
- Φ é o contexto social, sendo igual para todos os robôs do ambiente.

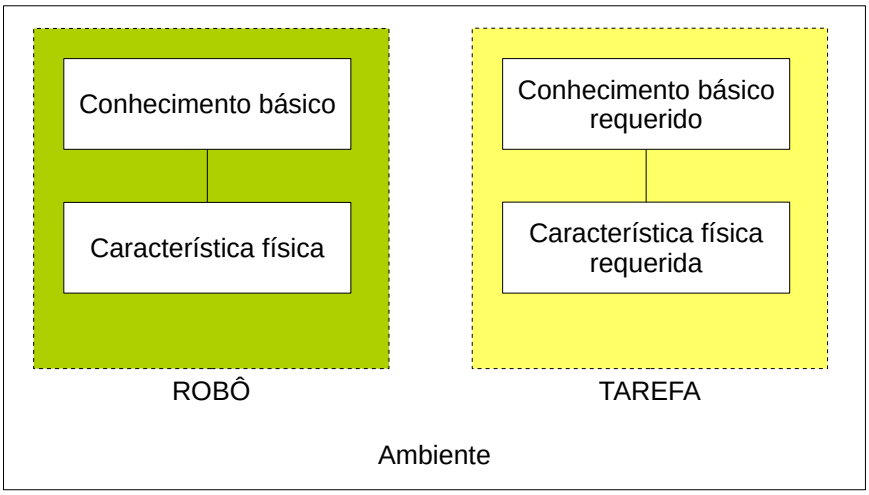

Figura 4.4: Definições do Ambiente (robôs e tarefas) para o Problema de Execução Cooperativa de Tarefas.

4.2 Especificações das Abordagens Sociais

Esta Seção apresenta o IDeM-MRS como um modelo teórico e formal de desenvolvimento intelectual para sistemas multirrobôs, aplicado ao CETP (Problema de Execução Cooperativa de Tarefas). Estão elucidados todos os processos e as definições utilizados para a concepção do formalismo. Aqui estão dispostos os módulos funcionais do modelo proposto, as regras que definem a metodologia do formalismo, bem como os algoritmos utilizados para a implementação do modelo.

A Figura 4.5 mostra as definições construídas através das abordagens sociais de aprendizagem analisadas (maiores detalhes dessas construções estão no Capítulo 3).

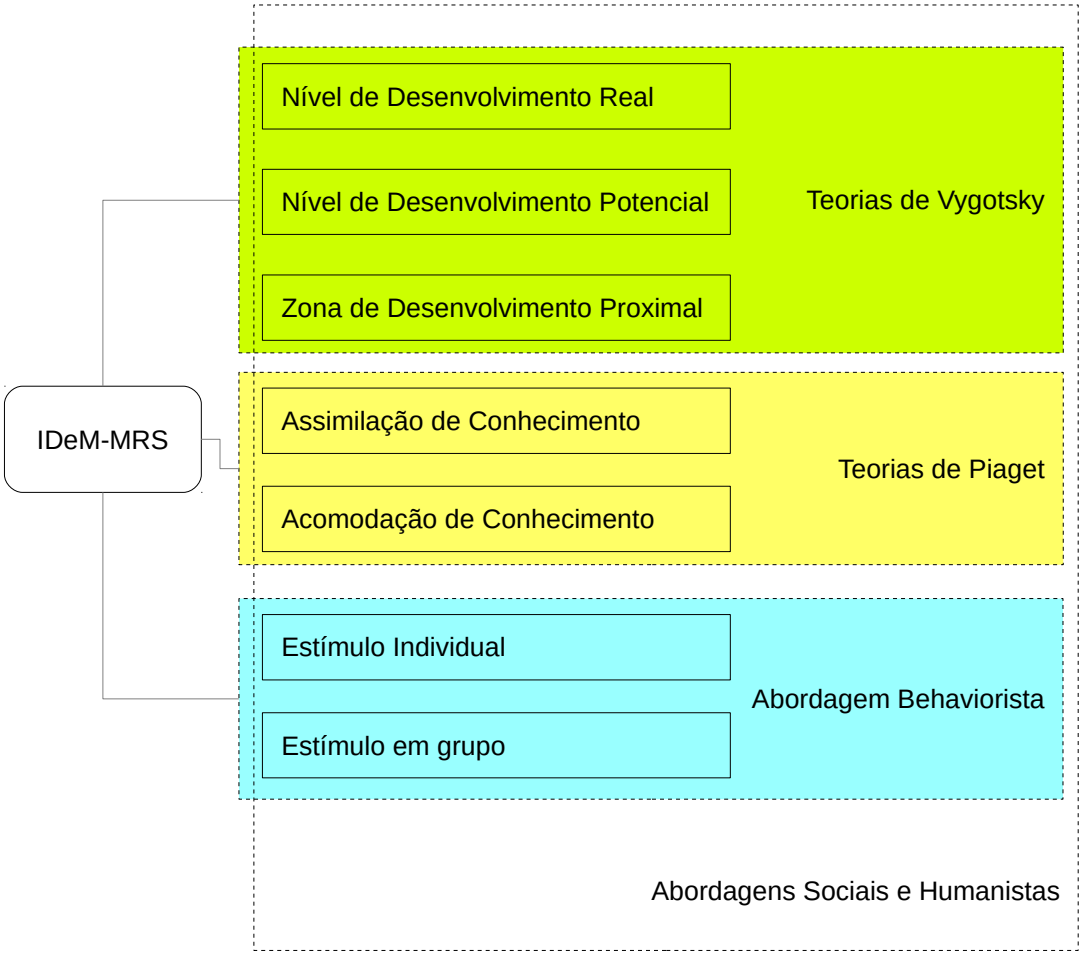

Figura 4.5: Definições das Abordagens Sociais para o Problema de Execução Cooperativa de Tarefas.

4.2.1 Zona de Desenvolvimento Proximal

A Zona de Desenvolvimento Proximal (*Zone of Proximal Development* - ZPD) relata se um dado robô α_i sabe, ou não, executar uma determinada tarefa τ_j, além de informar se ele é capaz, ou não, de aprendê-la. A ZPD é calculada através dos níveis Real e Potencial do robô em relação a uma determinada tarefa, conforme Equação (3.4), ou seja,

$$f\_zpd(\alpha_i, \tau_j) = f\_lpd(\alpha_i, \tau_j) - f\_lrd(\alpha_i, \tau_j).$$

Com isso, a definição da ZPD dos robôs permite uma análise através dos seguintes pontos:

- **O robô não possui o conhecimento nem a capacidade física necessários para resolver a tarefa.**
  Se $f\_zpd(\alpha_i, \tau_j) = -1$, então $f\_lpd(\alpha_i, \tau_j) = -1$ e $f\_lrd(\alpha_i, \tau_j) = 0$.

  Ou seja, o robô α_i não possui o conhecimento nem a capacidade física necessários para resolver a tarefa τ_j. Com isso, ele não sabe e não é capaz de aprender a execução de τ_j.

- **O robô possui o conhecimento e a capacidade física necessários para a execução da tarefa, mas não é capaz de aprimorar sua execução.**
  Se $f\_zpd(\alpha_i, \tau_j) = 0$, então $f\_lpd(\alpha_i, \tau_j) = 1$ e $f\_lrd(\alpha_i, \tau_j) = 1$.

  Ou seja, o robô α_i possui todo conhecimento e capacidade física acerca da execução da tarefa τ_j, mas não possui a capacidade para aprimorar sua execução.

- **O robô não possui todo conhecimento necessário para executar a tarefa, mas tem a capacidade física e é capaz de aprender a execução.**
  Se $f\_zpd(\alpha_i, \tau_j) = 1$, então $f\_lpd(\alpha_i, \tau_j) = 1$ e $f\_lrd(\alpha_i, \tau_j) = 0$.

  Ou seja, o robô α_i não possui todo conhecimento necessário para executar a tarefa τ_j, mas tem a capacidade física para tal. Nesse caso, α_i não sabe executar τ_j, mas é capaz de aprender.

- **O robô possui o conhecimento e capacidade física necessários para executar a tarefa, e ainda é capaz de aprimorar a execução.**
  Se $f\_zpd(\alpha_i, \tau_j) = -2$, então $f\_lpd(\alpha_i, \tau_j) = -1$ e $f\_lrd(\alpha_i, \tau_j) = 1$.

  Ou seja, o robô α_i possui o conhecimento e a capacidade física necessários para a execução da tarefa τ_j. Nesse caso, α_i já sabe executar a tarefa e ainda é capaz de aprimorar sua execução.

Definição (*Zona de Desenvolvimento Proximal): a Zona de Desenvolvimento Proximal de um robô informa se ele tem potencial de aprendizagem em relação a uma tarefa específica.*

4.2.2 Nível de Desenvolvimento Real

O Nível de Desenvolvimento Real (*Level of Real Development* - LRD) de um robô α_i em relação a uma tarefa τ_j define se α_i é capaz de resolver τ_j sozinho. Esse nível, que está especificado pela Equação (3.2), permite uma análise através dos seguintes pontos:

- **O robô não possui o conhecimento necessário para resolver a tarefa e/ou não possui capacidade física.**
  Se $f\_lrd(\alpha_i, \tau_j) = 0$, então α_i ou não possui todo conhecimento necessário para resolver τ_j sozinho, e/ou não possui capacidade física para tal.

- **O robô possui o conhecimento e a capacidade física necessários para resolver a tarefa.**
  Se $f\_lrd(\alpha_i, \tau_j) = 1$, então o robô α_i possui todo conhecimento e capacidade física necessários para resolver a tarefa τ_j sozinho.

Definição (*Nível de Desenvolvimento Real): o Nível de Desenvolvimento Real de um robô informa se ele possui conhecimento e capacidade física para a execução de uma determinada tarefa de forma autônoma.*

4.2.3 Nível de Desenvolvimento Potencial

O Nível de Desenvolvimento Potencial (*Level of Potential Development* - LPD) de um robô α_i em relação a uma tarefa τ_j é especificado por $f\_lpd(\alpha_i, \tau_j)$, disposto na Equação (3.3). O LPD dos robôs define a capacidade de aprendizagem desses agentes, ou seja, se α_i tem capacidade para aprender τ_j ou não.

É possível perceber que, caso o robô α_i saiba resolver a tarefa τ_j, ele ainda é capaz de aprimorá-la, uma vez que essa aprendizagem já foi feita. Com isso, percebe-se que o nível de desenvolvimento real está contido no potencial. Sendo assim, alguns pontos são importantes:

- **O robô tem capacidade para aprender a tarefa.**
  Se $f\_lpd(\alpha_i, \tau_j) = -1$, então o robô α_i tem capacidade para aprender a tarefa e ainda é capaz de aprimorar sua execução.

- **O robô não tem capacidade para aprender a tarefa.**
  Se $f\_lpd(\alpha_i, \tau_j) = 1$, então α_i não é capaz de aprender τ_j.

Definição (*Nível de desenvolvimento potencial): o nível de desenvolvimento potencial de um robô informa se ele possui capacidade de aprendizagem e execução de uma tarefa específica.*

4.2.4 Assimilação de Conhecimento

A Assimilação é observada quando o robô adquire um conhecimento novo acerca de uma determinada tarefa, desconhecida por ele até o momento. Essa aquisição é verificada em $f\_assimilation(\alpha_i, \alpha_x, \tau_j)$, Equação (3.6), quando o robô α_i recebe um conhecimento novo do robô α_x, acerca da execução da tarefa τ_j.

Sabendo que c' é um conhecimento que o robô α_i não possui, ou seja, $c' \notin C(\alpha_i)$; c' é necessário para execução da tarefa τ_j; e que o robô α_x possui c', a operação de Assimilação do conhecimento c' por α_i é $C(\alpha_i) = C(\alpha_i) \cup \{c'\}$. Sendo assim, é possível descrever os pontos importantes:

- **O robô realizou a Assimilação de Conhecimento com sucesso.**
  Se $f\_assimilation(\alpha_i, \alpha_x, \tau_j) = 1$, então o robô α_i assimilou conhecimento do robô α_x acerca da execução da tarefa τ_j.

- **O robô não pôde realizar a Assimilação de Conhecimento.**
  Se $f\_assimilation(\alpha_i, \alpha_x, \tau_j) = 0$, então o robô α_i não foi capaz de assimilar conhecimento do robô α_x acerca da tarefa τ_j.

Definição (*Assimilação): para haver inclusão do conhecimento em um robô, ele precisa não saber resolver a tarefa em questão e deve ser capaz de aprendê-la.*

4.2.5 Acomodação de Conhecimento

A Acomodação é caracterizada quando um robô atualiza um conhecimento acerca da execução de uma tarefa específica. O robô pussui um conhecimento e recebe outro, mais atual ou mais eficiente. Isso pode ocorrer quando há necessidade de melhoria na execução.

Sabendo que c' é o conhecimento novo e c é um conhecimento já instalado na base de dados do robô α_i, esse aperfeiçoamento é constatado com a função exposta na Equação (3.8), $f\_accommodation(\alpha_i, c, c')$. Nesse caso, o robô α_i atualiza seu conhecimento antigo c pelo novo c'. Essa operação é possível somente se $C(\alpha_i) = C(\alpha_i) - \{c\} \cup \{c'\}$. Sendo assim, é possível descrever os pontos importantes:

- **O robô realizou a Acomodação de Conhecimento com sucesso.**
  Se $f\_accommodation(\alpha_i, c, c') = 1$, então o robô α_i acomodou o conhecimento c', em substituição do c.

- **O robô não pôde realizar a Acomodação de Conhecimento.**
  Se $f\_accommodation(\alpha_i, c, c') = 0$, então o robô α_i não foi capaz de acomodar c'.

Definição (*Acomodação): para haver a alteração do conhecimento em um robô, ele precisa ter o conhecimento para resolver a tarefa e deve ter a possibilidade de aprimorar esse conhecimento.*

4.2.6 Estímulo individual

Quando requisitado para uma tarefa, o robô pode informar seu estímulo individual, o que informa se ele detém conhecimento acerca da resolução. A Equação (3.10) formaliza essa informação, afirmando se o robô α_i possui ou não todo conhecimento necessário para executar a tarefa τ_j, através da análise de $f\_\varepsilon(\tau_j, \alpha_i)$:

- **O robô sabe executar a tarefa.**
  Se $f\_\varepsilon(\tau_j, \alpha_i) = 1$, então o robô α_i sabe executar a tarefa τ_j.
  Sendo assim, quando a tarefa τ_j está no LRD do robô α_i, ou seja, $f\_lrd(\alpha_i, \tau_j) = 1$, então impreterivelmente ocorre $f\_\varepsilon(\tau_j, \alpha_i) = 1$, já que τ_j faz parte do nível de desenvolvimento real de α_i. Nesse caso, $f\_lrd(\alpha_i, \tau_j) = 1 \Rightarrow f\_\varepsilon(\tau_j, \alpha_i) = 1$.

- **O robô não sabe executar a tarefa.**
  Se $f\_\varepsilon(\tau_j, \alpha_i) = 0$, então α_i não possui o conhecimento para executar τ_j.
  Sendo assim, impreterivelmente ocorre $f\_lrd(\alpha_i, \tau_j) = 0$, já que τ_j não está no nível de desenvolvimento real de α_i. Nesse caso, $f\_\varepsilon(\tau_j, \alpha_i) = 0 \Rightarrow f\_lrd(\alpha_i, \tau_j) = 0$.

Definição (*Estímulo individual): um robô, quando estimulado por uma tarefa, informa se ele possui todo conhecimento acerca de sua execução.*

4.2.7 Estímulo em grupo

Em se tratando de um grupo de robôs, a função estímulo em grupo $f\_\varepsilon_{gr}(\tau_j, A')$, exposta na Equação (3.11), informa se um grupo de robôs A' detém todo conhecimento necessário para a execução de uma tarefa específica τ_j.

Para isso, é feita a união de todos conhecimentos e capacidades físicas de cada robô do grupo, como se eles fossem um único robô com todos esses itens, possibilitando a cooperação de seus recursos para resolver uma mesma tarefa. Essa pode ser uma ideia de um "super-robô", permitindo que o grupo de robôs coopere por recursos físicos (de *hardware*), compartilhando equipamentos. Sendo assim, algumas questões são importantes:

- **Há um grupo robôs capaz de executar a tarefa.**
  Se $f\_\varepsilon_{gr}(\tau_j, A') = 1$, o grupo de robôs A' é capaz de executar a tarefa τ_j, desde que:
  - $C(\tau_j) \subseteq \{\bigcup_{\alpha_i \in A'} C(\alpha_i)\}$;
  - $F(\tau_j) \subseteq \{\bigcup_{\alpha_i \in A'} F(\alpha_i)\}$.
- **Não há um grupo robôs capaz de executar a tarefa.**
  Se $f\_\varepsilon_{gr}(\tau_j, A') = 0$, então o grupo de robôs A' não é capaz de executar a tarefa τ_j.

Definição (*Estímulo em grupo): um grupo de robôs, quando estimulado por uma tarefa específica, informa se esse grupo possui todo conhecimento necessário para sua execução.*

4.3 Especificações dos estados dos robôs

O Modelo de Desenvolvimento Intelectual se adapta a qualquer ambiente multirrobô. Idealmente, esse ambiente pode assumir qualquer configuração desde que contenha um conjunto de robôs sem interferências externas e sem a especificação de uma alternativa para o mecanismo de coordenação. Sendo assim, é imprescindível que não haja qualquer robô com característica de liderança ou centralização de qualquer permissão diferente dos demais robôs do grupo.

Inicialmente cada robô, no momento de solicitação de execução de uma tarefa, está ativo em algum estado. Esse estado o caracteriza diante do grupo, podendo assumir certos valores que o diferencia em termos de aptidão, execução e instrução. Os estados estão definidos como: “Apto”, “Inapto”, “Instrutor Forte”, “Instrutor Fraco”, “Executor Forte” e “Executor Fraco”.

Para uma melhor visualização das diferenças dos robôs que pertencem, em um determinado momento, a cada um desses estados, a Tabela 4.1 trata da existência dos conceitos relacionados à execução de uma tarefa, que são conhecimento prévio (total ou parcial) e capacidade física de execução de uma tarefa:

- "Saber total"informa se o robô possui todo o conhecimento acerca da tarefa;
- "Saber parcial"especifica se o robô possui algum conhecimento, mas não todo;
- "Ser capaz"indica se o robô tem a capacidade física requerida para a tarefa.

Os estados dos robôs, bem como suas definições, estão descritos nas próximas seções, sabendo que:

- o robô α_i possui um conjunto de conhecimentos básicos e outro de características físicas: $\alpha_i = \{C(\alpha_i), F(\alpha_i)\}$;
- a tarefa τ_j possui um conjunto de conhecimentos e outro de capacidades físicas, que são requeridos para sua execução: $\tau_j = \{C(\tau_j), F(\tau_j)\}$;
- o robô α_i está sendo requisitado para a resolução da tarefa τ_j.

Estado	Saber total	Saber parcial	Ser capaz
Apto	Sim	Não	Sim
Executor Forte	Sim	Não	Não
Executor Fraco	Não	Sim	Não
Instrutor Forte	Não	Sim	Sim
Instrutor Fraco	Não	Não	Sim
Inapto	Não	Não	Não

Tabela 4.1: Características dos estados dos robôs em relação a uma tarefa.

4.3.1 Definições dos estados

Considerando o conhecimento que cada robô possui acerca da execução de tarefas do ambiente, bem como suas características físicas (características de *hardware*), o robô pode se encontrar nos seguintes estados:

- *Apto:* O robô que detém todos os conhecimentos e capacidades físicas necessários à execução da tarefa.
- *Executor Forte:* O robô que detém todos os conhecimentos da execução da tarefa, porém não possui a capacidade física requerida para executá-la.
- *Executor Fraco:* O robô que detém algum conhecimento sobre a execução da tarefa, mas não todos, porém sem nenhuma capacidade física para executá-la.
- *Instrutor Forte* O robô que possui toda capacidade física necessária para a execução da tarefa, e ainda algum conhecimento, mas não todos.
- *Instrutor Fraco* O robô que possui toda capacidade física necessária para a execução, porém com nenhum conhecimento.
- *Inapto* O robô que não detém conhecimento nem capacidade física para a execução da tarefa.

Sendo assim, as definições formais para esses estados são:

Definição (*Apto) o robô α_i diz-se Apto a resolver a tarefa τ_j se, e somente se, as condições* $C(\tau_j) \subset C(\alpha_i)$ *e* $F(\tau_j) \subset F(\alpha_i)$ *são satisfeitas.*

Definição (*Executor Forte) o robô α_i diz-se Executor Forte da tarefa τ_j se, e somente se,* $C(\tau_j) \backslash C(\alpha_i) = X$*, tal que* $X \neq \emptyset$ *e* $X \neq C(\tau_j)$*, e* $F(\tau_j) \subset F(\alpha_i)$*.

Definição (*Executor Fraco) o robô α_i diz-se Executor Fraco da tarefa τ_j se, e somente se,* $C(\tau_j) \subset C(\alpha_i) \backslash C(\tau_j) = C(\tau_j)$*, ou de outra forma,* $C(\tau_j) \cap C(\alpha_i) = \emptyset$*, e* $F(\tau_j) \subset F(\alpha_i)$*.

Definição (*Instrutor Forte) o robô α_i diz-se Instrutor Forte da tarefa τ_j se, e somente se,* $C(\tau_j) \subset C(\alpha_i)$ *e* $F(\tau_j) \not\subset F(\alpha_i)$*.

Definição (*Instrutor Fraco) o robô α_i diz-se Instrutor Fraco da tarefa τ_j se, e somente se,* $C(\tau_j) \backslash C(\alpha_i) = X$*, tal que* $X \neq \emptyset$ *e* $X \neq C(\tau_j)$*, e* $F(\tau_j) \not\subset F(\alpha_i)$*.

Definição (*Inapto) o robô α_i diz-se Inapto em relação à tarefa τ_j se, e somente se,* $C(\tau_j) \backslash C(\alpha_i) = C(\tau_j)$*, ou de outra forma,* $C(\tau_j) \cap C(\alpha_i) = \emptyset$ *e* $F(\tau_j) \not\subset F(\alpha_i)$*.

4.3.2 A Máquina de Estados mROBOS

O robô pode evoluir para outro estado dependendo do fluxo de conhecimento que é repassado a ele em cada momento. A Figura 4.6 mostra a Máquina de Estados *mROBOS* que descreve as transições de estados a que o robô pode ser submetido, dependendo da necessidade do momento. Essa máquina possui um único estado inicial (estado A, de cor amarelo), quatro estados intermediários (B, C, D e F, de cor verde) e seis finais (E, G, H, I, J e L, de cor azul).

A cada transição, o robô possui um novo estado. Inicialmente, todos os robôs encontram-se no estado A (estado inicial). Após as avaliações de conhecimento parcial ou total e de capacidade física necessários, os robôs chegam aos estados finais D, E, H, I, J ou L. Portanto, a Máquina de Estados *mROBOS* possui o estado inicial A, um conjunto de estados finais $\{D,E,H,I,J,L\}$ e intermediários $\{B,C,F,G\}$. Além disso, o alfabeto (conjunto de símbolos) para as respostas às perguntas que estão nas transições é $\{sim,nao\}$.

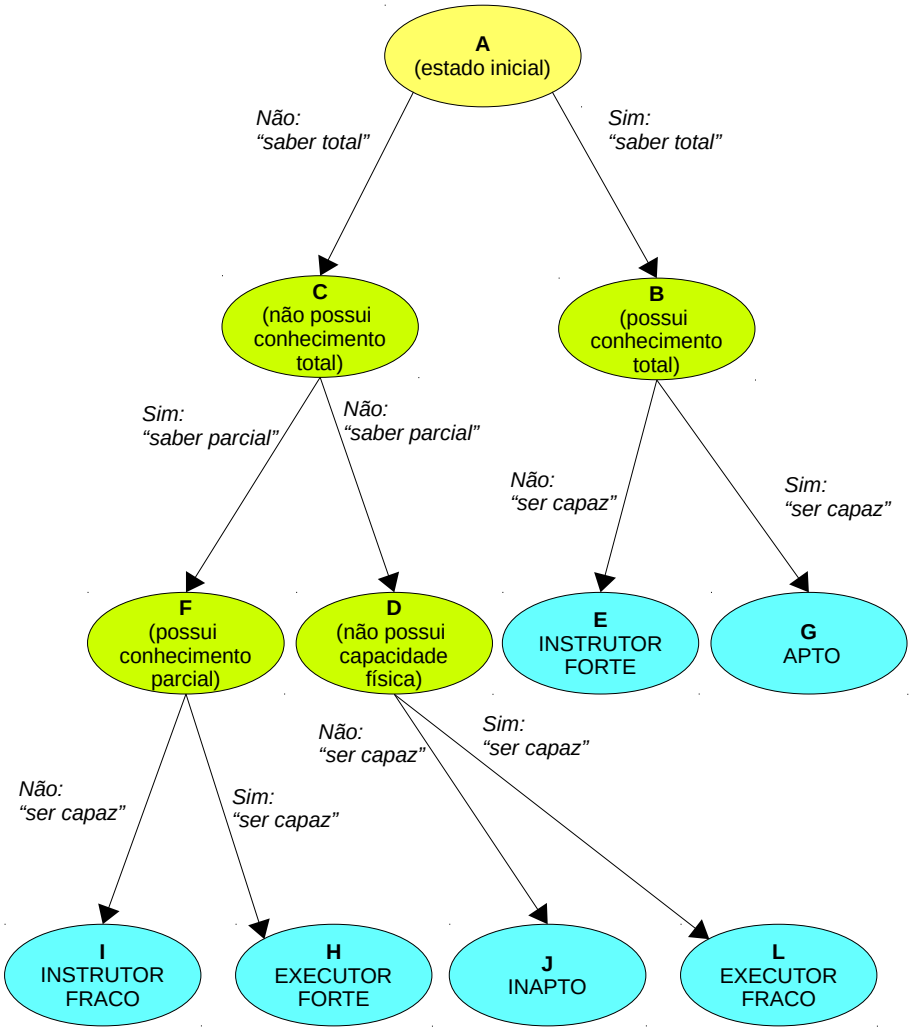

Figura 4.6: Fluxo de informação da Máquina de Estados *mROBOS*.

No momento em que o grupo de robôs é invocado para a resolução de uma tarefa da missão, cada robô é analisado para ser situado no estado correspondente. Todos robôs iniciam no estado A e são conduzidos gradualmente para os possíveis estados finais D, E, H, I, J ou L.

Se um dado robô α_i, acerca da execução de uma tarefa τ_j, possui algum conhecimento (mas não todo) e possui toda capacidade física, ele assumirá o estado H (executor). Ele percorrerá o fluxo dos estados $A-C-F-H$, respondendo *"Não"*, *"Sim"* e *"Sim"* para as perguntas *"Saber total?"*, *"Saber parcial?"* e *"Ser capaz?"*. Já um robô α_x, acerca da execução da mesma tarefa, possui todo conhecimento e toda capacidade física, ele assumirá o estado G (apto). Ele passará pelos estados $A-B-G$, respondendo *"Sim"* e *"Sim"* para as perguntas *"Saber total?"* e *"Ser capaz?"*.

A implementação da Máquina de Estados *mROBOS* está descrita no Algoritmo 1, que recebe como parâmetros de entrada: a tarefa corrente τ_j e o grupo R de robôs.

Algorithm 1: Obter estados de Robôs

Input: Tarefa τ_j de M e um grupo de robôs R
Output: A definição dos conjunto de estados dos robôs R

```
1:  begin
2:      Aptos = ∅;
3:      Instrutor_forte = ∅;
4:      Instrutor_fraco = ∅;
5:      Executor_forte = ∅;
6:      Executor_fraco = ∅;
7:      Inaptos = ∅;
8:      robots = R;
9:      if f_ε_gr(R, τ_j) ≠ ∅ then
10:         return Tarefa_impossvel;
11:     else
12:         while robos ≠ ∅ do
13:             r = Selecionar_robo(robos);
14:             if f_lrd(r, τ_j) == 1 then
15:                 Aptos = Aptos ∪ {r};
16:             else
17:                 if f_lpd(r, τ_j) == 1 then
18:                     if f_ε_r(τ_j, r) == 0 then
19:                         Executor_fraco = Executor_fraco ∪ {r};
20:                     else
21:                         Executor_forte = Executor_forte ∪ {r};
22:                 else
23:                     if f_ε_r(τ_j, r) == 0 then
24:                         Inapto = Inapto ∪ {r};
25:                     else
26:                         if f_ε_c(τ_j, r) ⊃ C(τ_j) then
27:                             Instrutor_forte = Instrutor_forte ∪ {r};
28:                         else
29:                             Instrutor_fraco = Instrutor_fraco ∪ {r};
30:             robos = robos − {r};
31:     return 1;
```

Inicialmente todos os estados estão sem robôs (linhas 2 a 7) e o grupo de robôs é alocado em *robos* (linha 8). A partir desse ponto, seguem as seguintes tomadas de decisão descritas a seguir.

- A condição da linha 9 verifica se todos os robôs R são capazes de gerar, em grupo, o conhecimento necessário para a resolução da tarefa corrente τ_j. Se sim, o procedimento de enquadrá-los em algum estado segue adiante. Caso contrário, o procedimento é finalizado com a messagem *"Tarefa impossível"*. Essa função informa o estímulo em grupo, através da Equação (3.11), e envolve conhecimento das abordagens de aprendizagem Social e Behaviorista.

- A função *Selecionar_robo* (linha 13) obtém um único robô do conjunto especificado como parâmetro. *robos* é decrescido a cada iteração até que não haja mais robô sem verificação. A seleção é feita randomicamente. Alguma heurística para melhoria dessa escolha pode ser implementada nessa rotina.

- O robô r é incluído no estado $Apto$ (linha 15), se ele possui conhecimento e capacidade física necessários para executar autonomamente a tarefa τ_j. A verificação se dá através da condição $f\_lrd(r,\tau_j) == 1$, na linha 14.

- A linha 18 verifica o nível de conhecimento do robô r acerca da tarefa τ_j, já que foi identificado que ele não detém o conhecimento total ($f\_lrd(r,\tau_j) == 0$, na linha 14), e que possui a capacidade física para a execução ($f\_lpd(r,\tau_j) == 1$, na linha 17). Dessa forma, se ele não possui conhecimento ($f\_\varepsilon_r(\tau_j,r) == 1$, linha 18), ele é um $Executor\_Fraco$ (linha 19). Caso contrário, ele é um $Executor\_Forte$ (linha 21). As funções f_lrd e f_lpd informam os níveis de desenvolvimento Real e Potencial, respectivamente, do robô em relação à tarefa, e utilizam a teoria de Lev Vygotsky com as equações (3.2) e (3.3).

- O robô r é incluído no estado $Inapto$ se ele não possui característica física para executar a tarefa τ_j (condição $f\_lpd(r,\tau_j) == 0$, na linha 17) e não possui qualquer tipo de informação acerca da execução ($f\_lpd(r,\tau_j) == 0$, na linha 18). No entanto, se r não possui característica física mas possui todo conhecimento necessário ($f\_\varepsilon_c(\tau_j,r) \supset C(\tau_j)$, linha 26), ele é um $Instrutor\_Forte$ (linha 27). Porém, com algum conhecimento (não total), ele é um $Instrutor\_Fraco$ (linha 29). A função f_ε_c informa o conhecimento obtido por estímulo da tarefa e foi extraída da abordagem Behaviorista de John Watson, através da Equação (3.9).

- Após enquadrar os robôs em seus devidos estados, decide-se qual robô deverá executar a tarefa, dentre os que estiverem nos estados G ($Apto$), H ($Executor\_Forte$) e L ($Executor\_Fraco$), seguindo essa ordem de prioridade de escolha. Caso não exista robô no estado G, o ambiente busca algum do H, e depois do L. Essa escolha faz com que o robô eleito passe para um outro estado chamado $Executor$. No entanto, essa transição não se dá de forma imediata. Pode ser necessário um repasse de conhecimento entre os robôs.

4.3.3 A Máquina de Estados mExecutor

O fluxo da transformação dos demais estados do robô para o *Executor* está informado na Máquina do Estado *mExecutor* da Figura 4.7. Essa máquina possui três possíveis estados iniciais (estados *A*, *B* e *C*, marcados de amarelo), três estados intermediários (*D*, *E* e *F*, de cor verde) e um estado final (*G*, de cor azul).

Da mesma forma que na Máquina de Estados *mROBOS*, a cada transição, o robô possui um novo estado. Inicialmente, os robôs podem pertencer ao estado *A* (*Instrutor Forte*), *B* ((*Instrutor Fraco*) ou *C* (*Apto em execução*). Após as avaliações de conhecimento parcial ou total, os robôs chegam aos estados *D* (*Executor Forte*), *E* (*Executor Fraco*) ou *F* (*Apto*). Quando um robô está em qualquer um desses estados intermediários, ele pode automaticamente ser transferido para o estado *G* (*Executor*).

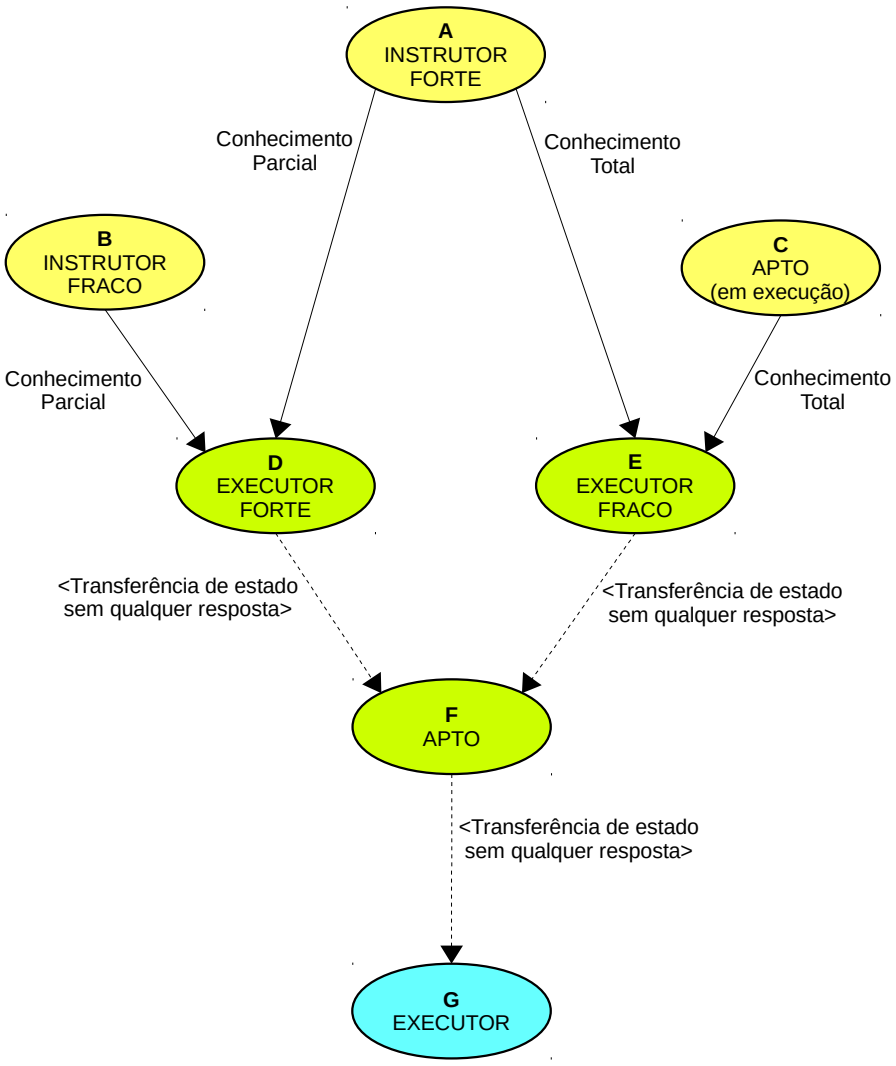

Figura 4.7: Fluxo de informação da Máquina de Estados *mExecutor*.

Dito de outra forma, os estados *Instrutor Fraco*, *Instrutor Forte* e *Apto* (em execução) são os que possuem robôs capazes de enviar conhecimento a um outro robô, que futuramente será eleito *Executor*.

Nesse caso, os estados intermediários, *Executor Forte*, *Executor Fraco* e *Apto*, poderão ser escolhidos para ocuparem o estado *Executor*. Para ser definido como tal, o robô eleito precisa pertencer a uma das três condições abaixo:

1. **Robô locado no estado *Apto* e ser ocioso.**
   - Dessa forma ele não precisa de repasse de conhecimento de algum outro robô do ambiente.
   - A transição para o estado *Executor* se dá de forma direta.
2. **Robô locado em *Executor Forte*.**
   - Nesse caso, ele deve receber algum conhecimento de outros robôs do ambiente, que estejam locados nos estados *Instrutor Fraco* ou *Instrutor Forte*, de modo que seja possível obter todo conhecimento necessário para execução da tarefa específica.
   - O conhecimento é adquirido por Assimilação, podendo também haver a Acomodação.
3. **Robô locado no estado *Executor Fraco*.**
   - O robô, então, deverá adquirir todo conhecimento acerca da tarefa de outros robôs do ambiente, que estejam em *Instrutor Forte* ou *Apto em execução*.
   - O conhecimento é adquirido apenas por Assimilação.

Nesses dois últimos casos, após o robô *Executor Forte* (ou ainda *Executor Fraco*) ter assimilado o devido conhecimento (parcial ou total, respectivamente), ele pode se transformar em robô *Apto* disponível para a resolução da referida tarefa. Com isso, ele assume o estado *Executor* e pode executar a tarefa em questão.

Um fato importante é uma situação em que uma tarefa requer algumas características físicas que nenhum dos robôs do grupo possui. Nesse caso, essa tarefa é chamada de *impossível*, já que nenhum robô pode executá-la.

Definição (*Tarefa Impossível) uma tarefa τ_j diz-se Impossível se, e somente se, a condição $F(\tau_j) \not\subset \bigcup_{i=1}^{m} F(\alpha_i)$ for satisfeita, considerando os robôs $A = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ em resolução de tarefas, tal que $\alpha_i = \{C(\alpha_i), F(\alpha_i)\}$ e m é a quantidade de robôs do grupo.*

4.4 Regras de cooperação para o IDeM-MRS

O método de solução para o Problema de Execução Cooperativa de Tarefas (CETP) proposto por esta pesquisa e implementada no IDeM-MRS (*Intellectual Development Model for Multi-Robot Systems*), define regras de cooperação, que permitem que os robôs cooperem para cumprir diferentes partes da missão, finalizando em sua execução completa. IDeM-MRS combina rotinas de atualização do sistema multirrobô, aplicação de regras de cooperação, alocação de tarefas em tempo real e, por fim, execução das tarefas da missão. Estes componentes estão integrados visando fornecer soluções eficientes para sistemas multirrobôs na resolução de CTEP, associando o benefício da aquisição de conhecimento por parte dos robôs.

Os módulos funcionais que constituem esses componentes, bem como o fluxo da informação entre eles, estão descritos na Figura 4.8. Essa modelagem foi adaptada do *framework* proposto por Talay et al. (2011) para a inclusão dos elementos específicos do IDeM-MRS.

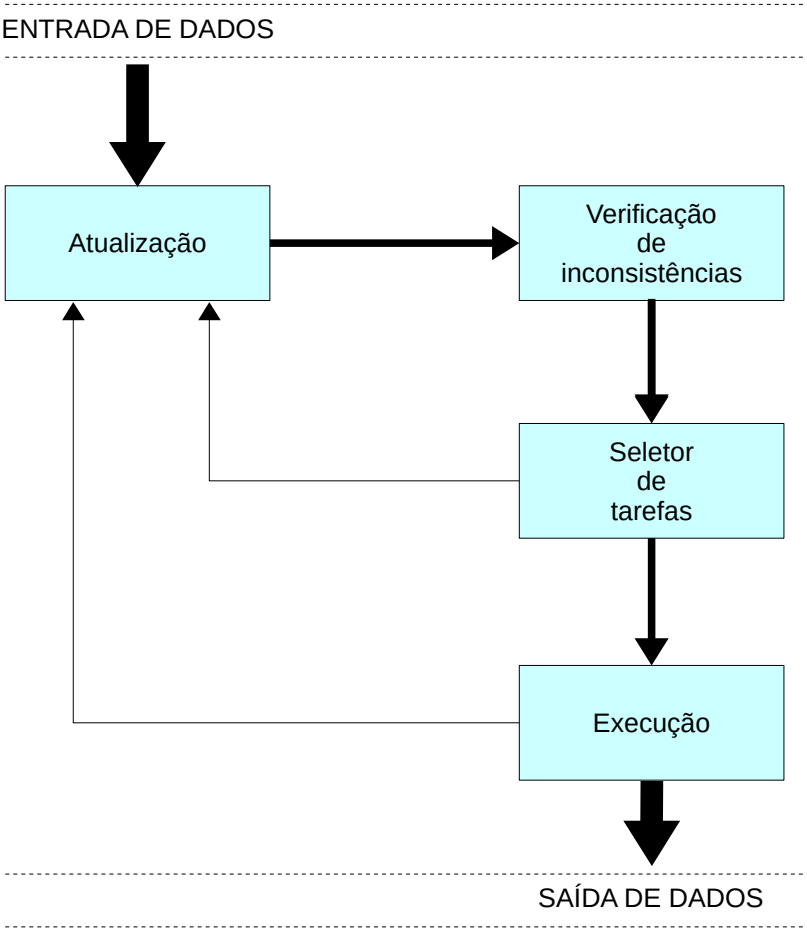

Figura 4.8: Módulos funcionais do IDeM-MRS.

Esse *framework* combina módulos funcionais que atualizam as instâncias do ambiente multirrobô, verificam a consistência dos dados, aplicam as regras de convivência para selecionar as tarefas e alocá-las aos robôs, e finalmente, executam a missão. IDeM-MRS é projetado com capacidade para lidar com situações de resolução de tarefas em tempo real. Sendo assim, essa modelagem pode demonstrar, com eficiência, cada interação entre os módulos funcionais, bem como o fluxo de informação que percorre no sistema.

O objetivo de um time de robôs $A = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$, tal que m é a quantidade de robôs do ambiente, é executar uma lista ordenada de tarefas, denominada missão $M = \{\tau_1, \tau_2, \ldots, \tau_n\}$, tal que n é a quantidade de tarefas da missão, sempre procurando maximizar o conjunto de conhecimento dos robôs $C(\alpha_i)$, tal que $0 < i \leq m$, e minimizar o tempo de execução total de M.

- **Atualização**:
  - É responsável por configurar o sistema multirrobô (MRS) e informar o objetivo global (a missão).
  - As informações acerca do MRS podem ser obtidas com algum agente externo (usuário), com algum robô do ambiente ou com alterações realizadas pelos módulos funcionais *Seletor de Tarefas* e *Execução*.
- **Verificação de Inconsistência**:
  - É responsável por iniciar procedimentos de advertência caso necessite informar inconsistência de algum dado do ambiente.
  - Pode ser considerado como uma inconsistência, por exemplo, algum robô que não possui dois conjuntos, o de conhecimento e o de característica física. Muito embora, seja totalmente possível ter nenhum elemento nesses dois conjuntos.
  - Em uma situação em que todos os dados do ambiente são consistentes, esse módulo informa ao *Seletor de Tarefas* que pode iniciar sua execução.
- **Seletor de Tarefas**:
  - É responsável por executar as regras de cooperação, obtidas por avaliações de modelos sociais de aprendizagem (LSM).
  - Ele aloca as tarefas da missão aos robôs mais adequados.
  - Nesse módulo, portanto, estão implementadas as regras do IDeM-MRS.
- **Execução**:
  - Implementa a execução da missão sincronizada e cooperativamente, conforme as listas de alocações estabelecidas pelo *Seletor de Tarefas*.

Um exemplo de fluxo das informações para um sistema multirrobô na resolução de um CTEP, quando exposto ao IDeM-MRS, está retratado através da ordem numérica (ver Figura 4.9):

1. Inicialmente, o módulo funcional *Atualização* recebe a especificação do sistema multirrobô em termos de definição do ambiente, dos robôs e da missão.

2. A especificação do ambiente é enviada para o módulo *Verificação de Inconsistência*, que a avalia procurando detectar dados inconsistentes que podem oferecer riscos à eficácia do IDeM-MRS.

3. Caso não haja dados inconsistentes, o módulo *Seletor de Tarefas* realiza, segundo as regras definidas pelo IDeM-MRS, a seleção do robô ideal para resolver cada tarefa candidata. Nesse ponto os conflitos são solucionados e as alterações do ambiente (conhecimento dos robôs, tarefas resolvidas e por qual robô) são efetivadas e enviadas para *Atualização* do sistema.

4. O módulo *Execução* finaliza a missão obedecendo a seleção robô/tarefa definida pelo IDeM-MRS no módulo *Seletor de Tarefas*. Ele informa uma lista de execução de toda missão, além de enviar atualizações com a confirmação de que a missão foi executada completamente.

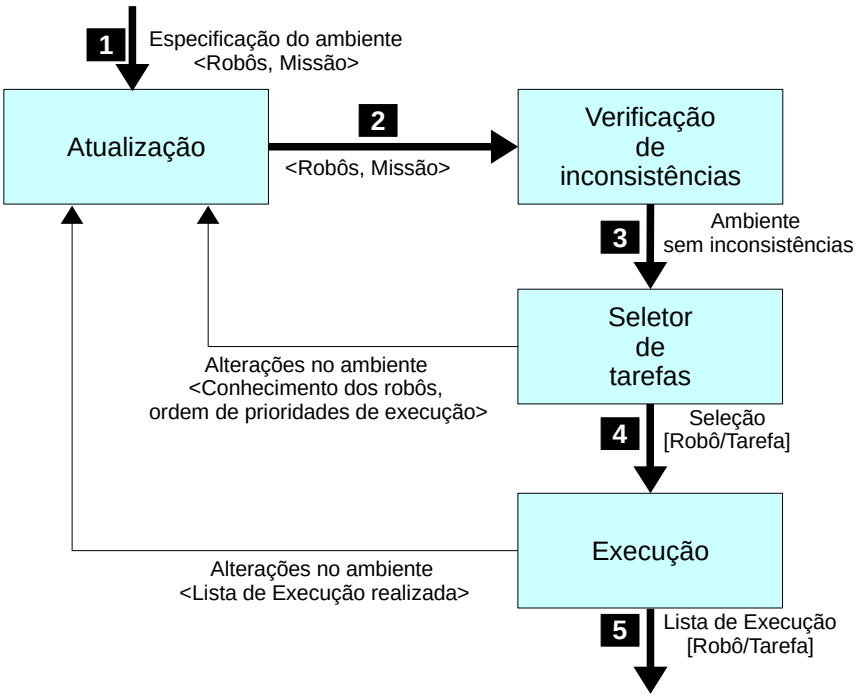

Figura 4.9: Fluxo de Informação dos Módulos funcionais do IDeM-MRS.

4.4.1 Representação do ambiente: robôs e tarefas

A missão em IDeM-MRS é representada por uma lista ordenada de tarefas, ou seja, $M = \{\tau_1, \tau_2, \ldots, \tau_n\}$. Essas tarefas são representadas com a informação necessária para sua execução pelos robôs e para a percepção de seu status pelo ambiente, conforme a 4-tupla $< id_r, reqc_r, recf_r, info_r >$, onde:

- id_r é um valor único de identificação da tarefa;
- $reqc_r$ é o conjunto de conhecimentos requeridos para executar a tarefa;
- $recf_r$ é o conjunto das características físicas requeridas para executar a tarefa;
- $info_r$ indica se a tarefa está em execução por algum robô ou em estado de espera.

O grupo de robôs não difere dessa representação. É representado como um conjunto de agentes robóticos que devem atuar no ambiente, ou seja $R = A = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$. Os robôs também são representadas como 4-tuplas $< id_t, reqc_t, recf_t, info_t >$, onde:

- id_t é um valor único de identificação do robô;
- $reqc_t$ é o conjunto de conhecimentos pertencente ao robô;
- $recf_t$ é o conjunto das características físicas do robô;
- $info_t$ indica se o robô está executando alguma tarefa (dedicado) ou não.

A Tabela 4.2 mostra um exemplo de representação de tarefas. A tarefa $T14$ está sendo executada por algum robô (já que $info_t = 1$), possui como conjunto de conhecimentos necessários a sua execução $C(T14) = \{c_4, c_6\}$ e características físicas requeridas $F(T14) = \{f_2, f_5, f_8\}$.

A Tabela 4.3 mostra um exemplo de representação de robôs. Com essa representação, o robô com identificação $R08$ está ocioso (já que $info_r = 0$), possui como conjunto de conhecimentos básicos $C(R08) = \{c_1, c_2, c_6\}$ e características físicas $F(R08) = \{f_1, f_2\}$.

id_t	$reqc_t$	$reqf_t$	$info_t$
T58	$\{c_5\}$	$\{f_1, f_2\}$	0
T14	$\{c_4, c_6\}$	$\{f_2, f_5, f_8\}$	1

Tabela 4.2: Exemplo de informações das tarefas, representadas no IDeM-MRS.

id_r	$reqc_r$	$reqf_r$	$info_r$
R08	$\{c_1, c_2, c_6\}$	$\{f_1, f_2\}$	0
R05	$\{c_5\}$	$\{f_2, f_5\}$	1

Tabela 4.3: Exemplo de informações dos robôs, representadas no IDeM-MRS.

4.4.2 Seletor de Tarefas

A configuração inicial do ambiente é dada pela lista de tarefas em sequencia (T) e um conjunto de robôs (R), sabendo que $T = \{\tau_1, \tau_2, \ldots, \tau_m\}$ forma a missão M, e que R representa os robôs ativos no ambiente, $R = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$. Tanto T quanto R possuem seus conjuntos de dados consistentes, uma vez que o módulo *Verificação de inconsistências* já analisou os dados e os enviou para o *Seletor de tarefas*. Portanto, sabe-se que tanto as tarefas quanto os robôs possuem conjuntos de conhecimentos e capacidades (ou características) físicas.

Cada tarefa é verificada em ordem conforme a missão é informada ao ambiente. Ou seja, a tarefa τ_i somente será atendida antes da τ_j se e somente se $i < j$, onde i e j são as ordens de precedência das tarefas. A seleção de tarefas é realizada de acordo com os seguintes passos:

1. Selecionar uma tarefa da missão. Essa seleção é feita de acordo com a ordem estabelecida na missão. Embora seja possível definir quais tarefas podem ser executadas em pararelo, este trabalho não investigou tal situação.

2. Definir os estados dos robôs, em relação à tarefa. Os robôs são alocados nos possíveis estados finais, de acordo com o fluxo de dados informado na Máquina de Estados da Figura 4.6. Em uma iteração para resolução de uma tarefa, cada robô somente terá um único estado final, dependendo de seus conhecimentos e características físicas.

3. Caso haja, selecionar um robô do estado $Apto$ e adicionar a relação robô/tarefa na lista de execução da missão. Nessa operação, é permitido que o robô α_i escolhido realize a Acomodação de Conhecimento acerca da tarefa, caso seja conveniente. Na Acomodação é utilizada a Equação (3.8), que permite a troca de conhecimento acerca da tarefa, com outros robôs do ambiente.

4. Caso não haja um robô no estado $Apto$, uma das duas operações abaixo poderá ser executada:

   (a) Selecionar um robô que esteja na condição de *Executor Forte*, que possa assimilar conhecimentos de algum outro, desde que esse esteja locado no estado *Instrutor Forte*. Nesse caso, a Assimilação de Conhecimento utiliza a função Estímulo Individual, através da Equação (3.10).

   (b) Selecionar um robô que esteja no estado *Executor Forte*, que possa assimilar conhecimentos de um grupo específico de robôs, desde que esses estejam no estado *Instrutor Fraco*. A Assimilação utiliza a função Estímulo em Grupo, Equação (3.11), uma vez que será necessário adquirir conhecimento cooperativamente.

5. Caso não haja um robô no estado *Apto* nem no *Executor Forte*, uma das duas operações abaixo poderá ser executada:

   (a) Selecionar um robô que esteja na condição de *Executor Fraco*, que possa assimilar conhecimentos de algum outro, desde que esse esteja locado no estado *Instrutor Forte*. Nesse caso, a Assimilação de Conhecimento utiliza a função Estímulo Individual, Equação (3.10).

   (b) Selecionar um robô que esteja no estado *Executor Fraco*, que possa assimilar conhecimentos de um grupo específico de robôs, desde que esses estejam no estado *Instrutor Fraco*. A Assimilação utiliza a função Estímulo em Grupo, Equação (3.11), uma vez que será necessário adquirir conhecimento cooperativamente.

   Após finalizado o processo de escolha do robô *Executor* e dos demais que passarão seus conhecimentos adiante, a lista de execução é atualizada.

6. Atualizar novamente a lista de tarefas que compõem a missão. Caso a missão não tenha sido finalizada, realizar os passos anteriores novamente.

A locação dos robôs nos estados correntes é feita de acordo com os níveis de desenvolvimento Real, Equação (3.2), e Potencial, Equação (3.3), também levando em consideração a Zona de Desenvolvimento Proximal, Equação (3.4). A cada entrada de um grupo de robôs diferente ou de uma nova missão, a lista de execução de tarefas será diferente, denominada *list_exec*. No entanto, essa lista também poderá apresentar diferentes sequencias de execuções para uma mesma missão e um mesmo grupo de robôs, desde que exista mais de um robô com mesmo conhecimento, ou mesma capacidade física. Isto se deve ao fato da escolha do "mais apto" dentre os robôs do estado *"Apto"* ser completamente aleatória. Nesse ponto, esta pesquisa não investigou heurísticas mais adequadas para análise do "mais apto". Porém, Barbosa et al. (2012) criaram heurísticas capazes de enquadrar os robôs em cenários, possibilitando a melhoria na escolha.

Contudo o módulo *Seletor de tarefas* sabe lidar com situações não triviais. De acordo com as ideias gerenciadas por esta pesquisa, que são baseadas em investigações de modelos sociais de aprendizagem, os passos acima descritos estão implementados no Algoritmo 2, que é o principal algoritmo desse módulo funcional. O algoritmo é responsável por informar a lista de seleção das tarefas pelos robôs (*list_exec*) a partir de uma missão ofertada a um ambiente contendo um grupo de robôs. Essa lista contém cada tarefa t_j associadas ao respectivo robô r_i que irá executá-la, na ordem especificada.

Como M é a missão composta pelas tarefas $T = \{\tau_1, \tau_2, \ldots, \tau_n\}$ na ordem que devem ser executadas, então τ_j possui um conjunto de conhecimentos necessários $C(\tau_j)$ e capacidades físicas requeridas $F(\tau_j)$ para sua resolução, onde $0 < j \leq n$ e n é a quantidade de tarefas da missão. Por outro lado, R são os robôs que atuam no ambiente, $R = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$. Então, α_i possui um conjunto de conhecimentos básicos $C(\alpha_i)$ e um outro de características físicas $F(\alpha_i)$, tal que $0 < i \leq m$ e que m é a quantidade de robôs do ambiente.

Algorithm 2: Seletor de tarefas IDeM-MRS

Input: Missão M e um grupo de robôs R
Output: Lista de seleção das tarefas pelos robôs $list\_exec$

```
1:  begin
2:      list_exec = ∅;
3:      tarefas = M;
4:      while tarefas ≠ ∅ do
5:          t = Selecionar_tarefa(tarefas);
6:          Setar_estados_robos(R,t);
7:          if Apto ≠ ∅ then
8:              r = Selecionar_robo(Apto);
9:              list_exec = list_exec ∪ {(t,r)};
10:         else
11:             if Executor_forte ≠ ∅ then
12:                 r = Selecionar_robo(Executor_forte);
13:                 if Instrutor_forte ≠ ∅ then
14:                     r_inst = Selecionar_robo(Instrutor_forte);
15:                     f_assimilation(r,r_inst,t);
16:                     list_exec = list_exec ∪ {(t,r)};
17:                 else
18:                     if Instrutor_fraco ≠ ∅ then
19:                         A' = Obter_grupo_robo();
20:                         if f_ε_gr(t,A') == 1 then
21:                             f_assimilation(r,A',t);
22:                             list_exec = list_exec ∪ {(t,r)};
23:             else
24:                 if Executor_fraco ≠ ∅ then
25:                     r = Selecionar_robo(Executor_fraco);
26:                     if Instrutor_forte ≠ ∅ then
27:                         r_inst = Selecionar_robo(Instrutor_forte);
28:                         f_assimilation(r,r_inst,t);
29:                         list_exec = list_exec ∪ {(t,r)};
30:                     else
31:                         if Instrutor_fraco ≠ ∅ then
32:                             A' = Obter_grupo_robo();
33:                             if f_ε_gr(t,A') == 1 then
34:                                 f_assimilation(r,A',t);
35:                                 list_exec = list_exec ∪ {(t,r)};
36:         tarefas = tarefas − {t};
37:     return list_exec;
```

O Algoritmo 2 necessita de M e R como parâmetros de entrada. A saída é uma lista *list_exec* que especifica quais robôs devem executar as tarefas da missão. Inicialmente *list_exec* está vazia (linha 2) e $tarefas$ recebe M, que compreende uma lista ordenada de tarefas (linha 3). As funções implementadas possuem as seguintes responsabilidades:

- *Selecionar_tarefa* (linha 5):
  - Escolhe uma tarefa dentre as que estão em uma lista, que é passada como parâmetro ($tarefas$).
  - No IDeM-MRS, a heurística adotada foi escolher a primeira tarefa da lista ordenada, que será a tarefa corrente da iteração. É possível adicionar novas heurísticas para a seleção de tarefas nesse procedimento.
- *Setar_estados_robos* (linha 6):
  - É responsável por enquadrar todos os robôs do conjunto R em seus devidos estados, de acordo com a tarefa analisada no momento e seguindo as respostas para a Máquina de Estados *mROBOS*, exposta na Figura 4.6.
  - Essa função é executada conforme o Algoritmo 1. Ao final, será informado quais robôs estão nos conjuntos que especificam os estados conforme a Máquina de Estados *mROBOS*.
- *Selecionar_robo* (linhas 7, 12, 14, 25 e 27):
  - Refere-se à metodologia de escolha de um robô dentre o conjunto especificado como parâmetro. Nesse ponto, a escolha está sendo realizada de forma randômica, porém pode ser agregada alguma heurística que objetive a melhoria dessa escolha, conforme o trabalho de Barbosa et al. (2012).
  - Caso haja algum robô $Apto$, a seleção de um robô para ser o $Executor$ será dentro desse estado (linha 7). Caso contrário, é feita uma série de verificações para possibilitar selecionar um robô do estado $Executor\_Forte$ (linha 12) ou $Executor\_Fraco$ (linha 25).
  - Se o robô está em $Executor\_Forte$, então um outro robô *r_inst* deve repassar conhecimento por Assimilação e/ou Acomodação. Nesse caso, *r_inst* está em $Instrutor\_Forte$ (linha 14), ou em $Instrutor\_Fraco$ (podendo ser necessário encontrar um grupo nesse estado).
  - Se o robô está em $Executor\_Fraco$, então um outro robô *r_inst* (ou um grupo de robôs A') deve repassar conhecimento por Assimilação. Nesse caso, *r_inst* está em $Instrutor\_Forte$ (linha 27) ou A' está em $Instrutor\_Fraco$ (linha 32).
- *Obter_grupo_robo* (linhas 19 e 32):
  - Escolhe um subconjunto de robôs dentre o conjunto informado por parâmetro. Essa função procura obter um conjunto A' de robôs, que juntos consigam suprir a necessidade de conhecimento acerca da tarefa em questão.

4.5 Configurações da implementação

Inicialmente a implementação foi realizada no Scilab, que é uma plataforma livre para computação numérica. Os resultados foram motivadores. Depois, essa metodologia foi implementada utilizando a linguagem de programação C++ [Deitel 2010]. Todas as implementações não necessitaram de qualquer material com capacidades extraordinárias. Nem sequer de mecanismos mais elaborados. Porém, foi utilizada a biblioteca *Standard Template Library* (STL) para auxiliar na utilização das estruturas de dados necessárias.

A estruturação do código computacional foi elaborada de forma coerente com o proposto nas premissas básicas da formulação do Problema de Execução Cooperativa de Tarefas (CETP). A Figura 4.10 mostra os arquivos de bibliotecas que dão suporte às classes de dados e operações elaboradas para a implementação do IDeM-MRS, que permitem a atuação dos módulos funcionais sem qualquer dificuldade.

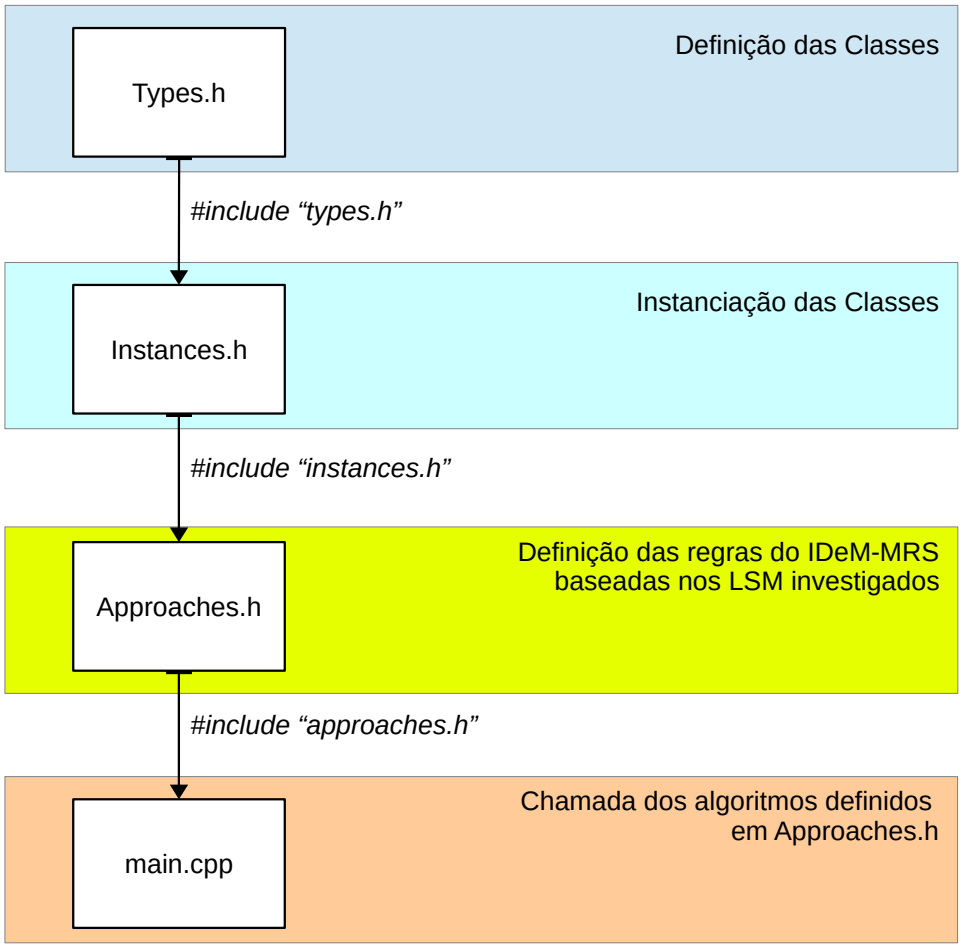

Figura 4.10: Classes implementadas.

4.5.1 Arquivo *Types.h*

O arquivo *type.h* contém a definição das classes de dados utilizados no ambiente. Toda informação a respeito dos robôs, das tarefas e da missão se encontram nesse arquivo. Ele é responsável por criar cada objeto dos tipos básicos *TRobot*, *TTask* e *TMission*. A Figura 4.11 mostra os atributos de cada classe individualmente.

As informações contidas nesse arquivo são manipuladas tanto pelo módulo funcional *Atualização* quanto pelo *Verificação de inconsistência*, mostrados na Figura 4.8. *Atualização* recebe os dados e os instancia nos objetos correspondentes. Já *Verificação de inconsistência* analisa esses objetos para identificar qualquer tipo de dados incoerente com o tipo declarado, ou para verificar se os dados inseridos estão na formatação correta que o módulo *Seletor de Tarefas* requer.

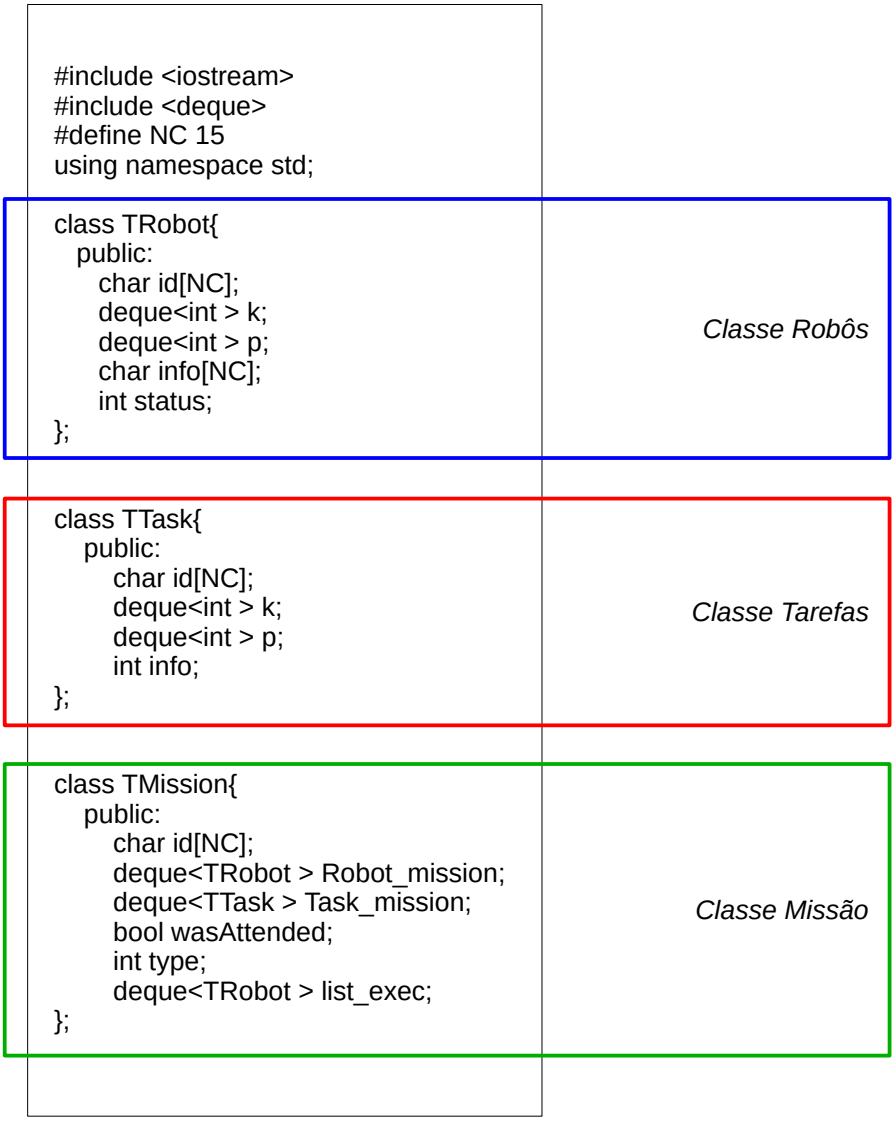

Figura 4.11: A classe Type.h contém os dados manipulados pelo formalismo.

Nesses termos, cada classe necessita ter alguns atributos, que estão descritos a seguir.

1. Classe *TTask*:

   - um identificador;
   - um conjunto de conhecimentos básicos necessários a sua execução;
   - um conjunto de capacidades físicas requeridas para sua execução;
   - informação sobre sua situação no instante t, se a tarefa está sendo executada ou está em estado de espera.

2. Classe *TRobot*:

   - um identificador;
   - um conjunto de conhecimentos básicos;
   - um conjunto de características físicas;
   - informação sobre sua situação no instante t, se o robô está sendo ocioso ou ocupado;
   - informação sobre o local de sua permanência, no instante t. Esse atributo é útil em situações em que a localização do robô tem um peso fundamental na escolha da lista de execução de tarefas.

3. Classe *TMission*:

   - um identificador;
   - um conjunto de robôs disponíveis no ambiente para execução das tarefas;
   - uma lista de tarefas que deve ser executada para que a missão seja alcançada;
   - um *flag* que informe se a missão foi executada com sucesso;
   - uma lista de execução que contém cada tarefa associada ao robô (ou ao grupo de robôs) que irá executá-la.

4.5.2 Arquivo *Instances.h*

Instances.h é responsável por instanciar os elementos, de acordo com os tipos de dados (contidos em *Types.h*). A cada nova configuração do ambiente, é realizada uma nova instanciação dos elementos, que são informados ao *instances.h* como parâmetros de entrada, através de arquivos *arq_Robot*, *arq_Task* e *arq_Mission*. Após a leitura destes, o *instance.h* aloca seus dados nos devidos atributos.

1. Informações dos Robôs:

O *arq_Robot* é o arquivo que contém informações acerca dos robôs que atuam no ambiente. A implementação requer que essas informações estejam dispostas em quarto partes, de acordo como mostra a Figura 4.12. Na primeira parte constam informações gerais que expressam a quantidade de robôs e o número de conhecimentos e características físicas disponíveis individualmente para o grupo. A segunda parte trata especificamente dos robôs, com informações de identificação, local de atuação (tarefa) e status. As duas últimas partes informam, respectivamente e na forma matricial, os conhecimentos e as características físicas que cada robô possui.

A tabela 4.4 mostra exemplos de dados que podem conter nesse arquivo de entrada. O dado $info_r$ dos robôs estão setados para *xxx* porque não está especificado o local de sua posição, já que, inicialmente, os robôs ainda não foram designados a alguma tarefa. O ideal é saber a posição inicial do robô. A partir dela, as outras serão estimadas em $t+1$, $t+2$ e assim por diante.

Todos possuem $status_r$ igual a 0 significando que estão ociosos, à espera de alguma tarefa. Também são informadas as matrizes de conhecimento e de característica física de cada robô. No exemplo, o arquivo mostra um conjunto de 10 robôs com possibilidade máxima de 10 diferentes tipos de conhecimentos e características físicas.

Robô (id_r)	**Conhecimento** ($reqc_r$)	**Característica física** ($reqf_r$)	**Informação** ($info_r$)	**Status** ($status_r$)
R08	$\{c_4\}$	$\{f_1\}$	xxx	0
R30	$\emptyset$	$\{f_1, f_2\}$	xxx	0
R10	$\{c_1\}$	$\{f_2, f_3\}$	xxx	0
R79	$\{c_1, c_4\}$	$\{f_1\}$	xxx	0
R76	$\{c_2\}$	$\{f_1, f_2, f_3\}$	xxx	0
R03	$\{c_1, c_3\}$	$\{f_2\}$	xxx	0
R07	$\{c_3\}$	$\{f_1, f_3\}$	xxx	0
R09	$\emptyset$	$\{f_1, f_2, f_3\}$	xxx	0
R05	$\{c_2, c_3\}$	$\{f_3\}$	xxx	0
R33	$\{c_2, c_3, c_4\}$	$\{f_1, f_2\}$	xxx	0

Tabela 4.4: Exemplo de dados contidos no arquivo de entrada *arq_Robot*.

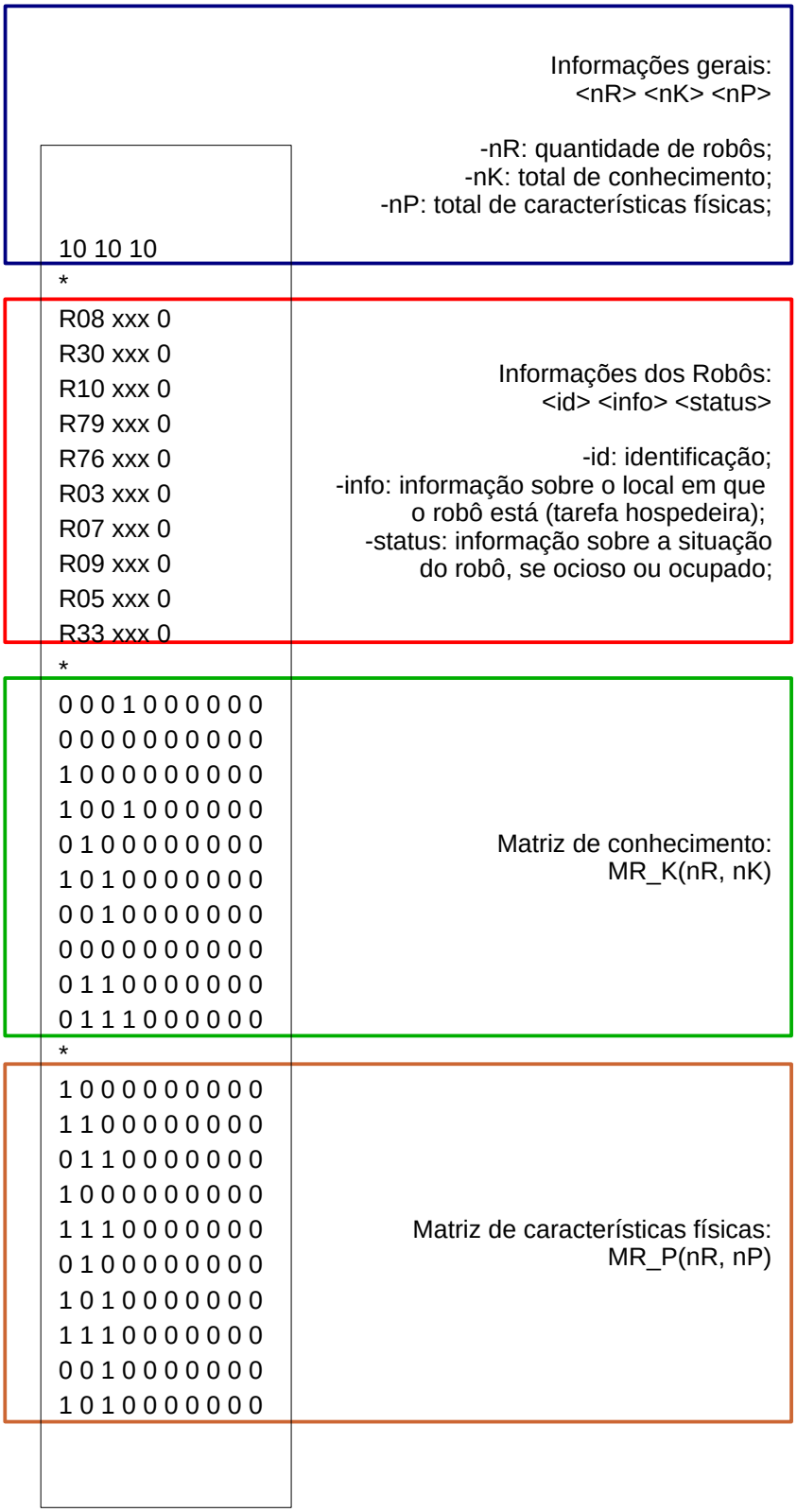

Figura 4.12: Estrutura do arquivo de entrada dos robôs (*arq_Robot*).

2. Informações da missão:

O arquivo com os dados da missão (*arq_Mission*) mantém o padrão de três partes, conforme mostrado na Figura 4.13. A primeira informa a identificação da missão. Já a segunda trata da lista de tarefas que deve ser executada, de forma ordenada, para que a missão tenha sucesso. A última parte informa a identificação dos robôs que podem atuar em conjunto para a execução total da missão.

A Tabela 4.5 apresenta um exemplo de dados que podem dispor o arquivo de entrada da missão. É importante salientar que as tarefas devem ser executadas na ordem em que aparecem dispostas no arquivo de entrada. Dessa forma, $T01$ deve ser a primeira tarefa a ser executada. Imediatamente após, $T79$, seguida de $T12$ e assim por diante, até que todas sejam finalizadas.

Figura 4.13: Estrutura do arquivo de entrada da missão (*arq_Mission*).

Missão (id_m)	**Lista de tarefas** (T)	**Conjunto de robôs** (R)
M01	{T01, T79, T12, T08, T15, T10}	{R08, R30, R10, R79, R76, R03, R07, R09, R05, R33}

Tabela 4.5: Exemplo de dados contidos no arquivo de entrada *arq_Mission*.

3. Informações das tarefas:

O arquivo com os dados das tarefas (*arq_Task*), representa as tarefas conforme o *arq_Robot* representa os robôs. *arq_Task* contém informações acerca de todas as tarefas que podem ser requeridas para resolução da missão. Conforme o arquivo de robôs, a implementação também requer que as informações estejam dispostas em quarto partes, de acordo como mostra a Figura 4.14.

A primeira parte do arquivo agrupa informações gerais que especificam a quantidade de tarefas e o número de conhecimentos e características físicas que podem ser requeridas pelas tarefas individualmente. Já a segunda parte contém informações acerca das tarefas, como identificação e situação de execução. Na forma matricial, as duas últimas partes informam os conhecimentos e as características físicas que cada tarefa, respectivamente, requer para sua resolução.

A Tabela 4.6 exemplifica dados que podem conter nesse arquivo de entrada. No exemplo, o arquivo está informando uma lista de 10 tarefas ($T01$, $T03$, $T06$, $T79$, $T16$, $T18$, $T12$, $T08$, $T15$, $T10$) com possibilidade máxima de 4 diferentes tipos de conhecimentos (c_1, c_2, c_3 e c_4) e 3 capacidades físicas (f_1, f_2 e f_3).

As tarefas, quando dispostas na lista de execução da missão, estão classificadas de acordo com três estados em relação à execução. Elas podem pertencer a "aguardando execução"($info_t = 0$), "em execução"($info_t = 1$) e "fializada"($info_t = 2$). Contudo, o dado $info_t$ está inicialmente setado para 0, indicando que todas as tarefas estão no estado "aguardando execução". A cada alteração de estado, há também uma alteração nesse dado. Sendo assim, os valores que indicam essa informação são:

- $info_t = 0$ indica que nenhum robô foi designado para a execução da tarefa.
- $info_t = 1$ afirma que a tarefa está sendo executada por algum robô.
- $info_t = 2$ afirma que a tarefa foi concluída com sucesso.

Tarefa (id_t)	**Conhecimento** ($reqc_t$)	**Característica física** ($reqf_t$)	**Informação** ($info_t$)
T01	$\{c_1\}$	$\{f_3\}$	0
T03	$\{c_1, c_2, c_3\}$	$\{f_1, f_3\}$	0
T06	$\{c_2, c_3\}$	$\{f_1, f_2\}$	0
T79	$\{c_2\}$	$\{f_2\}$	0
T16	$\{c_2, c_4\}$	$\{f_1\}$	0
T18	$\{c_4\}$	$\{f_2\}$	0
T12	$\{c_1, c_3\}$	$\{f_3\}$	0
T08	$\{c_3\}$	$\{f_1\}$	0
T15	$\{c_4\}$	$\{f_1, f_2, f_3\}$	0
T10	$\{c_1, c_4\}$	$\{f_1, f_2\}$	0

Tabela 4.6: Exemplo de dados contidos no arquivo de entrada *arq_Task*.

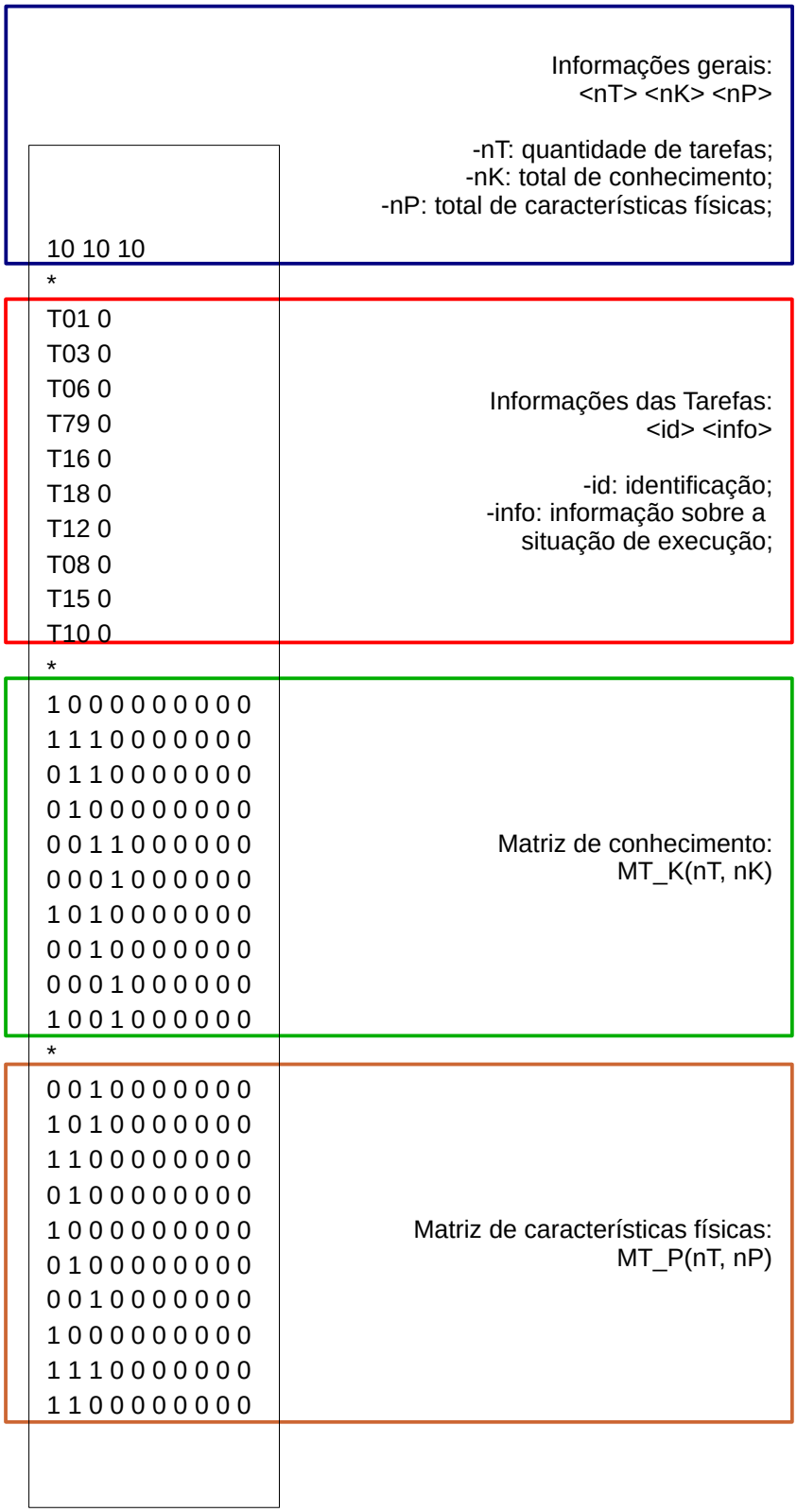

Figura 4.14: Estrutura do arquivo de entrada das tarefas (*arq_Task*).

4.5.3 Arquivo *Approaches.h*

O arquivo *Approaches.h* encapsula toda operação advinda dos modelos sociais de aprendizagem analisados, de acordo com as classes responsáveis. Quatro classes foram desenvolvidas: $Vygotsky$, $Behaviorist$, $Piaget$ e $Social\_Behaviorist$, que são manipuladas pelo módulo *Seletor de tarefas*. A Figura 4.15 mostra as classes implementadas nesse arquivo.

De acordo com cada abordagem de aprendizagem social, cada classe implementa suas funções específicas, através das equações definidas por esta pesquisa. A saber:

1. Classe *Vygotsky*:
   - Função referente ao Nível de Desenvolvimento Real (*Level of Real Development* - LRD), f_lrd
     - Equação (3.2);
   - Função referente ao Nível de Desenvolvimento Potencial (*Level of Potential Development* - LPD), f_lpd
     - Equação (3.3);
   - Função referente à Zona de Desenvolvimento Proximal (*Zone of Proximal Development* - ZPD), f_zpd
     - Equação (3.4);
2. Classe *Behaviorist*:
   - Função EstímuloResposta, $f\_\varepsilon r$
     - Equação (3.10);
   - Função EstímuloConhecimento, $f\_\varepsilon c$
     - Equação (3.9);
3. Classe *Piaget*:
   - Função Assimilação, $f\_assimilation$
     - Equação (3.6);
   - Função Acomodação, $f\_accommodation$
     - Equação (3.8);
4. Classe *Social_Behaviorist*:
   - Função Estímulo em Grupo, $f\_\varepsilon gr$
     - Equação (3.11).

```
#include <numeric>
#include <iostream>
#include <deque>
using std::cout;
using std::endl;

#include "instances.h"

class Vygotsky{
    public:

        Vygotsky ();
        int f_lrd (TRobot* r, TTask* t);
        int f_lpd (TRobot* r, TTask* t);
        int f_zpd (TRobot* r, TTask* t);
};

class Behaviorist{
    public:

        Behaviorist ();
        int f_εr (TTask* t, TRobot* r);
        deque<int> f_εc (TTask* t, TRobot* r);
};

class Piaget{
    public:

        Piaget ();
        int f_assimilation (TRobot* r_1, TRobot* r_2, TTask* t);
        int f_accommodation (TRobot* r, int k_1, int k_2);
};

class Social_Behaviorist{
    public:

        Social_Behaviorist ();
        int f_εgr (TTask* t, deque<TRobot*> A);
};
```

Classe Vygotsky

Classe Behaviorista

Classe Piaget

Classe Social e Behaviorista

Figura 4.15: Classes que implementam as teorias dos modelos sociais.

Capítulo 5

Experimentos e Resultados do IDeM-MRS

Esse capítulo mostra a validação das ações dos módulos funcionais do IDeM-MRS para casos simulados de execução cooperativa de tarefas por um grupo de robôs. São mostrados os experimentos (através de simulações) do modelo e a comparação desses resultados com os dos experimentos que não estão sendo orientados pelas regras expostas no IDeM-MRS.

Os testes foram realizados com simulações de ambientes com as definições construídas para o IDeM-MRS. Inicialmente a implementação foi realizada com o auxílio de uma ferramenta voltada para resolução de problemas numéricos, o Scilab, já que essa fase teve por finalidade validar a consistência do formalismo matemático proposto. Em um segundo momento, os métodos foram implementados com a linguagem de programação C++, utilizando o paradigma orientado a objetos, com o auxílio da biblioteca *Standard Template Library* - STL, [Deitel 2010].

Os resultados informam uma solução para o problema de cooperação entre um grupo de robôs para um fim específico, que nesse caso, é a execução de uma missão, que é comum a todos do grupo.

Como forma de avaliar todos os parâmetros da instância, foram adotados 03 cenários com diferentes heurísticas construtivas com o intuito de explorar a configuração inicial do ambiente multirrobô. Essas heurísticas modificam randomicamente algumas partes integrantes do ambiente, sem deixá-lo diferente, simulando um rearranjo do espaço trabalhado:

1. Permutação dos robôs;

2. Permutação das tarefas na lista;

3. Permutação dos robôs e das tarefas na lista;

As permutações visam gerar diferentes espaços de busca para gerar novas soluções da missão. Com esse intuito, o conjunto de robôs, a lista de tarefas e ambos são alterados e avaliados para adequação do modelo.

5.1 Definição de Cenários

5.1.1 Cenário 1: Permutação dos robôs (RO)

O primeiro critério permuta a lista de robôs disponíveis para comunicação (RO). Essa heurística atua somente sobre o conjunto de robôs R, permutando seus elementos. A Figura 5.1 mostra um exemplo de uma permutação em R. No caso, o conjunto inicial $R = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5\}$, após a permutação RO, passa a ser $R = \{\alpha_3, \alpha_2, \alpha_4, \alpha_1, \alpha_5\}$. Já a lista de tarefas $T = \{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5\}$ permanece igual.

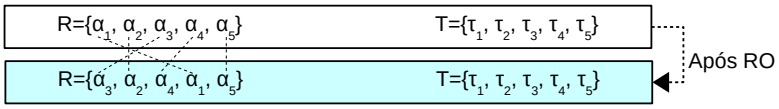

Figura 5.1: Exemplo de uma permutação gerada pela heurística construtiva RO.

5.1.2 Cenário 2: Permutação das tarefas na lista (TA)

Esse critério permuta as tarefas da lista, sem alterar o conjunto de robôs. A Figura 5.2 mostra que, após a permutação TA, a lista de tarefas inicial ($T = \{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5\}$) passa para $T = \{\tau_4, \tau_1, \tau_3, \tau_2, \tau_5\}$, e o conjunto dos robôs R permanece o mesmo.

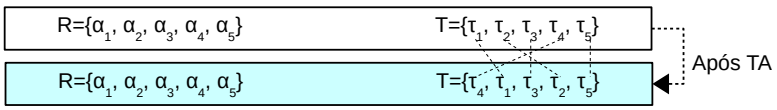

Figura 5.2: Exemplo de uma permutação gerada pela heurística construtiva TA.

5.1.3 Cenário 3: Permutação dos robôs e das tarefas na lista (RT)

O terceiro critério, denominado RT, permuta ambos, tanto R quanto T. A Figura 5.3 mostra essa situação. O conjunto de robôs se altera para $R = \{\alpha_3, \alpha_2, \alpha_4, \alpha_1, \alpha_5\}$ e a lista de tarefas, para $T = \{\tau_4, \tau_1, \tau_3, \tau_2, \tau_5\}$. Essas alterações nos elementos de cada componente se faz de maneira randômica.

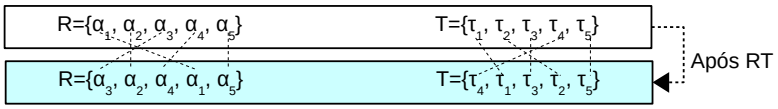

Figura 5.3: Exemplo de uma permutação gerada pela heurística construtiva RT.

5.2 Codificação das instâncias

Os experimentos foram realizados utilizando um grande número de instâncias, sendo necessário criar uma forma de organizar os resultados avaliados. Por isso, foi decidido criar um mapeamento para cada nomenclatura do experimento, adotando uma codificação em 5 grupos, que especificam:

1. quantidade de robôs ativos do ambiente;
2. quantidade de tarefas na Missão;
3. quantidade de ciclos que o algoritmo principal é executado;
4. heurística construtiva utilizada;
5. metodologia de execução, se com o modelo IDeM-MRS ou não, e a ordem de execução da instância.

A Figura 5.4 mostra esse esquema de códigos. No exemplo, a instância é definida com a codificação $10R10T\_100\_RO\_S1$, sendo assim, é avaliada para: 10 robôs heterogêneos ($10R$); a execução de uma Missão contendo 10 tarefas diferentes ($10T$); uma execução com 100 ciclos (100); com a heurística construtiva com permutação dos robôs (RO); a metodologia com aprendizagem em sua primeira amostra ($S1$).

Os códigos referentes à heurística construtiva e à metodologia, escolhidas para cada instância de teste, estão mostrados na Tabela 5.1.

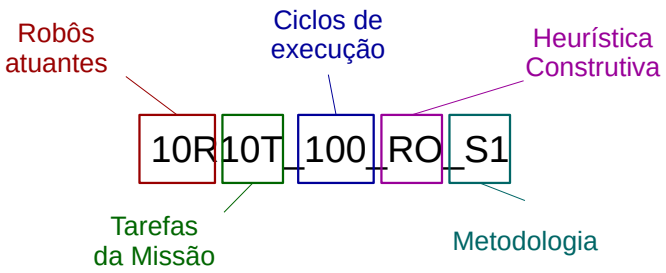

Figura 5.4: Esquema de codificação das instâncias de testes.

Grupo	Codificação	Descrição
Heurística	*RO*	Modifica os robôs ativos
Heurística	*TA*	Modifica as tarefas que constam na missão
Heurística	*RT*	Modifica tanto os robôs quanto as tarefas
Metodologia	*Sx*	Utiliza o modelo IDeM-MRS, em sua x-ésima amostra
Metodologia	*Nx*	Não utiliza o modelo IDeM-MRS, em sua x-ésima amostra

Tabela 5.1: Códigos da heurística construtiva e da metodologia, que compõem a codificação das instâncias no IDeM-MRS.

5.3 Quantidade de testes realizados

Os experimentos foram baseados em duas vertentes. A primeira não trabalha com a proposta do modelo IDeM-MRS. A segunda implementa a metodologia imposta pelo modelo. No entanto, para ambos os casos, foram analisadas 05 instâncias diferentes para cada heurística construtiva. Além disso, um total de 100 ciclos foram requisitados para cada instância. Nesse caso, somando as 5 instâncias por cada heurística e multiplicando pelo total de ciclos, encontra-se uma quantidade de 1.500 execuções. Porém, como se tem duas vertentes, esse número sobre para 3.000 execuções. A Figura 5.5 demostra um esquema para facilitar o entendimento do valor total de experimentos realizados.

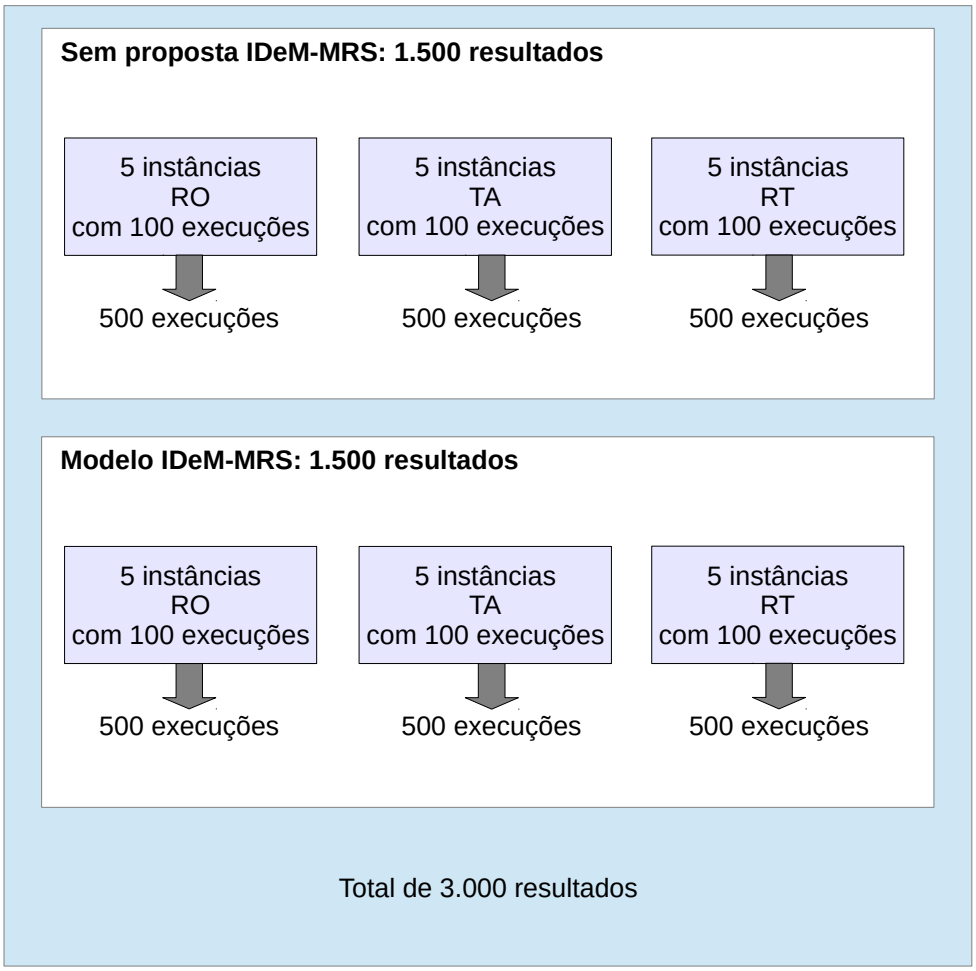

Figura 5.5: Quantidade de testes realizados.

5.4 Experimentos sem o IDeM-MRS

O primeiro experimento foi a simulação do ambiente sem qualquer política de ação entre os robôs contidos nele, e sem qualquer modelagem dos conceitos investigados neste trabalho. O Algoritmo 3, denominado *Seletor de tarefas sem cooperação*, define os passos principais para o experimento e foi implementado como uma forma de obter a lista de execução da missão, conforme a real ideia de execução. Não foi utilizado qualquer forma de cooperação entre os robôs do ambiente para a geração da lista de execução $list\_exec$.

O Algoritmo *Seletor de tarefas sem cooperação* estabelece uma lista de robôs para execução de uma missão, sem a utilização de qualquer melhoria na escolha dos responsáveis por cada tarefa. Igualmente ao Algoritmo 2 (*Seletor de tarefas IDeM-MRS*), este também necessita da missão M e de um conjunto de robôs R como parâmetros de entrada, além da saída ser uma lista $list\_exec$, que especifica exatamente os robôs que devem executar as tarefas da missão.

Inicialmente $list\_exec$ está vazia (linha 2) e $tarefas$ recebe uma cópia de M, que compreende uma lista ordenada de tarefas (linha 3). A função $Selecionar\_tarefa$ (linha 5) escolhe uma tarefa dentre as que estão na lista $tarefas$, passada como parâmetro. Escolhe-se a primeira tarefa da lista ordenada que será a tarefa corrente da iteração.

A função $Obter\_aptos$ (linha 6) seleciona os robôs, dentre os que estão contidos em R, os que podem executar a tarefa t. Contudo, $Aptos$ representa esses robôs. Por fim, a função *Selecionar_robo* (linha 8) escolhe randomicamente um robô dentre o conjunto informado como parâmetro.

Ao final do algoritmo, caso alguma tarefa seja uma impossível de ser executada, a missão não é finalizada com sucesso (linha 11). Caso contrário, as tarefas são distribuídas pelo grupo de robôs e a missão é completamente executadas (linha 13).

Algorithm 3: Seletor de tarefas sem cooperação

Input: Missão M e um grupo de robôs R
Output: Lista de seleção das tarefas pelos robôs $list\_exec$

```
1:  begin
2:      list_exec = ∅;
3:      tarefas = M;
4:      while tarefas ≠ ∅ do
5:          t = Selecionar_tarefa(tarefas);
6:          Aptos = Obter_aptos(R,t);
7:          if Aptos ≠ ∅ then
8:              r = Selecionar_robo(Aptos);
9:              list_exec = list_exec ∪ {(t,r)};
10:         else
11:             return Tarefa_impossivel;
12:         tarefas = tarefas − {t};
13:     return list_exec;
```

5.4.1 Resultados Obtidos sem cooperação

Os dados obtidos com a execução do Algoritmo 3 (*Seletor de tarefas sem cooperação*), que não utiliza qualquer mecanismo de cooperação para as demais instâncias analisadas, informam os valores obtidos com os experimentos para os seguintes contextos:

1. **Quantidade de tarefas da missão:**
   Tarefas que estão dispostas na lista ordenada que compreende a missão. Para que a missão seja considerada finalizada com sucesso, necessariamente todas as tarefas devem ser executadas.

2. **Quantidade de mensagens:**
   Mensagens que os robôs enviam para o ambiente, a cada solicitação de informações.

3. **Percentual de tarefas executadas pelos robôs:**
   Essa informação é dada baseada no total de tarefas solicitadas pela missão (sendo 100%). O cálculo é realizado através da Equação (5.1), onde n_1 é a quantidade de tarefas resolvidas da missão e n é a quantidade total de tarefas da missão.

$$ftask_{exec} = \frac{n_1.100}{n} \tag{5.1}$$

4. **Tempo de resposta do ambiente:**
   Tempo que o ambiente conseguiu retornar uma resposta às solicitações (mensagens) dos robôs durante toda execução da missão. Essa informação é dada em segundos.

5. **Quantidade de conhecimento do ambiente:**
   Quantidade de conhecimento que está disposto no ambiente. É todo conhecimento que cada robô do grupo pode conter durante a execução, ou ainda, já possuir como conhecimento inicial.

6. **Quantidade de conhecimento inicial:**
   Quantidade de conhecimento que o grupo de robôs possui antes da execução da missão. É calculada através de número total de conhecimento diferentes dos robôs, antes da execução.

7. **Quantidade de conhecimento final:**
   Quantidade de conhecimento que o grupo de robôs possui após a execução. É calculada através de número total de conhecimento diferentes dos robôs, após a execução.

8. **Total de conhecimento adquirido:**
   Essa informação é a diferença (em percentual) entre o conhecimento final e o inicial.

A tabela 5.2 dispõe as informações referentes aos 4 primeiros itens acima. Os demais itens estão dispostos na Tabela 5.3.

Testes		Tarefas		Mensagens	
Codificação	**Instância**	**Da missão (unidade)**	**Executadas (%)**	**No ambiente (unidade)**	**De respostas (s)**
10R10T_100_RO	N1	1000	40,0	600	34
	N2	1000	40,0	600	35
	N3	1000	40,0	600	33
	N4	1000	40,0	600	33
	N5	1000	40,0	600	33
10R10T_100_TA	N1	1000	40,0	600	37
	N2	1000	40,0	600	37
	N3	1000	40,0	600	35
	N4	1000	40,0	600	36
	N5	1000	40,0	600	33
10R10T_100_RT	N1	1000	40,0	600	45
	N2	1000	40,0	600	43
	N3	1000	40,0	600	42
	N4	1000	40,0	600	42
	N5	1000	40,0	600	40

Tabela 5.2: Resultados considerando a execução das tarefas e a troca de mensagens entre o ambiente, sem IDeM-MRS.

Testes		Total de conhecimento			
Codificação	**Instância**	**Do ambiente (unidade)**	**Inicial (unidade)**	**Final (unidade)**	**Adquirido (%)**
10R10T_100_RO	N1	40	13	13,0	0,0
	N2	40	13	13,0	0,0
	N3	40	13	13,0	0,0
	N4	40	13	13,0	0,0
	N5	40	13	13,0	0,0
10R10T_100_TA	N1	40	13	13,0	0,0
	N2	40	13	13,0	0,0
	N3	40	13	13,0	0,0
	N4	40	13	13,0	0,0
	N5	40	13	13,0	0,0
10R10T_100_RT	N1	40	13	13,0	0,0
	N2	40	13	13,0	0,0
	N3	40	13	13,0	0,0
	N4	40	13	13,0	0,0
	N5	40	13	13,0	0,0

Tabela 5.3: Resultados considerando a aquisição de conhecimento sem IDeM-MRS.

5.4.2 Avaliação dos resultados obtidos sem cooperação

Conforme o esperado, a abordagem sem cooperação não consolidou aquisição de conhecimento por parte dos agentes robóticos. Os 10 robôs *r* iniciaram a execução com quantidades de conhecimento iguais aos estabelecidos na Tabela 5.4. No entanto, ao fim das execuções, cada instância analisada não ofereceu alteração dessa configuração, visto na Tabela 5.5. Os robôs, ao término, tinham exatamente os mesmos conhecimentos do início do experimento.

A Figura 5.6 mostra graficamente a configuração inicial do ambiente. Isso faz concluir que, de fato, não houve melhoria na experiência do grupo, em termos de conhecimento adquirido com a execução da missão. As 1500 execuções das amostras $10R10T\_100\_RO\_N$, $10R10T\_100\_TA\_N$ e $10R10T\_100\_RT\_N$ demonstram que a configuração do ambiente permaneceu estática, sem qualquer alteração, conforme mostram as figuras 5.7, 5.8 e 5.9 respectivamente.

Identificação do robô	Número de conhecimento
r1	1
r2	0
r3	1
r4	2
r5	1
r6	2
r7	1
r8	0
r9	2
r10	3

Tabela 5.4: Quantidade de conhecimento que cada robô iniciou o processo de execução dos experimentos.

Identificação do robô	Quantidade de conhecimento		
	10R10T_100_RO_N	10R10T_100_TA_N	10R10T_100_RT_N
r1	1	1	1
r2	0	0	0
r3	1	1	1
r4	2	2	2
r5	1	1	1
r6	2	2	2
r7	1	1	1
r8	0	0	0
r9	2	2	2
r10	3	3	3

Tabela 5.5: Quantidade de conhecimento que cada robô finalizou o processo de execução dos experimentos, sem as regras do IDeM-MRS e de acordo com cada instância.

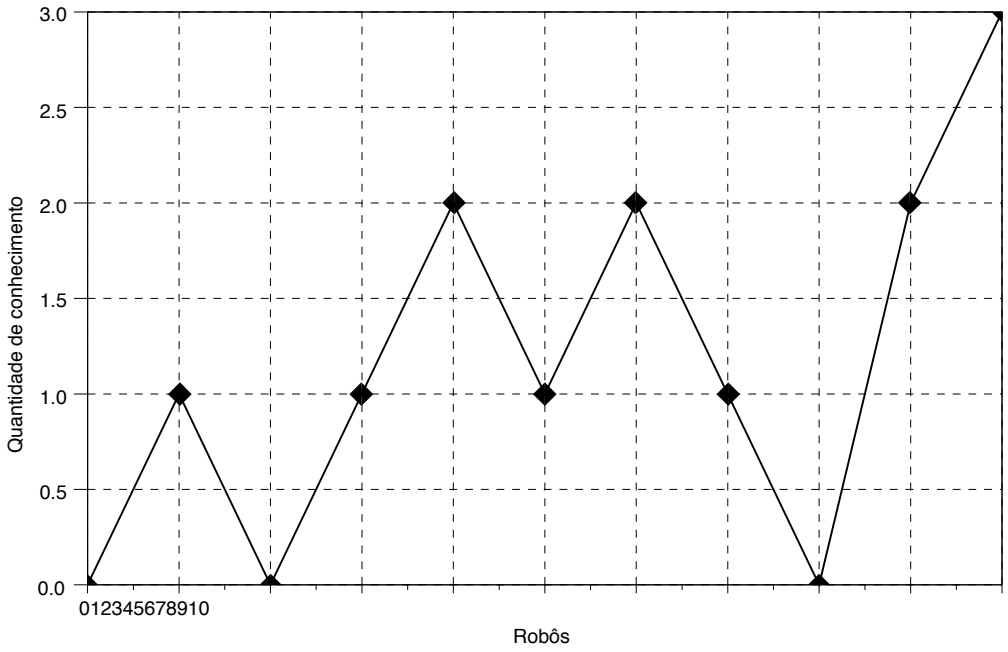

Figura 5.6: Quantidade de conhecimento inicial de cada robô para os experimentos sem o IDeM-MRS.

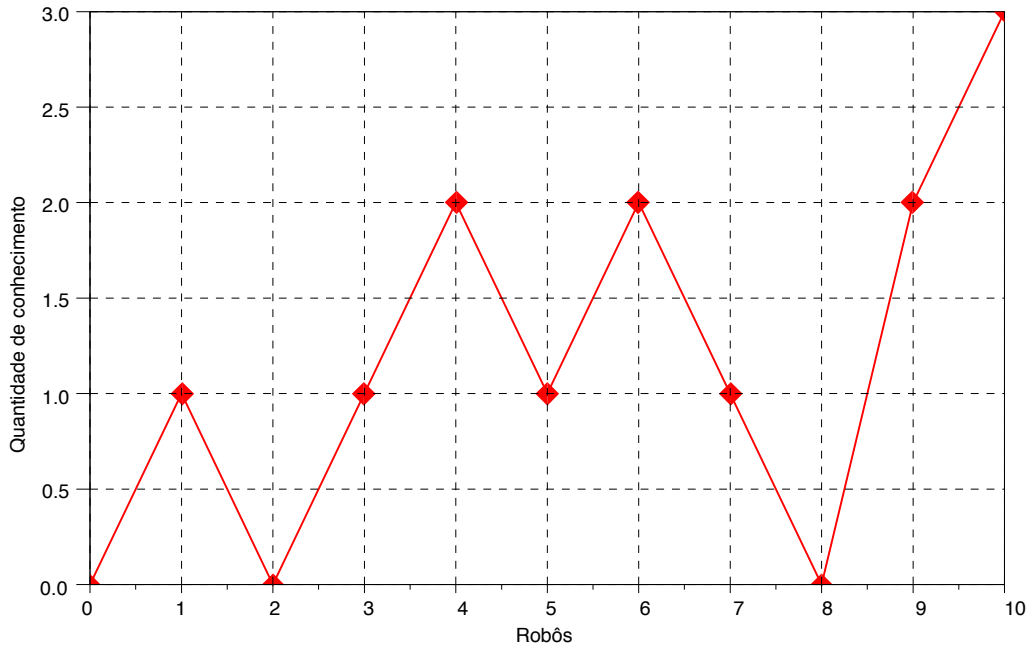

Figura 5.7: Quantidade de conhecimento de cada robô, após os experimentos das instâncias 10R10T_100_RO_N.

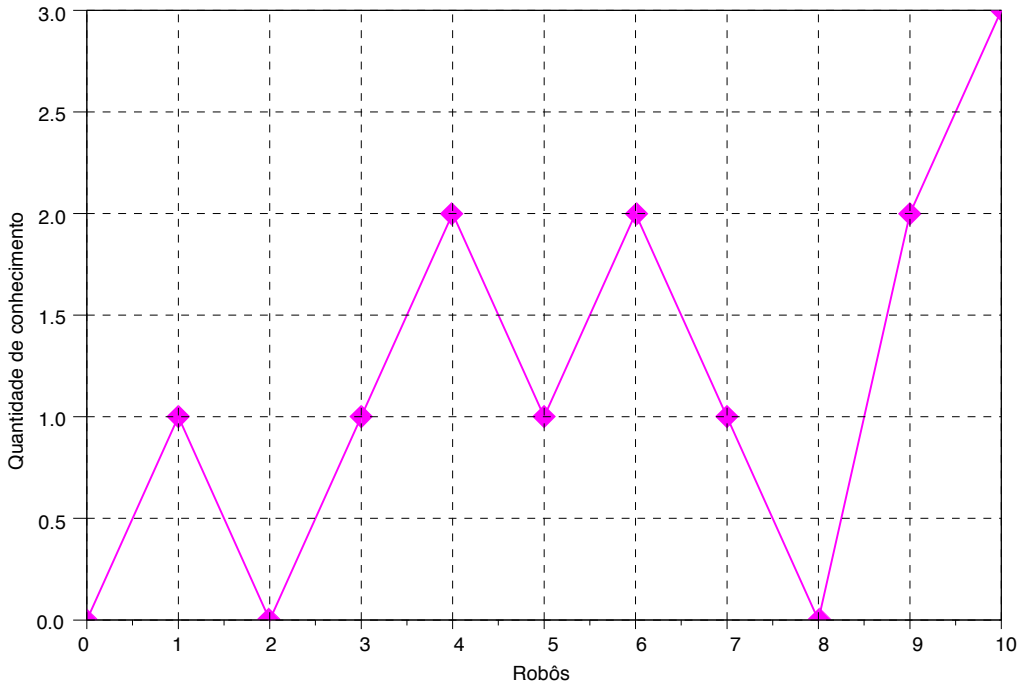

Figura 5.8: Quantidade de conhecimento de cada robô, após os experimentos das instâncias 10R10T_100_TA_N.

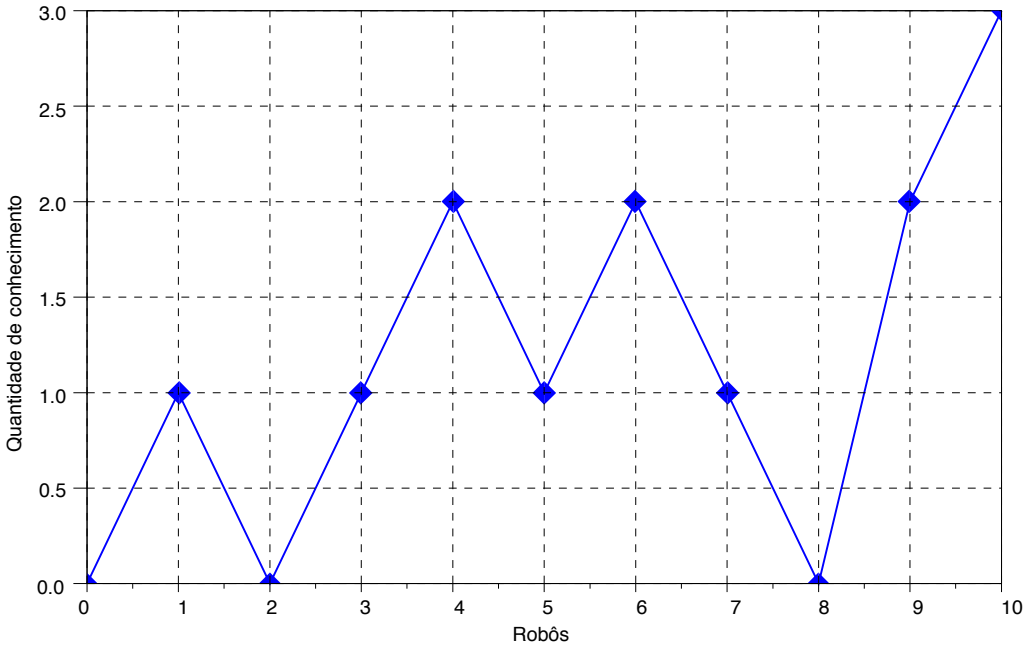

Figura 5.9: Quantidade de conhecimento de cada robô, após os experimentos das instâncias 10R10T_100_RT_N.

5.5 Experimentos com IDeM-MRS

Os experimentos foram realizados para comprovar a hipótese de aquisição de conhecimento oferecida pelo IDeM-MRS. A metodologia dessa proposta segue exatamente os módulos funcionais da Figura 4.8, através do Algoritmo 4, chamado *IDeM-MRS Principal*. Assim sendo, algumas funções possuem responsabilidades específicas:

- *Atualizar_ambiente* (linha 2): atualiza o ambiente segundo as informações passadas como parâmetro, referentes à missão M, ao grupo de robôs R e à lista T.

- *Verificar_inconsistencias* (linha 3): procedimento que analisa se há alguma incoerência nos dados passados após a atualização do ambiente. Um exemplo de incoerência é a falta de alguma informação necessária referente às tarefas (identificação, conjunto de conhecimento e característica física requeridos e status corrente), aos robôs ou à missão.

- *Receber_heuristica* (linha 4): analisa se o critério exploratório é a permutação dos robôs (*RO*), permutação das tarefas (*TA*) ou permutação dos robôs e das tarefas em conjunto (*RT*).

- *Aplicar_heuristica* (linha 5): faz as devidas permutações no ambiente multirrobô, conforme o critério exploratório inferido por *Receber_heuristica*. A metodologia de que se trata cada um desses critérios é apresentada na Seção 5.1.

- *Seletor_tarefas* (linha 6): se refere ao Algoritmo 2 exposto no Capítulo 4. É responsável por informar a lista que contém cada tarefa associada a um robô que irá executá-la.

- *Executar* (linha 7): realiza a execução da lista *list_exec* estabelecida segundo as regras do IDeM-MRS. Neste trabalho, essa execução é simulada.

Algorithm 4: IDeM-MRS principal

Input: Missão M e um grupo de robôs R
Output: Lista de seleção das tarefas pelos robôs *list_exec*

```
1:  begin
2:      Atualizar_ambiente(M,R,T);
3:      if Verificar_inconsistencias(M,R,T) == 0 then
4:          h = Receber_heuristica();
5:          Aplicar_heuristica(h);
6:          list_exec = Selector_tarefas(M,R);
7:          Executar(list_exec);
8:          return 1;
9:      else
10:         return 0;
```

5.5.1 Resultados Obtidos com IDeM-MRS

Os testes foram realizados através de três experimentos, separados pelo critério exploratório:

1. **Experimento 1:**
   Foi utilizado a herística construtiva *RO* (permutação de robôs) como critério exploratório da solução inicial, para cada ciclo de execução. As instâncias são codificadas como $10R10T\_100\_RO\_S$.

2. **Experimento 2:**
   Foi utilizado a heurística construtiva *TA* (permutação de tarefas). As instâncias são codificadas como $10R10T\_100\_TA\_S$.

3. **Experimento 3:**
   Foi utilizado a heurística construtiva *RT* (permutação de robôs e de tarefas). As instâncias são codificadas como $10R10T\_100\_RT\_S$.

Igualmente aos experimentos realizados na abordagem sem as regras do IDeM-MRS, o ambiente foi configurado conforme está demonstrado na Tabela 5.4. Os dados obtidos com a execução do Algoritmo 4, *IDeM-MRS Principal*, encontram-se nas tabelas 5.6 e 5.7. Também de acordo com a abordagem anterior, os contextos analisados foram:

1. Quantidade de tarefas da missão. Para que a missão seja considerada finalizada com sucesso, necessariamente todas as tarefas devem ser executadas.

2. Quantidade de mensagens que os robôs enviam para o ambiente, a cada solicitação de informações.

3. Percentual de tarefas executadas pelos robôs.

4. Tempo (em segundos) que o ambiente conseguiu retornar uma resposta às solicitações (mensagens) dos robôs durante toda execução.
5. Quantidade total de conhecimento que o ambiente é capaz de possuir.

6. Quantidade de conhecimento inicial que o grupo de robôs possui antes da execução da missão.
7. Quantidade de conhecimento final (uma média aritmética após a execução) que o grupo de robôs possui.

8. Total de conhecimento adquirido (em percentual) durante a execução.

A Tabela 5.6 dispõe as informações referentes às tarefas resolvidas e às trocas de mensagens. Em relação ao conhecimento adquirido pelos robôs individualmente, a Tabela 5.7 informa os dados.

Testes		Tarefas		Mensagens	
Codificação	**Instância**	**Da missão (unidade)**	**Executadas (%)**	**No ambiente (unidade)**	**De respostas (s)**
10R10T_100_RO	S1	1000	90,0	100	69
	S2	1000	90,0	100	76
	S3	1000	90,0	100	87
	S4	1000	90,0	100	69
	S5	1000	90,0	100	72
10R10T_100_TA	S1	1000	93,1	69	81
	S2	1000	93,0	70	92
	S3	1000	93,4	66	94
	S4	1000	93,0	70	85
	S5	1000	93,8	62	87
10R10T_100_RT	S1	1000	94,6	54	81
	S2	1000	94,3	57	87
	S3	1000	93,9	61	89
	S4	1000	94,6	54	90
	S5	1000	93,9	61	94

Tabela 5.6: Resultados considerando a execução das tarefas e a troca de mensagens entre o ambiente, com as regras do IDeM-MRS.

Testes		Total de conhecimento			
Codificação	**Instância**	**Do ambiente (unidade)**	**Inicial (unidade)**	**Final (unidade)**	**Adquirido (%)**
10R10T_100_RO	S1	40	13	17,6	17,0
	S2	40	13	17,5	16,7
	S3	40	13	17,4	16,3
	S4	40	13	17,1	15,2
	S5	40	13	17,5	16,7
10R10T_100_TA	S1	40	13	19,7	24,8
	S2	40	13	19,6	24,4
	S3	40	13	19,6	24,4
	S4	40	13	19,7	24,8
	S5	40	13	19,7	24,8
10R10T_100_RT	S1	40	13	17,6	17,0
	S2	40	13	17,7	17,4
	S3	40	13	17,9	18,1
	S4	40	13	17,6	17,0
	S5	40	13	18,0	18,5

Tabela 5.7: Resultados considerando a aquisição de conhecimento com as regras do IDeM-MRS.

5.5.2 Avaliação dos resultados quanto ao ganho de conhecimento

A avaliação dos resultados evolui em três patamares relacionados: ao ganho de conhecimento pelo ambiente, à execução das tarefas e ao tempo de resposta do ambiente. Em relação ao ganho de conhecimento adquirido pelo ambiente ao se utilizar o formalismo proposto, os testes realizados permitem concluir que houve aquisição de conhecimento, se caracterizando como aprendizagem. Ao término dos testes, os robôs estavam com mais conhecimento do que no início do processo. A Tabela 5.8 mostra a média de conhecimento que cada robô finalizou o processo de execução. Cada instância melhorou sua condição inicial.

A configuração inicial do ambiente (Figura 5.10) faz concluir que houve cooperação dos robôs para a melhoria do ambiente. Os resultados das 1500 execuções das amostras $10R10T\_100\_RO\_S$ (Experimento 1), $10R10T\_100\_TA\_S$ (Experimento 2) e $10R10T\_100\_RT\_S$ (Experimento 3), permitem concluir que houve ganho de conhecimento e, por isso, houve aprendizagem.

As instâncias que agiram sobre soluções iniciais baseadas na heurística *TA* permitiram uma aprendizagem mais eficiente. No Experimento 1, houve assimilação de conhecimento e os resultados apresentaram, em termos do conhecimento total adquirido por cada robô, um ganho médio de 16,4%.

O Experimento 2 finalizou os testes com um ganho médio de conhecimento de 24,7% para o ambiente (melhor que o experimento anterior). Uma atenção especial deve-se voltar para o robô *r*2 que conseguiu incorporar o máximo de conhecimento, mesmo inicialmente não possuindo sequer algum. Já o Experimento 3 resultou na média de 17,6% de aumento no conhecimento do ambiente. Quando comparados à abordagem sem o IDeM-MRS, esses números informam a melhoria do ambiente multirrobô. As figuras 5.11, 5.12 e 5.13 demonstram graficamente a média final de conhecimento que cada robô finalizou os experimentos.

Identificação do robô	**Conhecimento Inicial**	**Conhecimento Final**		
		Experimento 1	**Experimento 2**	**Experimento 3**
r1	1	1.0	1.0	1.0
r2	0	1.4	4.0	1.2
r3	1	1.1	1.0	1.2
r4	2	2.0	2.0	2.0
r5	1	2.2	1.0	2.3
r6	2	2.0	2.3	2.1
r7	1	1.1	2.3	1.2
r8	0	1.5	1.0	1.6
r9	2	2.1	2.0	2.1
r10	3	3.1	3.0	3.2

Tabela 5.8: Média de conhecimento que cada robô finalizou o processo de execução dos experimentos com as regras do IDeM-MRS, de acordo com cada instância.

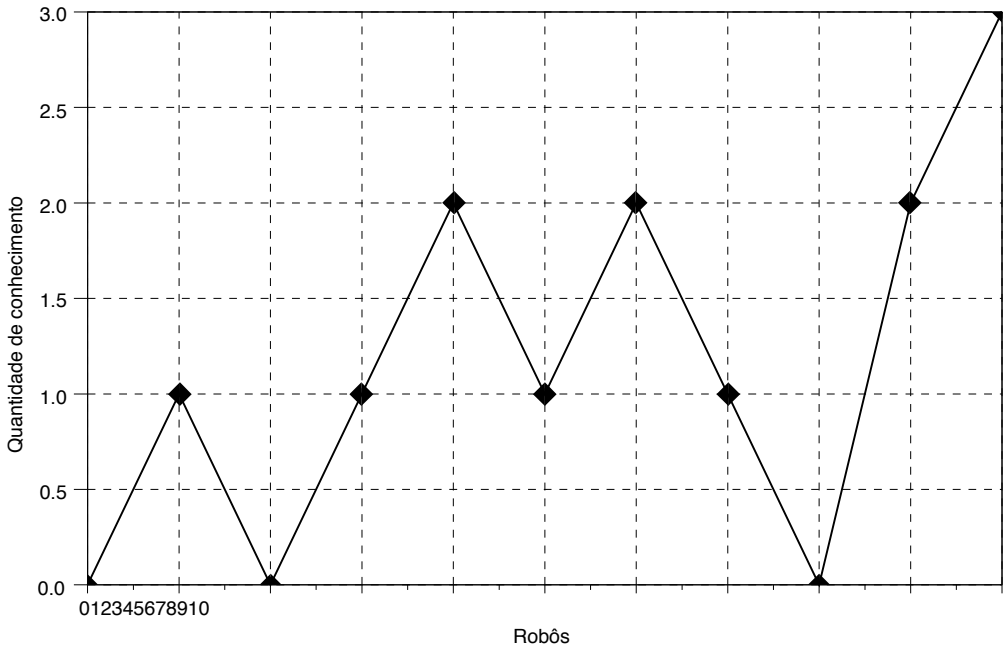

Figura 5.10: Quantidade de conhecimento inicial de cada robô para os experimentos com o IDeM-MRS.

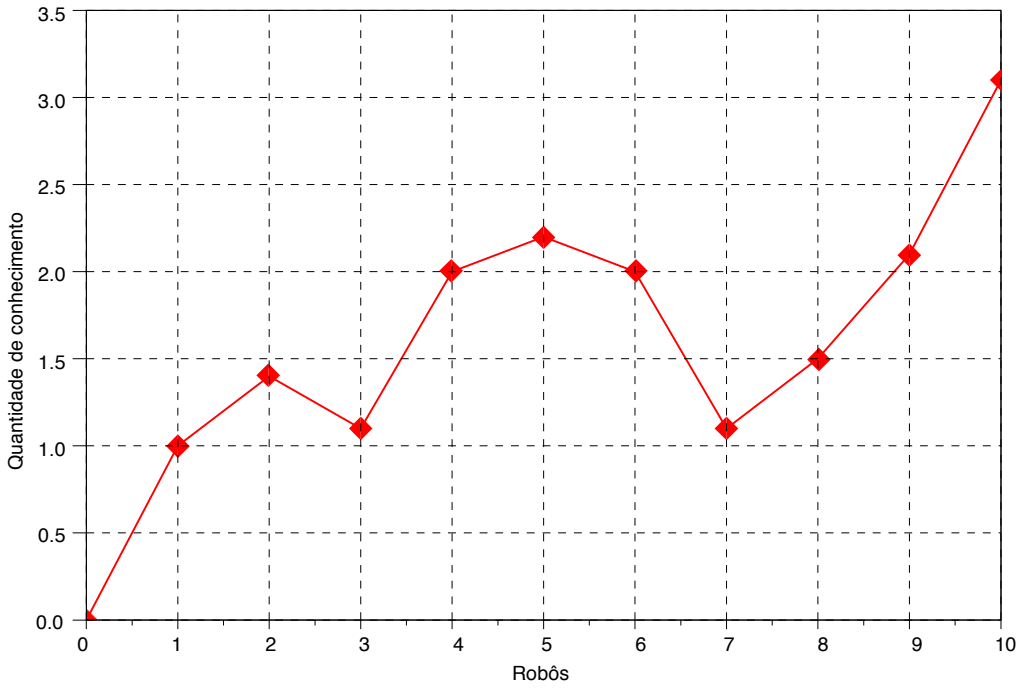

Figura 5.11: Quantidade de conhecimento de cada robô, após o Experimento 1 (instâncias 10R10T_100_RO_S).

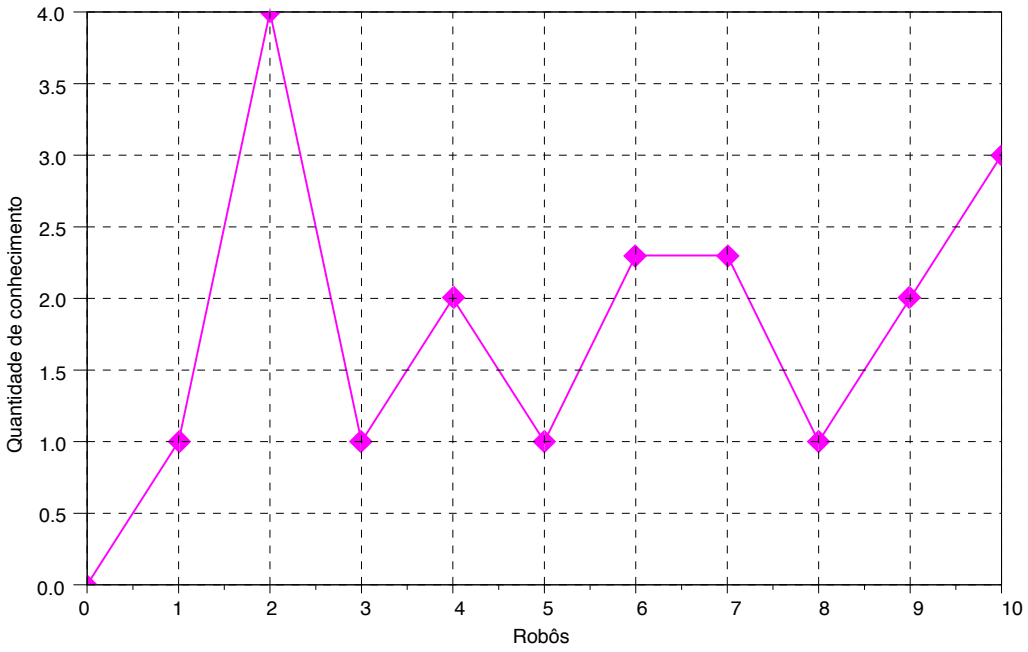

Figura 5.12: Quantidade de conhecimento de cada robô, após o Experimento 2 (instâncias 10R10T_100_TA_S).

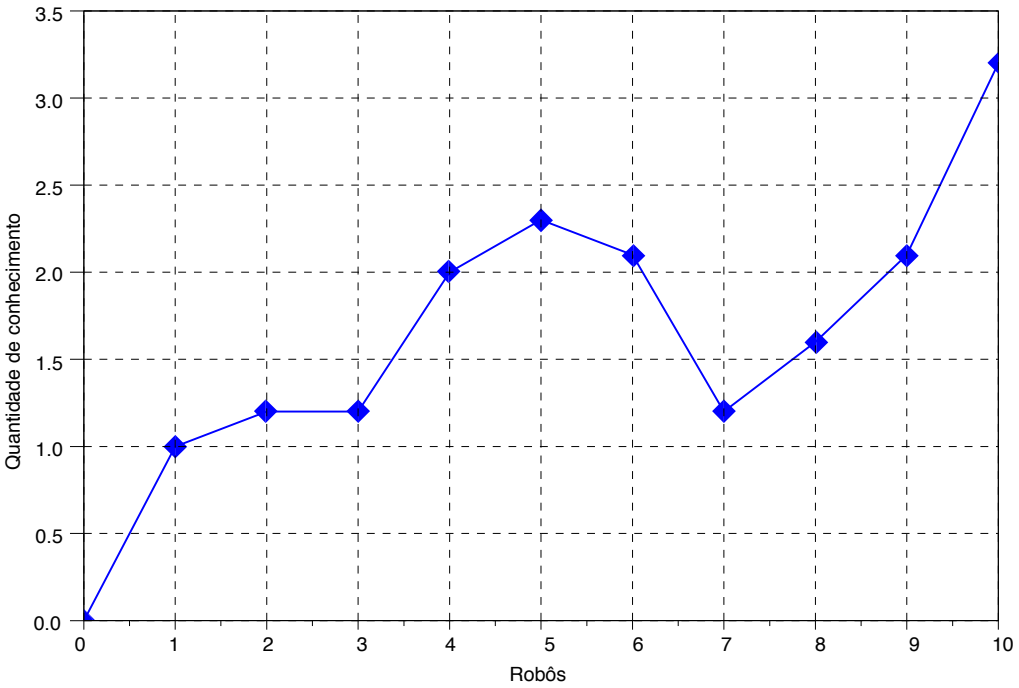

Figura 5.13: Quantidade de conhecimento de cada robô, após o Experimento 3 (instâncias 10R10T_100_RT_S).

Concluindo o benefício relacionado ao conhecimento adquirido através do IDeM-MRS, a Figura 5.14 mostra a eficiência das amostras entre si, sendo calculado em termos percentuais conforme a Tabela 5.9 mostra. As instâncias com o formalismo mostraram ganhos de até 24,7% de aumento de conhecimento para o ambiente.

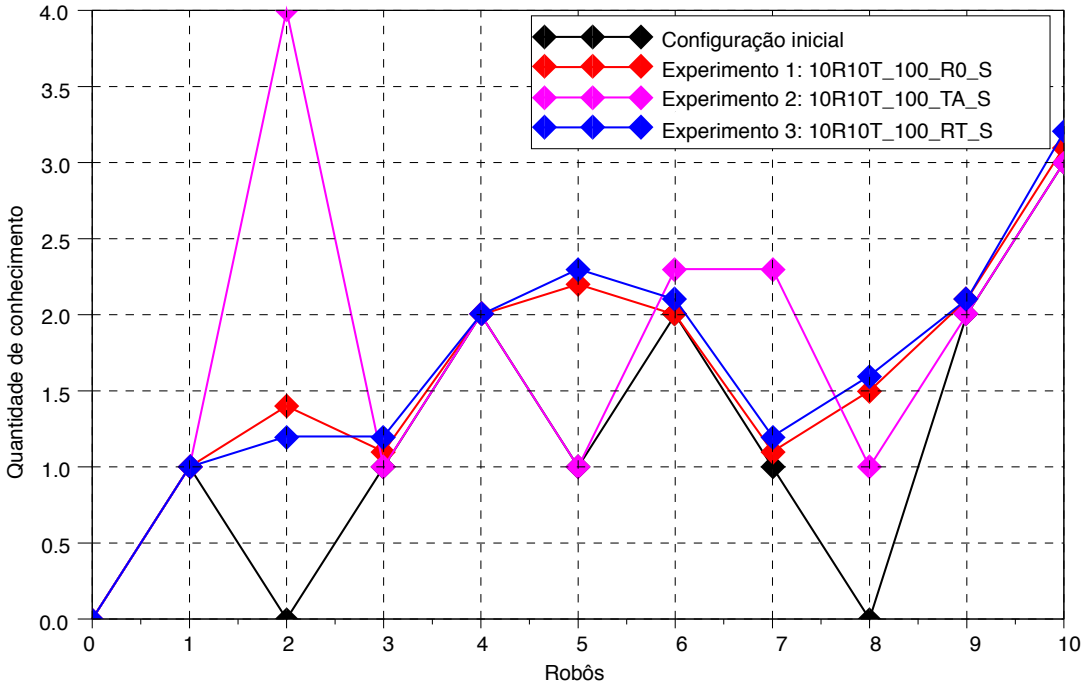

Figura 5.14: Conhecimentos adquiridos após os experimentos do IDeM-MRS.

Instância		**Ganho para o ambiente**
Sem IDeM-MRS	10*R*10*T*_100_*RO*_*N*	0,0%
	10*R*10*T*_100_*TA*_*N*	0,0%
	10*R*10*T*_100_*RT*_*N*	0,0%
Com IDeM-MRS	10*R*10*T*_100_*RO*_*S*	16,4%
	10*R*10*T*_100_*TA*_*S*	24,7%
	10*R*10*T*_100_*RT*_*S*	17,6%

Tabela 5.9: Resultados para conhecimentos adquiridos após os experimentos.

5.5.3 Avaliação dos resultados quanto à execução das tarefas

O gráfico da Figura 5.15 demonstra o percentual fixo de 40% de realização das tarefas requisitadas ao ambiente, quando exposto à abordagem sem a utilização do formalismo. Em contra-partida, foi calculado uma média de 92,5% de execução das tarefas requisitadas, quando utilizando o IDeM-MRS (ver Tabela 5.10). Nesse caso, o modelo IDeM-MRS possui maior probabilidade de execução das tarefas de uma missão, com autonomia (sem necessidade de algum agente externo ao ambiente), reduzindo consideravelmente a presença de tarefas impossíveis.

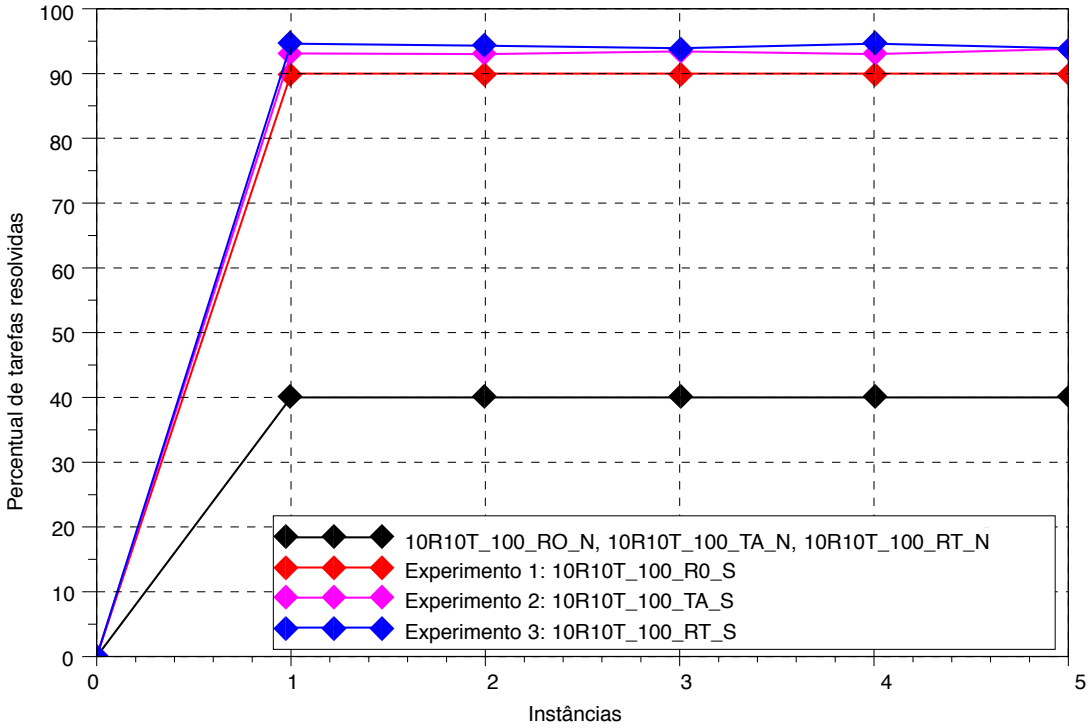

Figura 5.15: Tarefas executadas pelas instâncias.

Instância		Tarefas resolvidas
Sem IDeM-MRS	$10R10T\_100\_RO\_N$	40,0%
	$10R10T\_100\_TA\_N$	40,0%
	$10R10T\_100\_RT\_N$	40,0%
Com IDeM-MRS	$10R10T\_100\_RO\_S$	90,0%
	$10R10T\_100\_TA\_S$	93,3%
	$10R10T\_100\_RT\_S$	94,3%

Tabela 5.10: Resultados para tarefas resolvidas após os experimentos.

5.5.4 Avaliação quanto ao tempo de resposta do ambiente

O tempo de resposta para a execução da missão, foi diferente para todas as intâncias. O gráfico da Figura 5.16 mostra os tempos gastos nas instâncias analisadas. Porém, são pequenas diferenças se for considerado a unidade mensurada: segundos (ver Tabela 5.11).

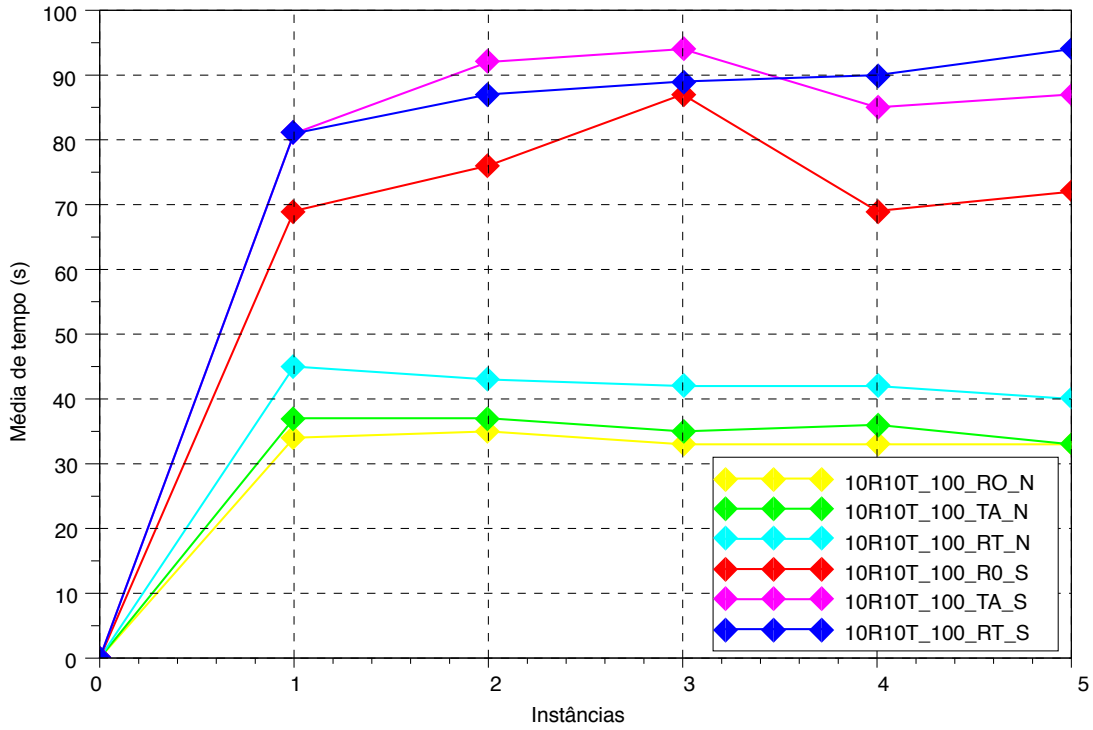

Figura 5.16: Tempo médio gasto para finalização dos experimentos.

Instância		Tempo médio gasto (s)
Sem IDeM-MRS	$10R10T\_100\_RO\_N$	33,6
	$10R10T\_100\_TA\_N$	35,6
	$10R10T\_100\_RT\_N$	42,4
Com IDeM-MRS	$10R10T\_100\_RO\_S$	74,6
	$10R10T\_100\_TA\_S$	87,8
	$10R10T\_100\_RT\_S$	88,2

Tabela 5.11: Resultados para o tempo médio gasto para finalização dos experimentos.

Capítulo 6

Conclusão

Neste trabalho, foi proposto um formalismo de aprendizagem que pode ser usado por sistemas multirrobôs para a resolução cooperativa de tarefas, utilizando abordagens sociais no processo de cooperação dos robôs. Para tal, foi proposto e implementado um formalismo matemático baseado nas teorias sociais de aprendizagem [Vygotsky 1967, Vygotsky 1978, Vygotsky 1991, Vygotsky 1993, Piaget 1975, Piaget & Inhelder 1982, Watson 1913, Wenger 2000], usando um modelo implementacional inicialmente baseado em Scilab e posteriormente em C++, visando testar sua aplicabilidade em sistemas multiagentes robóticos.

O cerne do modelo baseia-se na capacidade de troca de informação entre os agentes (um protocolo de comunicação é pressuposto). Esta comunicação encontra-se presente na maioria dos sistemas multiagentes. Na prática, a troca de código executável pode ser, em última análise, parte da solução procurada, em que um robô consegue passar, a outro, o código executável que ele conhece (ou parte dele) para a execução de tarefas. Com isso, o time de robôs consegue, efetivamente, assimilar ou adquirir conhecimentos a partir de interações com humanos (em hipótese), ou aumentar o conhecimento individual.

Conjecturas sobre como ocorre a aquisição de novos conhecimentos de forma autônoma não faz parte do objeto de estudo desta tese, mas o sistema proposto pode ser adaptado para atuar também nessas situações (onde uma forma de aprendizado automatizado esteja presente).

Verificando os resultados empíricos e avaliando em termos de unidades de conhecimentos adquiridos, é possível concluir que o formalismo permite a troca de experiências e, com isso, influencia positivamente na cooperação do grupo em função da execução de tarefas. Somando-se a isso e avaliando os resultados em termos de quantidades de tarefas executadas pelo ambiente autonomamente (sem a interferência externa), é possível perceber que o formalismo direciona o ambiente, propiciando liberdade de ação, sabedoria e autonomia aos robôs componentes do grupo, independentemente das ações de agentes externos.

Após o formalismo ser executado, sem nenhuma heurística na escolha do robô mais apto (dentre os aptos) a passar experiências aos demais, ou dos que deveriam ter prioridade na execução da tarefa atual, o resultado mostrou ganho de conhecimento com simples trocas de mensagems entre os robôs.

Mesmo com muitas características peculiares de um ambiente multirrobô, com essa pesquisa torna-se possível utilizar um formalismo matemático com regras de aprendiza-

gem de qualquer grupo de robôs inseridos em um ambiente fechado e dinâmico. Essas regras devem ser responsáveis por tornar a cooperação o mais eficiente possível entre o grupo, e são baseadas em regras sociais de indivíduos reais.

Após a execução do problema de execução cooperativa de tarefas, observa-se que o sistema multirrobô obtém um ganho de conhecimento, conforme comprovado pelos experimentos realizados nesta pesquisa. Cada abordagem analisada foi responsável pela construção de formulações para o objetivo final, concebendo um modelo matemático fiel às sugestões para os conceitos dos robôs, mapeados de acordo com estudos para indivíduos reais.

Segundo [Reinelt 1994], resultados empíricos, em aplicações práticas, nem sempre permitem obter soluções viáveis devido a várias razões, como a modelagem incorreta do problema em questão ou a falta de tempo suficiente para encontrar a solução ideal. Contudo, diante dos resultados obtidos pelos experimentos, é possível concluir que o formalismo proposto é capaz de atuar nessas situações reais, com algumas modificações. E ainda prover benefícios ao ambiente em termos de conhecimento individual de cada robô, conforme foi comprovado empiricamente. Esta pesquisa, portanto, permite conjecturar que outras aplicações mais gerais, envolvendo a resolução de problemas de grupo, podem se utilizar de tal abordagem com sucesso.

6.1 Trabalhos Futuros

Este pesquisa pode evoluir através de alguns pontos, descritos abaixo:

1. Aplicação do IDeM-MRS em sistemas multirrobôs reais, com base em um espaço pré-definido e conhecido. O IDeM-MRS pode atuar em estudos de caso de mapeamento e localização de robôs. A localização inicial dos robôs deve ser conhecida, previamente. Nesse caso, pode ser utilizado o mapeamento de ambientes por robôs em uma representação em grade de ocupação 3D (visual), com localização simultânea (VSLAM).

2. Adicionar heurísticas para apoio às escolhas quando se tem mais de um robô em mesma situação. Com isso procura-se a otimização de escolhas.

3. Aplicar esse modelo a grandes grupos de entidades, heterogêneas ou não e contrapor os resultados com metodologias utilizadas atualmente para esse fim, tais como enxames de abelhas e colônia de formigas, entre outros.

4. Implementar a *Camada de Percepção* e a *Camada de Operação*. Com a inclusão desses componentes se torna possível, respectivamente, receber a configuração do ambiente através de percepções de mecanismos de hardware, e executar a missão através de comandos aos elementos físicos responsáveis por tais funções nos robôs.

5. Adicionar incertezas no conhecimento repassado. Quando não se tem certeza do conhecimento do robô (ou do grupo) que está em condição de instrução de outro,

uma boa alternativa seria analisar a fonte desse conhecimento.

6. Verificar a eficiência do IDeM-MRS em missões com tarefa paralelas. Experimentar o modelo para prover a cooperação de robôs em situações em que alguns estarão executando diferentes tarefas ao mesmo tempo.

7. Analisar a cooperação dos robôs para resolução de uma tarefa conhecida por todos do grupo. Nesse caso, não há troca de experiências, mas há uma forte comunicação entre os robôs do grupo. Como teste prático, por exemplo, pode-se criar uma missão onde o objetivo é juntar as peças pretas em um determinado local do ambiente.

8. Analisar a adição da cooperação particionada, que pode ser em duas direções:
   - cooperação particionada no conhecimento, em que um robô pode ensinar parte de uma tarefa para outro. O restante da informação pode ser repassada, ou não, por um outro robô do ambiente.

   - cooperação particionada na capacidade física, em que os robôs devem compartilhar recursos de hardware, quando nenhum robô do ambiente possui toda capacidade requerida. O IDeM-MRS prevê esse tipo de cooperação através do estímulo em grupo, porém de fato, não foi possível propor resultados factíveis para essa alternativa através da simulação. Portanto, a incorporação efetiva dessa cooperação de recursos de hardware pode ser obtida com o desenvolvimento de um trabalho futuro, excedendo a esfera da simulação para o real.

Referências Bibliográficas

Barbosa, Danilo Ricardo, Rosiery Maia & Anderson Souza (2012), 'Heurística para gerenciar a cooperação de um grupo de robôs em execução cooperativa de tarefas', *Holos - ISSN 1807-1600* **1**(0).

Barrios-Aranibar, Dennis & Pablo Javier Alsina (2007), Imitation learning: An application in a micro robot soccer game, *em* N.Nedjah, L.dos Santos Coelho & L.de Macedo Mourelle, eds., 'Mobile Robots: The Evolutionary Approach', Vol. 50 de *Studies in Computational Intelligence*, Springer, pp. 201–219.

Beni, G. (2005), From swarm intelligence to swarm robotics, *em* 'Proceedings of the SAB 2004 Workshop on Swarm Robotics, Lecture notes in Computer Science', 3342, Santa Monica, CA, USA, pp. 1–9.

Billard, Aude & Maja Mataric (2001), 'Learning human arm movements by imitation: Evaluation of a biologically-inspired connectionist architecture', *Robotics and Autonomous Systems* **941**, 1–16.

Bonabeau, Eric, Marco Dorigo & Guy Theraulaz (2001), 'Swarm intelligence: From natural to artificial systems', *J. Artificial Societies and Social Simulation* pp. –1–1.

Borzello, E. & Laurence D. Merkle (2005), Multi-agent cooperation using the ant algorithm with variable pheromone placement, *em* 'Congress on Evolutionary Computation'05', pp. 1232–1237.

Botelho, S. & R. Alami (2000), Robots that cooperatively enhance their plans, *em* 'Proceeding of 5th International Symposium on Distributed Autonomous Robotic Systems DARS2000, Lecture notes in Computer Science, Springer Verlag'.

Brucker, Peter (2002), 'Scheduling and constraint propagation', *Discrete Applied Mathematics* **123**(1-3), 227–256.

Brucker, Peter (2004), *Scheduling algorithms (4. ed.)*, Springer.

Cao, Y. U., Alex S. Fukunaga & A. B. Kahng (1997), 'Cooperativemobile robotics: Antecedents and directions', pp. 1–23.

Carrascosa, C., J. Bajo, V. Julian, J. M. Corchado & V. Botti (2008), 'Hybrid multi-agent architecture as a real-time problem-solving model', *Expert Syst. Appl.* **34**(1), 2–17.

Carver, Norman, Zarko Cvetanovic & Victor R. Lesser (1991), Sophisticated cooperation in fa/c distributed problem solving systems, *em* 'AAAI'91', pp. 191–198.

Cass, SStephen (2001), 'Robosoccer: A new breed of robots takes to the playing field', pp. 75–77.

Chaimowicz, L., T. Sugar, V. Kumar & M. F. M. Campos (2001), An architecture for tightly coupled multi-robot cooperation, *em* 'ICRA', IEEE, pp. 2992–2997.

Chen, Deng-Neng, B. Jeng, Wei-Po Lee & Cheng-Hung Chuang (2008), 'An agent-based model for consumer-to-business electronic commerce', *Expert Systems with Applications* **34**(1), 469 – 481.

Deitel, Paul J. (2010), *C++ How to Program. P.J. Deitel, H.M. Deitel*, 7th[a] edição, Pearson Education.

Durfee, Edmund H. (2001*a*), Distributed problem solving and planning, *em* 'EASSS'01', pp. 118–149.

Durfee, Edmund H. (2001*b*), 'Scaling up agent coordination strategies', *Computer* **34**(7), 39–46.

Engelberger, J. F. (1993), 'Health-care robotics goes commercial: the helpmate experience', **11**, 517–523.

Fox, M. S. (1981), 'An organization view of distributed systems', **11**(1), 70–80.

Frawley, W. (2000), *Vygotsky e a Ciência Cognitiva: linguagem e integração das mentes social e computacional*, Artmed, São Paulo.

Graf, R. & P. Weckesser (1998), 'Autonomous roomservice in a hotel', pp. 641–647.

He, Yulan, Siu Cheung Hui & Yongxiang Sim (2006), *Information Retrieval Technology*, Vol. 4182, Lecture notes in Computer Science, Springer Berlin / Heidelberg, capítulo A Novel Ant-Based Clustering Approach for Document Clustering, pp. 537–544.

Jennings, Nicholas R. (1993), 'Specification and implementation of a belief desire-joint intention architecture for cooperative problem solving', *Int. J. Cooperative Inf. Syst.* pp. 289–318.

J.Piaget (1985), *The Equilibration of Cognitive Structures: The Central Problem of Intellectual Development*, University of Chicago Press, Chicago.

Kalmar, Zsolt, Csaba Szepesvari & Andras Loerincz (1998), 'Module-based reinforcement learning: Experiments with a real robot', *Machine Learning* **31**, 55.

Kambayashi, Y., Y. Tsujimura, H. Yamachi, M. Takimoto & H. Yamamoto (2009), Design of a multi-robot system using mobile agents with ant colony clustering, *em* 'System Sciences, 2009. HICSS '09. 42nd Hawaii International Conference on', pp. 1–10.

Lau, H. Y. K. (2007), 'An immunity approach to strategic behavioral control', **20**(3), 289–306.

León, A., E.F. Morales, L. Altamirano & J.R. Ruiz (2011), Teaching a robot new tasks through imitation and feedback, *em* 'Proc. ICML Workshop: New Developments in Imitation Learning'.

Liu, Lin & Yuehuan Wang (2008), Multi-agent cooperation method based on intention recognition, *em* 'Proceedings of the 2008 Congress on Image and Signal Processing, Vol. 4 - Volume 04', CISP '08, IEEE Computer Society, Washington, DC, USA, pp. 216–219.

Maia, Rosiery, Anderson Souza & Luiz Gonçalves (2011), Concepção de um formalismo de aprendizagem baseado em modelos sociais para um time de robôs em execução cooperativa de tarefas, *em* 'Anais do X Simpósio Brasileiro de Automação Inteligente', pp. 575–581.

Malone, Thomas W. & Kevin Crowston (1994), 'The interdisciplinary study of coordination', *ACM Computing Surveys* **26**, 87–119.

Martinoli, Alcherio, K. Easton & W. Agassounon (2004), 'Modeling swarm robotic systems: a case study in collaborative distributed manipulation', **23**, 415–436.

Mes, M., M. Van Der Heijden & A. Van Harten (2007), 'Comparison of agent-based scheduling to look-ahead heuristics for real-time transportation problems', **181**, 59–75.

Murphy, Robin R. (2000), *Introduction to AI Robotics*, fisrt[a] edição, MIT Press, Cambridge, Massachusetts, EUA.

Nawarecki, Edward, Grzegorz Dobrowolski, Stanis Ciszewski & Marek Kisiel-Dorohinicki (2003), Ontology of cooperating agents by means of knowledge components, *em* 'Proceedings of the 3rd Central and Eastern European conference on Multi-agent systems', CEEMAS 03, Springer-Verlag, Berlin, Heidelberg, pp. 180–190.

Noreils, Fabrice R. (1993), 'Toward a robot architecture integrating cooperation between mobile robots: application to indoor environment', *Int. J. Rob. Res.* **12**(1), 79–98.

Oliveira, D. & A. L. C. Bazzan (2006), 'Traffic lights control with adaptive group formation based on swarm intelligence', pp. 520–521. Lecture notes in Computer Science, including subseries Lecture notes in Artificial Intelligence and Lecture notes in Bioinformatics, 4150 LNCS.

Oliveira, M. K. (1997), *Vygotsky: aprendizado e desenvolvimento um processo sócio-histórico*, Scipione, Sao Paulo.

Piaget, Jean (1975), *A equilibração das estruturas cognitivas*, Zahar, Rio de Janeiro.

Piaget, Jean & Barbel Inhelder (1982), *A psicologia da criança*, Difel, São Paulo.

Reinelt, Gerhard (1994), *The Traveling Salesman: Computational Solutions for TSP Applications*, Vol. 840 de *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, Germany.

Rocha, Rui P. P. (2006), Building volumetric Maps with Cooperative Mobile Robots and Useful Information Sharring: A Distributed Control Aproach based on Entropy, Tese de doutorado, Faculdade de engenharia da Universidade do Porto, Porto, Portugal.

Rooker, M. N. & A. Birk (2005), 'Combining exploration and ad-hoc networking in robocup rescue', **3276**, 236–246.

Santos, C. L. R. & J. S. Sichman (1997), 'Significado e representação de organizações em sistemas multi-agentes: Uma análise preliminar'.

Schmickl, T., R. Thenius, C. Moeslinger, G. Radspieler, S. Kernbach, M. Szymanski & K. Crailsheim (2009), 'Get in touch: Cooperative decision making based on robotto-robot collisions', **18**, 133–155.

Schwartz, Rina & Sarit Kraus (1997), Negotiation on data allocation in multi-agent environments, *em* 'AAAI/IAAI'97', pp. 29–35.

Serment, L., R. Grabowski, C. Paredis & P. Khosla (2002), 'Multibots - the development of a framework and algorithms for a heterogeneous robot team', pp. 31–40.

Smart, William D. & Leslie Pack Kaelbling (2002), 'Effective reinforcement learning for mobile robots'.

Song, Dan, Kai Huebner, Ville Kyrki & Danica Kragic (2010), Learning task constraints for robot grasping using graphical models., *em* 'IROS', IEEE, pp. 1579–1585.

Souza, A. A. S., R. S. Maia & L. M. G. Gonçalves (2010), Uma proposta de um time de robï¿½s heterogï¿½neos baseados na tecnologia sun spot para mapeamento de ambientes, *em* 'Proceedings of Workshop de Robótica Aplicada e Automação ROBOCONTROL 2010', Bauru.

Surmann, Hartmut & Antonio Morales (2002), 'Scheduling tasks to a team of autonomous mobile service robots in indoor enviroments', **8**(8), 809–833.

Talay, Sanem Sariel, Tucker R. Balch & N. Erdogan (2011), 'A generic framework for distributed multirobot cooperation', *Journal of Intelligent and Robotic Systems* **63**(2), 323–358.

Talay, Sanem Sariel, Tucker R. Balch & Nadia Erdogan (2007), Incremental multi-robot task selection for resource constrained and interrelated tasks, *em* 'IROS', IEEE, pp. 2314–2319.

Toksari, M. Duran (2007), 'Ant colony optimization approach to estimate energy demand of turkey', **35**, 3984–3990.

Veloso, Manuela, Peter Stone & Michael Bowling (1999), Anticipation as a key for collaboration in a team of agents: A case study in robotic soccer, *em* 'In Proceedings of SPIE Sensor Fusion and Decentralized Control in Robotic Systems II'.

Vestli, S.J. & N. Tschichold-Gurman (1996), Mops, a system for mail distribution in office type buildings, *em* 'Advanced Mobile Robot, 1996., Proceedings of the First Euromicro Workshop on', pp. 151 –158.

Vygotsky, L. S. (1967), *Play and its roles in mental development of child*.

Vygotsky, L. S. (1978), *Mind and Society The Development of Higher Psychological Processes*, Harvard University Press, Cambridge.

Vygotsky, L. S. (1991), *A Formação Social da Mente*, Martins Fontes, São Paulo.

Vygotsky, L. S. (1993), *Pensamento e Linguagem*, Martins Fontes, São Paulo.

Wadsworth, Barry (1996), *Inteligência e Afetividade da Criança*, Enio Matheus Guazzelli, São Paulo.

Watson, John Broatus (1913), 'Psychology as the behaviorist views it', *Psychological Review* **20**(2), 158–177.

Weglarz, Jan, Joanna Józefowska, Marek Mika & Grzegorz Waligóra (2011), 'Project scheduling with finite or infinite number of activity processing modes - A survey', *European Journal of Operational Research* **208**(3), 177–205.

Wenger, E. (2000), 'Communities of practice and social learning systems', (2), 225–246.

Wenpin, Jiao (2010), 'Multi-agent cooperation via reasoning about the behavior of others', *Computational Intelligence* **26**(1), 57–83.

Wertsch, J.V. (1997), 'Vygotsky and the formation of the mind', *Cambridge* .

Zhang, Yu & Richard A. Volz (2005), Modeling cooperation by observation in agent team, *em* 'Proceedings of the IEEE International Conference on Systems, Man and Cybernetics SMC', pp. 536–541.