
بسم الله الرحمن الرحیم

دانشکده مهندسی کامپیوتر و فناوری اطلاعات

گروه هوش مصنوعی

رساله دکتری

# کاوش هدف در رباتیک جمعی با استفاده از تئوری کنترل سوپروایزری

نگارنده: فائزه میرزائی

استاد راهنما

دکتر علی‌اکبر پویان

استاد مشاور

دکتر سعیده فردوسی

شهریور ۱۳۹۸

شماره:
تاریخ:
ویرایش:

**باسمه تعالی**

مدیریت تحصیلات تکمیلی

**فرم شماره ۱۲: صورت جلسه نهایی دفاع از رساله دکتری (Ph.D)**

(ویژه دانشجویان ورودی های ۹۴ و ما قبل)

بدینوسیله گواهی می شود خانم **فائزه میرزائی** دانشجوی دکتری رشته **مهندسی کامپیوتر** به شماره دانشجویی **۹۳۰۱۴۵۵** ورودی **مهر** ماه سال **۱۳۹۳** در تاریخ **۱۳۹۸/۰۶/۲۷** از رساله **نظری**☑/ **عملی**☐ خود با عنوان : **کاوش هدف در رباتیک جمعی با استفاده از تئوری کنترل سوپروایزری**

دفاع و با اخذ نمره ..... به درجه : ..... نائل گردید.

الف) درجه عالی: نمره ۲۰-۱۹ ☑ ب) درجه بسیار خوب: نمره ۱۸/۹۹ – ۱۷ ☐

ج) درجه خوب: نمره۱۶/۹۹– ۱۵ ☐ د) غیر قابل قبول و نیاز به دفاع مجدد دارد ☐

ه) رساله نیاز به اصلاحات دارد ☐

| ردیف | هیئت داوران | نام و نام خانوادگی | مرتبه علمی | امضاء |
|---|---|---|---|---|
| ۱ | دکتر علی‌اکبر پویان | استاد راهنما | استادیار | |
| ۲ | دکتر سعیده فردوسی | استاد مشاور | استادیار | |
| ۳ | دکتر حمید حسن‌پور | استاد مدعو داخلی | استاد | |
| ۴ | دکتر امیدرضا معروضی | استاد مدعو داخلی | استادیار | |
| ۵ | دکتر عباس دیده‌بان | استاد مدعو خارجی | دانشیار | |
| ۶ | دکتر فاطمه جعفری نژاد | سرپرست ( نماینده ) تحصیلات تکمیلی دانشکده | استادیار | |

**مدیر محترم تحصیلات تکمیلی دانشگاه:**

ضمن تأیید مراتب فوق مقرر فرمائید اقدامات لازم در خصوص انجام مراحل دانش آموختگی خانم **فائزه میرزائی** بعمل آید.

**نام و نام خانوادگی رئیس دانشکده:**

**تاریخ و امضاء و مهر دانشکده:**

**تقدیم به**

# همسر مهربانم،

**که در تمام طول تحصیل همراه و همگام من بوده است.**

# پدر و مادر و پدر و مادر همسرم،

**که دعای خیرشان همیشه بدرقه راهم بوده است.**

# دخترم،

**که نور امید را به زندگیم تابید.**

# تشکر و قدردانی

پروردگارا،

نه می‌توانم موهایشان را که در راه عزت من سفید شد، سیاه کنم و نه برای دست‌های پینه بسته‌ی شان که ثمره تلاش برای افتخار من است، مرهمی دارم . پس توفیقم ده که هر لحظه شکر گزارشان باشم و ثانیه‌های عمرم را در عصای دست بودن‌شان بگذرانم.

شکر و سپاس خدا را که بزرگترین امید و یاور در لحظه لحظه زندگی است.

سپاس خدای بزرگ را که مرا یاری رساند تا بتوانم این مقطع تحصیلی را به پایان رسانده و گامی در راستای اعتلای علم بردارم.

با تقدیر و تشکر شایسته از استاد فرهیخته و فرزانه جناب آقای دکتر پویان که با نکته‌های دلاویز و گفته‌های بلند، صحیفه‌های سخن را علم‌پرور نمود و همواره راهنما و راه‌گشای نگارنده در اتمام و اکمال رساله بوده است.

از استاد صبور و مهربان، سرکار خانم دکتر فردوسی، که زحمت مشاوره این رساله را در حالی متقبل شدند که بدون مساعدت ایشان، این پروژه به نتیجه مطلوب نمی‌رسید.

از همسرم، که زحماتش در واژه نمی‌گنجد و بدون او پیمودن این راه میسر نبود.

# تعهد نامه

اینجانب **فائزه میرزائی** دانشجوی دوره دکتری رشته **مهندسی کامپیوتر، گرایش هوش مصنوعی** دانشکده **مهندسی کامپیوتر و فناوری اطلاعات** دانشگاه صنعتی شاهرود نویسنده پایان نامه **کاوش هدف در رباتیک جمعی با استفاده از تئوری کنترل سوپروایزری** تحت راهنمائی **آقای دکتر علی‌اکبر پویان** متعهد می شوم .

- تحقیقات در این پایان نامه توسط اینجانب انجام شده است و از صحت و اصالت برخوردار است.
- در استفاده از نتایج پژوهشهای محققان دیگر به مرجع مورد استفاده استناد شده است.
- مطالب مندرج در پایان نامه تاکنون توسط خود یا فرد دیگری برای دریافت هیچ نوع مدرک یا امتیازی در هیچ جا ارائه نشده است.
- کلیه حقوق معنوی این اثر متعلق به دانشگاه صنعتی شاهرود می باشد و مقالات مستخرج با نام « دانشگاه صنعتی شاهرود » و یا « Shahrood University of Technology » به چاپ خواهد رسید.
- حقوق معنوی تمام افرادی که در به دست آمدن نتایج اصلی پایان نامه تأثیرگذار بوده اند در مقالات مستخرج از پایان نامه رعایت می گردد.
- در کلیه مراحل انجام این پایان نامه، در مواردی که از موجود زنده ( یا بافتهای آنها ) استفاده شده است ضوابط و اصول اخلاقی رعایت شده است.
- در کلیه مراحل انجام این پایان نامه، در مواردی که به حوزه اطلاعات شخصی افراد دسترسی یافته یا استفاده شده است اصل رازداری، ضوابط و اصول اخلاق انسانی رعایت شده است.

**تاریخ ۱۳۹۸/۰۶/۰۱**

**امضای دانشجو**

## مالکیت نتایج و حق نشر

- کلیه حقوق معنوی این اثر و محصولات آن (مقالات مستخرج ، کتاب ، برنامه های رایانه ای ، نرم افزار ها و تجهیزات ساخته شده است) متعلق به دانشگاه صنعتی شاهرود می باشد. این مطلب باید به نحو مقتضی در تولیدات علمی مربوطه ذکر شود.
- استفاده از اطلاعات و نتایج موجود در پایان نامه بدون ذکر مرجع مجاز نمی باشد.

## چکیده

علم رباتیک جمعی (Swarm Robotics) با الهام از طبیعت و تلفیق هوش جمعی و رباتیک پا به عرصه‌ی وجود گذاشت. ویژگی‌های خاص این سیستم‌ها از جمله انعطاف در انجام وظایف، مقاومت در برابر خسارت و مقیاس‌پذیری، مزایای متعددی به همراه داشته و رباتیک جمعی را از دیگر سیستم‌های رباتیک متمایز می‌نماید. از طرفی وجود چنین ویژگی‌هایی باعث گسترش دامنه‌ی کاربردهای رباتیک جمعی در محیط‌های خطرناک، صنعتی، نظامی و امثال آن می‌گردد. در نتیجه در سال‌های اخیر محققان توجه ویژه‌ای به حوزه‌ی رباتیک جمعی داشته‌اند. به دلیل عدم وجود نرم‌افزار کنترلی مشترک برای سیستم‌های رباتیک جمعی، تحلیل، نگهداری و اعتبارسنجی این سیستم‌ها امر دشواری است. استفاده از روش‌های فرمال برای مدل‌سازی و طراحی سیستم‌های رباتیک جمعی، مشکلات گفته شده را تا حد زیادی رفع می‌نماید. ولی در روش‌های فرمال نیز تضمینی برای انطباق خواسته‌ها و مشخصات مسئله با نتیجه‌ی پیاده‌سازی وجود ندارد. رویکرد پیشنهادی، استفاده از تئوری کنترل سوپروایزری (SCT) به عنوان ابزار طراحی و مدل‌سازی سیستم است. با استفاده از این تئوری می‌توان به تولید خودکار کدهای کنترلی پرداخت و مشکل روش‌های فرمال در تضمین انطباق مشخصات سیستم را برطرف کرد.

یکی از وظایف تعریف شده در سیستم‌های رباتیک جمعی، مسئله‌ی کاوش هدف با الهام از عملیات جستجوی غذای مورچه‌ها است. این وظیفه ترکیبی از چند وظیفه‌ی دیگر چون کاوش تجمعی، حمل-ونقل تجمعی و تخصیص وظایف بوده و یک وظیفه پیچیده در حوزه‌ی رباتیک جمعی بشمار می‌رود. در رساله‌ی جاری، به منظور طراحی و پیاده‌سازی وظیفه کاوش هدف ابزار جامعی پیشنهاد گشت که رفتارها و نیازهای سیستم را در قالب مولدها دریافت کرده و کد کنترلی موردنیاز را به صورت خودکار تولید می‌کند. ابزار پیشنهاد شده نسخه‌ی قدرتمندتری از SCT را با افزودن دو مولفه‌ی احتمالات و زمان معرفی می‌کند که تئوری کنترل سوپروایزری احتمالاتی و زمان‌دار (ptSCT) نامیده می‌شود. برای مقایسه قدرت ptSCT نسبت به SCT، دو وظیفه‌ی همگام‌سازی و دوری از موانع توسط هر دو روش، طراحی و پیاده‌سازی گشت و نتایج بدست آمده مورد تحلیل قرار گرفتند. برای بررسی عملکرد وظیفه‌ی کاوش هدف طراحی شده، ۲۴۰۰ آزمایش با استفاده از ربات E-puck و در محیط شبیه‌سازی ARGoS صورت گرفته و جوانب مختلفی از جمله تاثیر تعداد اهداف، موانع و ربات‌ها و تاثیر استراحت بر عملکرد سیستم مورد بررسی قرار گرفته‌اند. نتایج بدست آمده نشان از عملکرد قابل قبول سیستم طراحی شده دارند. به این ترتیب، می‌توان از سکوی پیشنهادی برای طراحی و پیاده‌سازی سایر وظایف رباتیک جمعی نیز بهره برد.



**کلمات کلیدی:** رباتیک جمعی، هوش جمعی، تئوری کنترل سوپروایزری، کاوش هدف، روش‌های مدل‌سازی فرمال

## مقالات مستخرج از رساله

### مقالات داخلی

| ردیف | مقاله | وضعیت | تاریخ |
|---|---|---|---|
| ۱ | فائزه میرزائی، علی‌اکبر پویان، "مروری بر رباتیک جمعی و جایگاه آن در سیستم‌های چندرباته"، مجله مهندسی برق و الکترونیک ایران | چاپ شده | تابستان ۹۸ |
| ۲ | فائزه میرزائی، علی‌اکبر پویان، سعیده فردوسی، "شبیه‌سازی و پیاده‌سازی وظایف دوری از موانع و همگام‌سازی با استفاده از تئوری کنترل سوپروایزری احتمالی و زمان‌دار در رباتیک جمعی"، مجله مدل‌سازی در مهندسی | چاپ شده | زمستان ۹۷ |

### مقالات خارجی

| Row | Paper | Status | Date |
|---|---|---|---|
| 1 | Faezeh Mirzaei, Ali Akbar Pouyan, Saideh Ferdowsi, “Automatic Controller Design for Swarm Robotics using Probabilistic Timed Supervisory Control Theory (ptSCT)”, Autonomous Robots | Under Review | June 2019 |

### مقالات کنفرانسی

| ردیف | مقاله | وضعیت | تاریخ |
|---|---|---|---|
| ۱ | فائزه میرزائی، علی‌اکبر پویان، سعیده فردوسی، "پیاده‌سازی و شبیه‌سازی استراتژی تجمع ربات‌ها با استفاده از تئوری کنترل سوپروایزری در رباتیک جمعی"، کنفرانس پردازش سیگنال و سیستم‌های هوشمند | پذیرفته و ارائه شده | آذر ۹۶ |

| | | | |
|---|---|---|---|
| ۲ | فائزه میرزائی، علی‌اکبر پویان، سعیده فردوسی، "پیاده‌سازی تئوری کنترل سوپروایزری احتمالی و زمان‌دار در رباتیک جمعی"، کنفرانس ملی فناوری‌های نوین در مهندسی برق و کامپیوتر | پذیرفته و ارائه شده | دی ۹۶ |

# فهرست مطالب

# فهرست شکل‌ها

# فهرست جدول‌ها

# علائم اختصاری

| | |
|---|---|
| **ACO** | Ant Colony Optimization |
| **Bio-PEPA** | Biochemical Performance Evaluation Process Algebra |
| **con** | Controllable |
| **des** | Destination |
| **DES** | Descret Event System |
| **DFA** | Deterministic finite Automata |
| **DTD** | Document Type Definition |
| **FSM** | Finite State Machine |
| **IR** | Infra Red |
| **MIPS** | Million Instruction Per Second |
| **PFSM** | Probability Finite State Machine |
| **Prob** | Probability |
| **PSO** | Particle Swarm Optimization |
| **ptSCT** | Probabilistic and Timed Supervisory Control Theory |
| **SCT** | Supervisory Control Theory |
| **SR** | Swarm Robotics |
| **src** | Source |
| **UAV** | Unmanned Aerial Vehicle |
| **VGA** | Video Graphics Array |
| **XML** | eXtensible Markup Language |

# واژه‌نامه

**A**

| | |
|---|---|
| Actions | اعمال |
| Active | کنش گر |
| Actuators | محرک |
| Ad Hoc | تک منظوره |
| Adaptive | وفقی |
| Adaptivity | وفق پذیری |
| Additive Uniform Noise | نویز یکنواخت تجمعی |
| Aggregation | گردهمایی |
| Ant Colonies | کلونی مورچه ها |
| Area Sweeping | پوشش محیط |
| Arrangement | چیدمان |
| Artificial Pheromone | فرومون مصنوعی |
| Artificial Physics | فیزیک مجازی |
| Automatic Design | طراحی خودکار |
| Automatic Modular Design | طراحی ماژولار خودکار |
| Autonomous | خودمختار |
| Autonomy | استقلال |

**B**

| | |
|---|---|
| Biochemical Performance Evaluation Process Algebra | جبر فرایند ارزیابی کارایی زیست شیمی |
| Black Box | جعبه سیاه |
| Bottom-Up | پایین به بالا |
| Bucket Brigading | دست به دست کردن |
| Byte | بایت |

**C**

| | |
|---|---|
| Callback Function | تابع فراخوان |
| Cartography | نقشه کشی |
| Centralization | تمرکز |
| Chain Formation | آرایش زنجیره ای |
| Cognition | شناخت |
| Collectiove Exploration | کاوش تجمعی |
| Collective Behavior | رفتار تجمعی |
| Collective Decision Making | تصمیم گیری تجمعی |
| Collective Movement | حرکت تجمعی |
| Collective Reconfigurability | تغییرات پیکربندی اجتماع |
| Collective Robotic | رباتیک تجمعی |
| Collective Size | اندازه اجتماع |
| Collective Transportation | حمل و نقل تجمعی |
| Collision | تصادم |
| Communication | ارتباط |
| Communication Bandwidth | پهنای باند ارتباطی |
| Communication Range | بازه ی ارتباطی |
| Communication Topology | توپولوژی ارتباطی |
| Composition | ترکیب بندی |
| Conditions | شروط |
| Consensus Achievement | اتفاق نظر |
| Control Specification | ضابطه کنترلی |

| | |
|---|---|
| Convergence | همگرایی |
| Cooperation | تعاون |
| Coordination | هماهنگی |
| Coordination motion | حرکت هماهنگ |

**D**

| | |
|---|---|
| Deadlock | بن بست |
| Decomposition | تجزیه |
| Descrete Event System | سیستم رخداد گسسته |
| Deterministic Finite Automaton | آتاماتای متناهی معین |
| Dispersion | پراکندگی |
| Distributed Coordination | هماهنگی توزیع شده |
| Distributed Robotic | رباتیک توزیع شده |
| Document Type Definition | تعریف نوع سند |

**E**

| | |
|---|---|
| Efficiency | بهینگی |
| Enabled | فعال |

**F**

| | |
|---|---|
| Fault Tolerance | تحمل خطا |
| Finite State Machine | ماشین حالت متناهی |
| Flexibility | انعطاف |
| Fokker-Planck Equation | معادله ی فاکر پلنک |
| Foraging | کاوش هدف |
| Formal | رسمی |
| Free Behavior Model | مدل رفتار آزاد |

**G**

| | |
|---|---|
| Generator | مولد |
| Generators Player | اجرا کننده مولد |
| Global Reward | پاداش سراسری |
| Gradient Base | مبتنی بر گرادیان |
| Gripper Actuator | محرک دستگیره |
| Ground Sensor | حسگر زمین |

**H**

| | |
|---|---|
| Heterogeneous | ناهمگن |
| Homogeneous | همگن |

**I**

| | |
|---|---|
| Individuals | اعضا |
| Infrared | مادون قرمز |
| Interactions | تعاملات |

**K**

| | |
|---|---|
| Knowledge | دانش |

**L**

| | |
|---|---|
| Large number | تعداد زیاد |
| Light sensor | حسگر نور |
| Livelock | حلقه بینهایت |
| Local modular | ماژولار محلی |

**M**

| | |
|---|---|
| Macroscopic | ماکروسکوپی |
| Manual Design | طراحی دستی |
| Markov Chain | زنجیره مارکو |
| Median | میانه |
| Microscopic | میکروسکوپی |

| | |
|---|---|
| Model Checking | بررسی مدل |
| Modeling | مدل سازی |
| Monolithic | یکپارچه |
| Multi Agent Systems | ماکروسکوپی |
| Multiple Interactions | طراحی دستی |

**N**

| | |
|---|---|
| Navigation Behaviors | رفتار حرکتی و جهت یابی |
| Negative Feedback | واکنش منفی |
| Network Routing | مسیریابی شبکه |

**O**

| | |
|---|---|
| Offline | برون خط |
| Omnidirectional Camera Sensor | حسگر دوربین ۳۶۰ درجه |
| Online | برخط |
| Operational Procedure | رویه عملیاتی |
| Optical Sensor | حسگر نوری |
| Organization | سازماندهی |

**P**

| | |
|---|---|
| Parallel Composition | ترکیب موازی |
| Particle Swarm Optimization | بهینه سازی ازدحام ذرات |
| Passive | کنش پذیر |
| Path Formation | تشکیل مسیر |
| Pattern Formation | تشکیل الگو |
| Petri Net | شبکه پتری |
| Phase Synchronization | همگام سازی فازها |
| Pheromone | فرومون |
| Physics Engine | موتور فیزیک |
| Platform | سکو |
| Positive Feedback | واکنش مثبت |
| Probabilistic Finite State Machine | ماشین حالت متناهی احتمالاتی |
| Processing Ability | توانایی پردازشی |
| Programmable Logic Controller | کنترل کننده‌های منطقی قابل برنامه‌ریزی |
| Properties | ویژگی |
| Property Driven Design | طراحی مبتنی بر ویژگی |
| Proximity Sensor | حسگر مجاورت سنج |
| Push Down Automaton | آتاماتای پشته ای |

**R**

| | |
|---|---|
| Randomness | تصادفی بودن |
| Reactivity | واکنشی |
| Redundancy | فراوانی |
| Regular Language | زبان منظم |
| Reinforcement Learning | یادگیری تقویتی |
| Requirement | نیازها |
| Reusability | قابلیت استفاده مجدد |
| Robots Colony | کلونی ربات ها |
| Robustness | مقاومت |
| Roulette Wheel | چرخ رولت |

**S**

| | |
|---|---|
| Scalability | مقیاس پذیری |
| Segregation | تفکیک |

| | | | |
|---|---|---|---|
| خود اتصالی | Self-Assembling | محرک چرخ | Wheel Actuator |
| خودسازماندهی | Self-Organizing | | |
| شبکه حسگر | Sensor Networks | | |
| مدلسازی مبتنی بر حسگر | Sensor-Based Modeling | | |
| مشورت اجتماعی | Social Deliberation | | |
| ضابطه | Specification | | |
| پایداری | Stability | | |
| نشانه گذاری | Stigmergy | | |
| تئوری کنترل سوپروایزری | Supervisory Control Theory | | |
| هوش جمعی | Swarm Intelligence | | |
| رباتیک جمعی | Swarm Robotic | | |
| سنتز | Synthesis | | |

**T**

| | |
|---|---|
| اهداف | Targets |
| تخصیص وظیفه | Task Allocation |
| منطق زمانی | Temporal Logic |
| بالا به پایین | Top-Down |
| سعی و خطا | Trial and Error |
| ماشین تورینگ | Turing Machine |

**U**

| | |
|---|---|
| مافوق صوت | Ultrasound |
| هواپیمای بدون سرنشین | Unmanned Aerial Vehicle |

**V**

| | |
|---|---|
| نیروهای مجازی | Virtual Forces |

**W**

# فصل

# ۱- مقدمه

رباتیک جمعی[۱] (SR) یکی از موضوعات تحقیق مورد توجه در تکنولوژی رباتیک است که حوزه‌های هوش مصنوعی، تئوری کنترل، رباتیک، مهندسی سیستم و بیولوژی را در کنار یکدیگر قرار می‌دهد. برخی از محققان علم رباتیک، از هوش انسان و برخی دیگر از رفتار حیوانات در طبیعت الهام می‌گیرند [۱]. رباتیک جمعی نیز در تلاش است تا نه تنها ذهن حیوانات، بلکه رفتار آن‌ها در موقعیت‌های مختلف و در محیط زندگی‌شان را تقلید نماید. حیوانات و موجوداتی که به صورت اجتماعی زندگی کرده و دارای هوش جمعی[۲] هستند، جرقه‌ی اولیه‌ی رباتیک جمعی را در ذهن پژوهشگران ایجاد کرده‌اند. این موجودات رفتار خودسازمانده[۳] دارند؛ به عنوان مثال، دسته‌ی پرندگان مهاجر، کلونی مورچه‌ها[۴]، دسته‌ی ماهی‌ها، اجتماع زنبورها، موریانه‌ها، باکتری‌ها و تک سلولی‌ها چند نمونه از این موارد هستند. شکل ۱-۱ نمونه‌ای از حیواناتی که به صورت اجتماعی زندگی می‌کنند را نشان می‌دهد.

**شکل ۱-۱: نمونه‌ای از اجتماع حیوانات. بنای نشان داده شده در تصویر توسط موریانه‌ها ساخته شده است [۲, ۳].**

[۱] Swarm Robotics
[۲] Swarm Intelligence
[۳] Self-Organizing
[۴] Ant Colonies

هر یک از اعضا[1]ی اجتماعات مورد اشاره به تنهایی توانایی‌های ارتباطی[2]، محاسباتی و شناختی[3] اندکی دارند ولی در کنار هم و با استفاده از هوش جمعی قادرند ماموریت‌های بزرگی را به انجام رسانند [۴]. اجتماعات حیوانات اندازه‌های متفاوتی دارند؛ هرچه تعداد اعضا کم‌تر باشد، نقش هر یک از آن‌ها پیچیده تر بوده و توانایی بیشتری لازم دارند. کلونی مورچه‌ها می‌تواند به راحتی هزارپا یا ملخی با جسه‌ی ده برابر یک مورچه را شکست داده و یا طعمه‌ای بزرگ را حمل نماید؛ آن‌ها با استفاده از تماس، صدا و فرومونی[4] که ترشح می‌کنند با یکدیگر در ارتباط هستند [۵]. موریانه‌ها قادرند بناهای تپه‌ای بسیار پیچیده بسازند. حشرات می‌توانند به راحتی در محیطی ناشناخته و بزرگ، به صورت گروهی به جستجوی منابع غذایی بپردازند. دسته‌ی ماهی‌ها در کنار یکدیگر حرکت می‌کنند بدون اینکه به یکدیگر برخورد نمایند؛ آن‌ها قادرند شکل اجتماع خود را تغییر داده و در هنگام حمله‌ی دشمنان از خود مراقبت نمایند [۶]؛ وجود چشم ماهی‌ها در دو طرف سرشان و نشانه‌ای که بر باله‌ی آن‌هاست به هماهنگی آن‌ها کمک می‌کند. پرندگان در هنگام مهاجرت با آرایش خاصی پرواز کرده و با استفاده از حس بویایی، قطب نمای خورشیدی، محاسبات زمانی و میدان مغناطیسی، مقصد خود را می‌یابند [۷].

نکته‌ی قابل توجه درباره‌ی اجتماع حیوانات، عدم وجود رهبر و هماهنگ کننده است. مطمئنا در چنین سازمان هوشمندی، مکانیزم کشف نشده‌ای وجود دارد که وظایف اصلی را به قسمت‌های ساده تقسیم کرده تا اعضا بتوانند آن‌ها را انجام دهند و سپس نتیجه‌ی کار آن‌ها را ترکیب نموده و موجب چنین رفتار تجمعی[5] شگفت انگیزی شود. هدف از علم رباتیک جمعی و هوش جمعی نیز شناخت این مکانیزم است.

موجودات اجتماعی از حالت سراسری کلونی خود اطلاعی ندارند. دانش آن‌ها در بین افراد توزیع شده است؛ به شکلی که هر یک از اعضا نمی‌تواند بدون کمک بقیه، وظیفه‌ی اصلی را کامل کند. اعضا می‌توانند تبادل اطلاعات داشته باشند و منبع غذا یا خطر حمله را به دیگران اطلاع دهند. این اطلاعات به صورت محلی جابجا شده و دانشی از موقعیت کلی وجود ندارد. از طرفی، حشرات می‌توانند رفتار خود را با توجه به تغییراتی که مابقی اعضای دسته در محیط داده‌اند، وفق دهند. به عنوان مثال، موریانه‌ها پس از تغییر ساختار خانه‌شان، متفاوت عمل می‌کنند [۸].

---

[1] Individuals

[2] Communication

[3] Cognition

[4] Pheromone

[5] Collective Behavior

در سال‌های اخیر عبارت رباتیک جمعی به عنوان کاربرد هوش جمعی در سیستم‌های چندرباته مطرح شده است [۹]. عباراتی مانند رباتیک تجمعی[۱] [۱۰]، رباتیک توزیع شده[۲] [۱۱] و کلونی ربات‌ها[۳] [۱۲] نیز وجود دارند که برای سیستم‌های چندرباته به کار می‌روند و تفاوت‌ها و اشتراکاتی با هم دارند. ویژگی‌های خاص رباتیک جمعی در کنار تعریف آن می‌تواند تمایزی بین این حوزه و دیگر موارد ایجاد نماید [۱۳]. رباتیک جمعی، مطالعه‌ای بر چگونگی طراحی اجتماعی از ربات‌ها است؛ به گونه‌ای که با استفاده از تعاملات[۴] ربات‌ها و تعامل ربات‌ها و محیط، رفتار تجمعی داشته باشند. تعاملات در کلونی انتشار یافته و بدین ترتیب اعضای کلونی در کنار یکدیگر قادر به انجام اموری خواهند بود که دستیابی به آن برای یک ربات ممکن نیست. این رفتار تجمعی، تحت عنوان خودسازماندهی شناخته می‌شود. تئوری خودسازماندهی از علوم فیزیک و شیمی گرفته شده و مبتنی بر ترکیب چهار قانون پایه است: «واکنش مثبت[۵]»، «واکنش منفی[۶]»، «تصادفی بودن[۷]» و «تعاملات چندگانه[۸]» [۸].

تعریف دقیق‌تر رباتیک جمعی توسط مرجع [۵] ارائه شده است؛ طبق این تعریف، رباتیک جمعی، مطالعه‌ی چگونگی طراحی ربات‌های ساده و قابل تعاملات فیزیکی است به گونه‌ای که با استفاده از تعاملات بین رباتی و بین ربات و محیط بتوانند رفتار تجمعی داشته باشند. در کنار این تعریف، مشخصاتی نیز برای رباتیک جمعی مطرح شده است که در ادامه تشریح خواهند شد.

رباتیک جمعی زیر مجموعه‌ی خاصی از رباتیک تجمعی است و به طور کلی به دلیل وجود بیش از یک ربات در سیستم، نوعی از سیستم‌های چندرباته محسوب می‌شود. در ادامه به معرفی سلسه مراتب وراثت در سیستم‌های چندرباته به عنوان پدر علم رباتیک جمعی پرداخته شده است.

## ۱-۱- سیستم‌های چندرباته

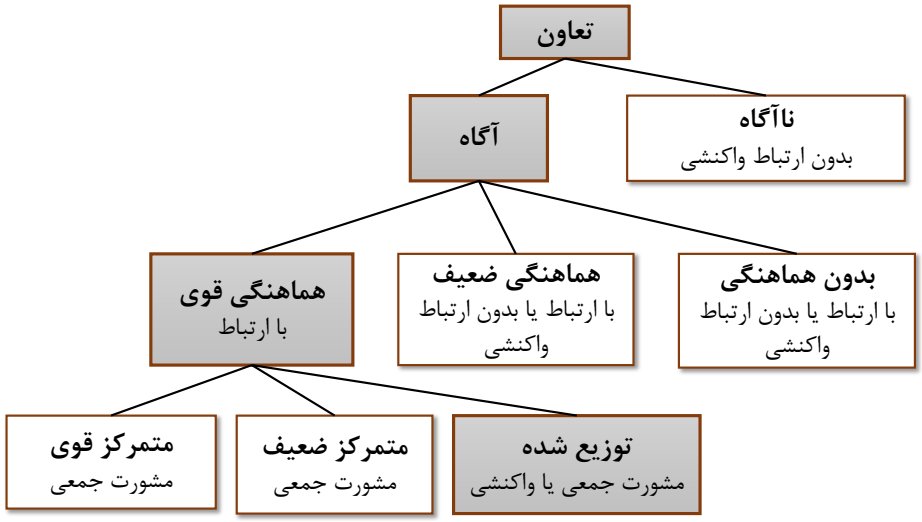
سیستم‌های چندرباته به دلیل توانایی اندک ربات‌های منفرد در پردازش اطلاعات و موارد دیگری که ربات‌ها به تنهایی قادر به انجام آن‌ها نیستند، پا به عرصه‌ی وجود گذاشتند. ربات‌های موجود در سیستم باید با یکدیگر همکاری و تعاون داشته باشند تا بتوانند وظایف بزرگ را مدیریت نمایند. سیستم‌های

---

[۱] Collective Robotic
[۲] Distributed Robotic
[۳] Robots Colony
[۴] Interactions
[۵] Positive Feedback
[۶] Negative Feedback
[۷] Randomness
[۸] Multiple Interactions

چندرباته قادرند وظایفی چون ماموریت‌های جستجو و نجات، شکار، نظارت بر محیط و کاوش فضا را با موفقیت و بسیار بهتر از قبل انجام دهند.

سیستم‌های چندرباته را می‌توان به روش‌های مختلفی طبقه‌بندی نمود. برای نمونه، طبقه‌بندی مطرح شده در [۱۴] از چهار سطح تشکیل شده است: «تعاون[۱]»، «دانش[۲]»، «هماهنگی[۳]» و «سازمان‌دهی[۴]». علاوه بر آن، به دو ویژگی «ارتباطات» و «ترکیب‌بندی[۵]» نیز اشاره شده است که به صورت عمود بر طبقه‌بندی اصلی قابل تعریف هستند. در سطح اول طبقه‌بندی، «تعاون» قرار دارد. ربات‌ها در انجام وظایف سراسری به شکلی همکاری می‌نمایند که آن وظیفه توسط یک ربات منفرد قابل انجام نباشد یا با انجام کار گروهی، کارایی سیستم ارتقا یابد. در ادامه، تنها به سیستم‌های چندرباته متعاون پرداخته شده است؛ در این سیستم‌ها یک هدف سراسری برای کل اعضا وجود دارد. سیستم‌هایی که ربات‌های آن در یک محیط مشترک به انجام امور متفاوت می‌پردازند (غیر متعاون)، در این بخش مورد بحث قرار نگرفته‌اند. شکل ۱-۲ طبقه‌بندی مطرح شده در [۱۴] را نشان می‌دهد.




**شکل ۱-۲: طبقه‌بندی سیستم‌های چندرباته. ویژگی‌های رباتیک جمعی در این طبقه‌بندی با رنگ خاکستری متمایز شده‌اند [۱۴].**

مهم‌ترین ویژگی در یک سیستم متعاون، اطلاع از وجود دیگر هم‌تیمی‌ها است. در سیستم‌های ناآگاه، ربات‌ها از وجود یا عدم وجود دیگر ربات‌ها اطلاعی ندارند. این ویژگی، سطح دوم طبقه‌بندی با عنوان «دانش» را ایجاد می‌نماید. تعاون در سیستم‌های ناآگاه ضعیف‌ترین نوع تعاون است. به عنوان مثال در

---

[۱] Cooperation
[۲] Knowledge
[۳] Coordination
[۴] Organization
[۵] Composition

عملیات هل دادن یک جعبه، ربات‌ها می‌توانند در کنار یکدیگر بدون اطلاع از حضور دیگر ربات‌ها به همکاری و انجام یک هدف مشترک بپردازند.

سطح سوم طبقه‌بندی، «هماهنگی» نام دارد. در سیستم‌های هماهنگ، تعاون ربات‌ها به گونه‌ای است که هر ربات برای انجام هر عمل، عملکرد دیگر ربات‌ها را در نظر می‌گیرد. بنابراین عملیات به شکل منسجم و با کارایی بالا به اتمام می‌رسد. هماهنگی الزاما از ویژگی سیستم‌های متعاون نیست؛ در واقع سیستم‌های هماهنگی وجود دارند که متعاون نیستند. به عنوان مثال هماهنگی در بین ربات‌های صنعتی که ابزار آلات را به اشتراک می‌گذارند لازم است؛ با اینکه ربات‌ها متفاوتند و اهداف مشترکی ندارند. هماهنگی در این سیستم‌ها از بروز تداخل جلوگیری می‌کند. واضح است که سیستم‌های ناآگاه نمی‌توانند هماهنگی داشته باشند؛ زیرا ربات‌ها از وجود یکدیگر اطلاعی ندارند.

هماهنگی به دو نوع قوی و ضعیف قابل تقسیم است؛ هماهنگی قوی بر اساس قوانین از پیش تعیین شده یا یادگرفته شده است. این قوانین و پروتکل‌ها چگونگی تعامل دو یا بیشتر ربات با یکدیگر را تعیین می‌نمایند. هماهنگی ضعیف به پروتکل وابسته نیست و به خطای ارتباطی مقاوم‌تر است. ولی توانایی‌های سازمانی کم‌تری نسبت به هماهنگی قوی دارد. هر چه محیط پویاتر و هدف پیچیده‌تر باشد، هماهنگی قوی برای رسیدن به هدف کارآمدتر است. هماهنگی همیشه یک ویژگی لازم در سیستم‌های متعاون نیست. با این حال انعطاف سیستم را با فراهم کردن امکان استفاده بهتر از منابع مشترک، افزایش می‌دهد. مزیت سیستم‌های ناهماهنگ، طراحی ساده‌تر آن‌ها است که منجر به ریسک خطای کم‌تر می‌شود؛ ولی این ویژگی به دلیل تداخل‌های ممکن در عملکرد ربات‌ها باعث اتلاف منابع خواهد شد.

سازمان‌دهی نام چهارمین سطح طبقه‌بندی است که وجود و عدم وجود تمرکز[۱] در سیستم را تعیین می‌نماید. در سیستم‌های توزیع شده، ربات‌ها کاملا خودمختار[۲] بوده و رهبر وجود ندارد. تمرکز می‌تواند دارای ساختار سلسله مراتبی باشد و هر رهبر توسط رهبری دیگر مدیریت شود. در حالت تمرکز قوی، تصمیمات توسط یک رهبر گرفته می‌شود و این رهبر تا پایان عملیات تعویض نمی‌شود. تمرکز ضعیف هنگامی رخ می‌دهد که امکان تعویض رهبر در حین عملیات وجود داشته باشد و بر اساس تغییرات محیط یا شکست رهبر قبل، نقش رهبری به ربات دیگر انتساب یابد. مشکل تمرکز قوی وابستگی شدید به ارتباطات است؛ اگر خطایی رخ دهد یا رهبر دچار مشکل شود، کل سیستم شکست می‌خورد. مزیت سیستم‌های متمرکز امکان وجود رهبری توانا و مناسب است که بتواند به راحتی داده‌ها را پردازش کرده و تصمیات را اتخاد نماید. در این سیستم‌ها، بقیه‌ی اعضای تیم وظایف ساده‌تری را انجام می‌دهند.

---

[۱] Centralization

[۲] Autonomous

سیستم‌های توزیع شده مشکلات بالا را ندارند، زیرا هر ربات در تصمیم‌گیری خودمختار است؛ ولی برای تشخیص هماهنگی بین اعضا پیچیدگی بیشتری لازم است.

در این طبقه‌بندی دو فاکتور دیگر نیز وجود دارد که به صورت عمود بر نمودار قابل تفسیر است: «ارتباطات» و «ترکیب‌بندی» سیستم. تعاون بین ربات‌ها معمولا توسط مکانیزم ارتباطی صورت می‌گیرد که اجازه می‌دهد ربات‌ها تبادل اطلاعات داشته باشند. در طبقه‌بندی معرفی شده در [۱۴]، هماهنگی قوی حتما باید ارتباط داشته باشد. در دسته‌ی هماهنگی ضعیف و یا ناهماهنگ، تبادل اطلاعات می‌تواند وجود داشته یا نداشته باشد. سیستم‌های ناآگاه ارتباط ندارند. بسته به روش تبادل اطلاعات، دو نوع ارتباط وجود دارد: ارتباط مستقیم و غیرمستقیم. در ارتباط مستقیم، با استفاده از سخت افزار، سیگنالی قابل فهم به دیگر ربات‌ها ارسال می‌شود. در ارتباط غیرمستقیم که به آن نشانه‌گذاری[۱] نیز گفته می‌شود، نشانه یا تاثیری بر محیط گذاشته شده و به این ترتیب بقیه‌ی ربات‌ها نشانه را تفسیر می‌نمایند؛ همانند فرومونی که مورچه‌ها در محیط برجا می‌گذارند.

فاکتور «ترکیب‌بندی» خود به دو نوع «همگن[۲]» و «ناهمگن[۳]» تقسیم می‌شود. در ساختار همگن تمامی ربات‌های سیستم از نظر سخت‌افزاری و نرم‌افزار کنترلی، یکسان هستند. اگر سیستم همگنِ قطعی باشد، ربات‌های مختلف در ازای دریافت ورودی یکسان، خروجی یکسان تولید می‌کنند. ربات‌ها تنها در صورتی که موقعیت‌شان در محیط متفاوت باشد و یا سیستم غیرقطعی باشد، متفاوت عمل می‌نمایند. در سیستم‌های ناهمگن، سخت‌افزار یا نرم‌افزار کنترلی ربات‌ها متفاوت است. در یک سیستم همگن، به دلیل رفتار یکسان ربات‌ها در صورتی که رباتی با مشکل مواجه شود، وظیفه‌ی او توسط یک ربات دیگر مدیریت می‌شود؛ در نتیجه تحمل‌پذیری خطا و مقاومت این سیستم‌ها بالاتر است. ضعف سیستم‌های همگن در «وفق‌پذیری[۴]» پایین آن‌هاست؛ زیرا تفاوتی بین ربات‌ها وجود ندارد. سیستم‌های ناهمگن در موقعیت‌های جدید و محیط‌های پویا راحت‌تر وفق می‌یابند. سیستم‌های ناهمگن در محیط‌های پویا و شرایط پیش‌بینی نشده، تحمل‌پذیری خطای بالاتری دارند؛ زیرا در صورتی که رباتی با مشکل مواجه شود، ربات دیگری وظیفه‌ی او را انجام می‌دهد. توجه به این نکته ضروری است که در سیستم‌های ناهمگن نیز ربات‌های متفاوت می‌توانند وظیفه‌ی مشابه‌ای را انجام دهند.

---

[۱] Stigmergy

[۲] Homogeneous

[۳] Heterogeneous

[۴] Adaptivity

دو ویژگی دیگر سیستم‌های رباتیک ، «مشورت اجتماعی[۱]» و «واکنشی[۲]» نام دارند. در رفتار مشورت اجتماعی، به تیم اجازه داده می‌شود تا با تغییرات محیط هماهنگ شوند. وظایف تیم توسط استراتژی-هایی مشخص می‌شود تا از تمامی منابع در دسترس برای رسیدن به هدف استفاده گردد. در رفتار واکنشی، هر یک از اعضا با استفاده از استراتژی منفرد خود با تغییرات محیط هماهنگ شده و سعی بر اتمام وظیفه‌ی خود دارد. در طبقه‌بندی ارائه شده در مرجع [۱۴]، همه‌ی سیستم‌هایی که هماهنگ قوی نیستند، به صورت واکنشی عمل می‌نمایند. ممکن است هر دو روش در سیستم‌های هماهنگ قوی استفاده شود. در سیستم‌های توزیع شده، می‌توان به صورت واکنشی یا مشورت اجتماعی عمل نمود. رهبر گروه در دو حالت تمرکز قوی و ضعیف، استراتژی را تغییر داده و رفتار سیستم را تعیین می‌کند.

طبقه‌بندی دیگری توسط [۱۵] مطرح شده است (جدول ۱-۱). در این طبقه‌بندی سیستم‌های چندرباته با توجه به ویژگی‌هایی همچون «اندازه‌ی اجتماع[۳]»، «بازه‌ی ارتباطی[۴]»، «توپولوژی ارتباطی[۵]»، «پهنای باند ارتباطی[۶]»، «تغییرات پیکربندی اجتماع[۷]»، «توانایی پردازشی[۸]» و «ترکیب‌بندی اجتماع» دسته-بندی می‌شوند. در ادامه هر یک از این ویژگی‌ها مختصرا توضیح داده می‌شود.

**جدول ۱-۱: طبقه‌بندی سیستم‌های چندرباته [۱۵]. ویژگی‌های رباتیک جمعی در طبقه‌بندی پیشنهادی با رنگ خاکستری مشخص شده‌اند.**

| نام ویژگی | انواع حالات | | | |
|---|---|---|---|---|
| اندازه‌ی اجتماع | SIZE-ALONE | SIZE-PAIR | SIZE-LIM | SIZE-INF |
| بازه ارتباطی | COM-NONE | COM-NEAR | COM-INF | - |
| توپولوژی ارتباطی | TOP-BROAD | TOP-ADD | TOP-TREE | TOP-GRAPH |
| پهنای باند ارتباطی | BAND-INF | BAND-MOTION | BAND-LOW | BAND-ZERO |
| پیکربندی مجدد اجتماع | ARR-STATIC | ARR-COMM | ARR-DYN | - |
| توانایی محاسباتی | PROC-SUM | PROC-FSA | PROC-PDA | PROC-TME |
| ترکیب‌بندی اجتماع | CMP-IDENT | CMP-HOM | CMP-HET | - |

[۱] Social Deliberation
[۲] Reactivity
[۳] Collective Size
[۴] Communication Range
[۵] Communication Topology
[۶] Communication Bandwidth
[۷] Collective Reconfigurability
[۸] Processing Ability

منظور از اندازه‌ی اجتماع، تعداد ربات‌های موجود در سیستم است که می‌تواند چهار حالت مختلف داشته باشد؛ SIZE-ALONE به معنی یک ربات منفرد است. SIZE-PAIR به دو ربات اشاره داشته و در SIZE-LIM تعداد ربات‌ها در محدوده‌ی کوچکی می‌تواند متفاوت باشد. SIZE-INF بیانگر تعداد بی‌نهایت ربات است.

بازه‌ی ارتباطی، بیشینه‌ی فاصله‌ی دو ربات که ارتباط بین آن‌ها ممکن باشد را نشان می‌دهد. این ویژگی سه حالت دارد؛ COM-NONE به معنی عدم اجازه‌ی ارتباط مستقیم است. در COM-NEAR هر ربات فقط با ربات‌های نزدیک خود ارتباط دارد. در COM-INF ربات می‌تواند بدون محدودیت با هر ربات دیگری ارتباط داشته باشد. فاصله در اینجا به صورت اقلیدسی یا بر اساس توپولوژی محاسبه می‌گردد.

توپولوژی ارتباطی نیز به چهار دسته تقسیم می‌شود؛ در TOP-BROAD ربات‌ها نمی‌توانند به ربات خاصی پیام دهند و پیام‌ها برای همه‌ی ربات‌ها ارسال می‌شوند. در TOP-ADD هر ربات می‌تواند با استفاده از نام یا آدرس ربات دیگر به او پیام دهد. در TOP-TREE ربات‌ها در یک ساختار درختی قرار داشته و ممکن است فقط بتوانند به پدران یا فرزندانشان پیام دهند. این نوع از توپولوژی‌ها در سیستم-های نظارتی و کنترلی کاربرد دارد. در TOP-GRAPH ربات‌ها به شکل یک گراف با یکدیگر در ارتباط هستند. این روش نسبت به توپولوژی درختی مقاوم‌تر است زیرا وجود لینک‌های بیشتر منجر به قطع نشدن راحت رابطه‌ی بین اعضای اجتماع می‌گردد. استراتژی‌های ارتباطی مانند درخت یا آدرس به شکست ربات‌ها حساس هستند؛ شکست یک ربات، باعث توقف کار ربات دیگر در سمت دیگر ارتباط می‌شود.

پهنای باند ارتباط می‌تواند یکی از چهار نوع زیر باشد؛ در BAND-INF ارتباط آزاد است؛ در این حالت، پهنای باند به اندازه‌ای بزرگ است که محدودیتی در ارسال اطلاعات وجود ندارد. در BAND-MOTION هزینه‌ی ارتباطات با هزینه‌ی جابجایی ربات بین مکان‌های مختلف یکسان است؛ این مسئله را می‌توان الهامی از رقص زنبورهای عسل دانست؛ رقص زنبور توسط زنبورهای نزدیکش دیده و تفسیر می‌شود. در BAND-LOW هزینه‌ی ارتباط بسیار زیاد بوده و از هزینه‌ی جابجایی ربات‌ها بیشتر است. در این شرایط بهتر است ربات‌ها مستقل باشند. استفاده از BAND-LOW هنگامی قابل پذیرش است که هدف از چندرباته بودن سیستم، افزایش فراوانی ربات‌ها باشد نه افزایش بهینگی[1] سیستم. در BAND-ZERO ارتباط مجاز نیست و ربات‌ها نمی‌توانند یکدیگر را احساس کنند؛ در طبقه‌بندی پیشین، این سیستم‌ها «ناآگاه» نامیده شدند.

[1] Efficiency

به نرخی که اجتماع بتواند خود را بازسازماندهی کند، پیکربندی مجدد اجتماع گویند. در واقع نرخی که هر عضو بتواند نسبت به دیگران حرکت کند. به عنوان مثال، زنبورها می‌توانند به سرعت با توجه به حرکت دیگری، حرکت خود را تغییر دهند ولی خودروها در یک بزرگراه نمی‌توانند. در ARR-STATIC چیدمان[۱] ایستا و توپولوژی ثابت است. در ARR-COMM چیدمان مجدد به صورت هماهنگ شده صورت می‌گیرد و در واقع ربات تنها با اعضایی که ارتباط دارد، تغییر چیدمان انجام می‌دهد. ARR-DYN به چیدمان پویا اشاره دارد که در آن پیوند اعضای اجتماع می‌تواند تغییر کند.

در ویژگی توانایی پردازش هر یک از اعضا، به هر یک از ربات‌ها مدل محاسباتی بخصوصی نسبت داده می‌شود. می‌توان برای هر ربات یک مدل محاسباتی ساده‌تر و ضعیف‌تر از ماشین تورینگ[۲] استفاده نمود. با این حال، مدل ترکیبی کل اجتماع ممکن است قدرتمند و پیچیده شود. برای سادگی بیشتر تنها مدل‌های ترتیبی ساده در فضای گسسته عنوان شده‌اند. PROC-SUM واحد ساده‌ای است که برای شبیه‌سازی شبکه عصبی استفاده می‌شود ولی ممکن است به علت سادگی زیاد برای مدل کردن یک ربات مناسب نباشد. در PROC-FSA از ماشین حالت متناهی برای مدل‌کردن محاسبات هر ربات استفاده می‌شود. PROC-PDA نیز از آتاماتای پشته‌ای[۳] برای مدل کردن بهره می‌برد. PROC-TME معادل ماشین تورینگ است. اغلب سیستم‌ها از این نوع مدل محاسباتی استفاده می‌نمایند.

ترکیب‌بندی اجتماع نیز پیش‌تر توضیح داده شد. در CMP-IDENT اعضای اجتماع از نظر سخت‌افزاری و نرم‌افزاری همگن هستند. در CMP-HOM اعضا از نظر ویژگی‌های فیزیکی همگن می‌باشند. در CMP-HET ربات‌ها از نظر فیزیکی یکسان نیستند و در نتیجه رفتار متفاوتی خواهند داشت.

## ۱-۲- رباتیک جمعی

در سال ۲۰۰۰ برای اولین بار پروژه‌ی رباتیک جمعی ایجاد شد [۱۶]. مارکو دوریگو[۴] مخترع روش کلونی مورچه‌ها (ACO) این پروژه را با هدف مطالعه‌ی روش‌های جدید طراحی و پیاده‌سازی خودسازماندهی و خود اتصالی[۵] اشیا تعریف نمود. دوریگو در [۱۷] علاوه بر ارائه‌ی تعریفی از رباتیک جمعی، بیان نمود که اجتماع ربات‌ها باید به گونه‌ای باشد که ربات‌های بسیار کوچک و ارزان قیمت در کنار یکدیگر وظایفی را انجام دهند که یک یا چند ربات پیچیده قادر به انجام آن‌ها باشند. از طرفی،

---

۱ Arrangement

۲ Turing Machine

۳ Push Down Automaton

۴ Marco Dorigo

۵ Self-assembling

طراحی ربات‌ها (از دو جنبه‌ی فیزیکی و رفتارهای کنترلی) باید به شیوه‌ای باشد که رفتار تجمعی از تعاملات بین ربات‌ها و ربات‌ها و محیط ایجاد گردد [۱۸].

دوریگو پس از آن قیودی را به منظور تفکیک رباتیک جمعی از دیگر سیستم‌های چندرباته مطرح نمود [۱۹]. در این بخش به تعدادی از این قیدها از قبیل «استقلال[۱]»، «تعداد زیاد[۲]»، «توانایی محدود»، «مقیاس‌پذیری[۳]»، «مقاومت[۴]»، «انعطاف[۵]»، «هماهنگی توزیع‌شده[۶]» و «همگن بودن» اشاره می‌شود.

- **استقلال**: ربات‌ها در این سیستم باید بتوانند به صورت فیزیکی با محیط تعامل داشته باشند و بر آن اثر بگذارند. به عنوان مثال، شبکه‌ی حسگرها[۷]، قدرت تعامل با محیط ندارند؛ بنابراین این سیستم‌ها، رباتیک جمعی به حساب نمی‌آیند. در این سیستم‌ها رهبر کنترل کننده وجود ندارد.
- **تعداد زیاد**: تعداد ربات‌های موجود در سیستم باید زیاد باشد. همان‌گونه که در اجتماعات حیواناتی مانند مورچه یا زنبور شاهد چند صد تا چند هزار حشره در دسته هستیم. ممکن است بسته به نیاز مسئله، چند گروه از ربات‌ها وجود داشته و هر گروه شامل چندین ربات باشد. در رباتیک جمعی تعداد گروه‌ها کم و اعضای هر گروه زیاد است.
- **توانایی محدود**: توانایی‌های هر یک از ربات‌ها باید کم باشد؛ به گونه‌ای که توانایی انجام وظایف بزرگ را به تنهایی نداشته باشند. این مسئله به کاهش هزینه‌ی ربات نیز کمک می‌کند که از قیود اصلی رباتیک جمعی است.
- **مقیاس‌پذیری**: سیستم باید با افزایش تعداد ربات‌ها به درستی عمل کند و حتی در شرایط مناسب، کارایی کل مجموعه ارتقا یابد.
- **انعطاف**: ربات‌ها توانایی تولید راه‌حل‌های ماژوله برای وظایف مختلف را دارند. به این معنی که برای انجام عملیات مختلف، نیاز به تنظیمات سخت‌افزاری زیاد یا تغییر سکو[۸]های نرم‌افزاری نباشد. مورچه‌ها با وجود ساختار ثابت در شرایط شکار، یافتن غذا و تشکیل زنجیره، متفاوت عمل می‌کنند؛ در حالت شکار، با ایجاد ارتشی قوی و همکاری، شکار بزرگ را به لانه می‌برند

---

۱ Autonomy
۲ Large Number
۳ Scalability
۴ Robustness
۵ Flexibility
۶ Distributed Coordination
۷ Sensor Networks
۸ Platform

و در یافتن غذا، هر مورچه به صورت منفرد به جستجوی غذا می‌رود و با استفاده از قرار دادن فرومون روی محیط با یکدیگر هماهنگ می‌شوند.

- **مقاومت**: سیستم ربات‌ها در شرایطی که تعدادی از ربات‌ها دچار خسارت شوند یا محیط آشوبی شود، باید به کار خود ادامه دهد؛ حتی اگر لازم باشد مقداری از کارایی کم شود. در واقع با وجود فراوانی[1] در این سیستم‌ها، خسارت یک ربات، توسط ربات‌های دیگر جبران می‌شود.
- **هماهنگی توزیع‌شده**: در سیستم رباتیک جمعی، هماهنگی بین ربات‌ها توزیع شده است و هر ربات حسگرها و ارتباطات محدود و محلی دارد.
- **همگن بودن**: در رباتیک جمعی، تعداد محدودی گروه وجود دارد که ربات‌های هر گروه از نظر فیزیکی و نرم‌افزاری مشابه‌اند و نقش یکسان در عملیات دارند. ربات‌های گروه‌های مختلف ممکن است از نظر رفتار یا نرم افزار کنترلی متفاوت باشند.

در بخش پیش دو نوع طبقه‌بندی برای سیستم‌های چندرباته معرفی شد. در این‌جا موقعیت رباتیک جمعی در این دو طبقه‌بندی بیان می‌شود. ویژگی‌های رباتیک جمعی با رنگ خاکستری در شکل ۱-۲ و جدول ۱-۱ متمایز شده‌اند. رباتیک جمعی یک سیستم متعاون و آگاه (ربات‌ها از وجود یکدیگر مطلع هستند) با هماهنگی قوی بین اعضا است که به صورت توزیع شده و بدون رهبری عمل می‌نمایند. اندازه‌ی اجتماع در رباتیک جمعی می‌تواند در محدوده‌ی بسیار بزرگی متغیر باشد و این مقیاس‌پذیری این سیستم‌ها را نشان می‌دهد. ارتباط در این سیستم‌ها محلی است یعنی ربات‌ها تنها توانایی ارتباط با ربات‌هایی که در نزدیکی آن‌هاست را دارند. توپولوژی ارتباطی بین ربات‌ها از نوع گراف بوده و هزینه‌ی ارتباطی بین ربات‌ها معادل هزینه‌ی جابجایی آن‌ها است. ربات‌ها به طور کلی قادرند چیدمان خود با ربات‌هایی که با آن‌ها در ارتباط هستند را تغییر دهند؛ این تغییر چیدمان می‌تواند از نوع پویا نیز باشد. مدل محاسباتی ربات‌ها از نوع ماشین تورینگ است. ترکیب‌بندی ربات‌ها نیز به صورت همگن می‌باشد. علت انتخاب نکردن حالت CMP-IDENT، امکان وجود گروه‌های مختلف با نرم‌افزار کنترلی و رفتارهای متفاوت است. ولی در هر گروه ربات‌ها به صورت CMP-IDENT بوده و از نظر سخت‌افزاری و نرم‌افزاری کاملا یکسان می‌باشند.

## ۱-۳- مقایسه رباتیک جمعی و سیستم‌های تک‌رباته

سیستم‌های تک‌رباته با الهام از انسان و رباتیک جمعی با الهام از حیوانات اجتماعی ایجاد شده‌اند. شبیه‌سازی تعاملات انسان با استفاده از تکنولوژی‌های موجود، سخت‌تر از شبیه‌سازی رفتار حیوانات

[1] Redundancy

است. بنابراین امید است که رباتیک جمعی بتواند در انجام وظایف پیچیده و با ابعاد وسیع، بهتر عمل نماید. به منظور اتمام یک وظیفه‌ی مشخص، یک ربات منفرد باید با ساختار و ماژول‌های کنترلی پیچیده طراحی شود و در نتیجه هزینه‌ی طراحی، ساخت و نگهداری بالایی خواهد داشت. از طرفی، سیستم تک‌رباته آسیب‌پذیری بالایی دارد؛ شکستن بخشی از ربات منجر به توقف کار ربات خواهد شد؛ و این دست اتفاقات قابل پیش‌بینی نیز نیستند. رباتیک جمعی با استفاده از هماهنگی‌های درون گروهی می‌تواند وظایف یک سیستم تک‌رباته را بر عهده گرفته و هزینه‌ی ساخت و نگهداری کمتری داشته باشد. از دیگر مزایای رباتیک جمعی، موازی بودن فرایندها است. وجود تعداد زیادی ربات درون این سیستم کمک می‌کند تا در یک محیط بزرگ بتوان اهداف[۱] زیادی را در زمان کم جستجو نمود.

خاصیت مقیاس‌پذیری رباتیک جمعی باعث می‌شود که در کاربردهای دنیای واقعی، بدون نیاز به تغییر در سخت افزار و نرم‌افزار و بدون کاهش کارایی سیستم بتوان تعداد ربات‌ها را کاهش یا افزایش داد. مزیت دیگر آن خاصیت انعطاف‌پذیری است؛ در رباتیک جمعی نیازی به تغییرات تنظیمات سخت‌افزاری برای انجام وظایف مختلف نیست.

ویژگی «مقاومت» از دیگر مزایای استفاده از رباتیک جمعی در مقایسه با یک ربات است. به جای صرف هزینه‌ی بالا برای تهیه‌ی یک ربات پیچیده که با شکستش کل سیستم دچار خسارت می‌شود؛ تعدادی ربات ارزان قیمت خریداری شده که با شکست تعدادی از آن‌ها، بقیه‌ی ربات‌ها به کار خود ادامه می‌دهند و هدف اصلی سیستم را دنبال می‌کنند. این ویژگی در ماموریت‌های خطرناک، ارزش زیادی دارد.

به دلیل کوچک و ساده بودن ربات‌ها در رباتیک جمعی، مصرف انرژی آن‌ها در مقایسه با ربات‌های منفرد بسیار اندک است. طول عمر باتری‌های رباتیک جمعی با وجود اندازه‌ی کوچک‌تر، بیش‌تر است؛ بنابراین در محیط‌هایی که امکان شارژ ربات‌ها نیست و سیم‌کشی الکتریکی وجود ندارد، رباتیک جمعی بسیار مفیدتر از ربات منفرد است.

کنترل هواپیماهای بدون سرنشین[۲] ، عملیات نجات پس از وقایع طبیعی، استخراج معدن، کاربردهای نظامی و حمل و نقل گروهی، نمونه‌ای از کاربردهای رباتیک جمعی است. ربات‌ها در این ماموریت‌ها با همکاری و هماهنگی می‌توانند وظایف را به خوبی به پایان برسانند. در حالی یک ربات منفرد توانایی انجام این عملیات‌ها را ندارد [۲۰].

به طور کلی افزایش کارایی با استفاده از موازی‌سازی، توانایی انجام عملیات و وظایف پیچیده‌تر، حسگرهای توزیع شده، عملیات توزیع شده و تحمل‌پذیری خطا از مزایای سیستم‌های چندرباته نسبت

---

[۱] Targets

[۲] Unmanned Aerial Vehicle(UAV)

به سیستم‌های رباتیک منفرد است. تداخل ربات‌ها با یکدیگر، عدم اطمینان از عملکرد دیگر ربات‌ها و هزینه‌ی بالای تهیه‌ی سیستم از جمله معایب سیستم‌های چندرباته است. در صورتی که ربات‌ها از عملکرد دیگر ربات‌ها اطلاعی نداشته باشند، ممکن است به جای همکاری به رقابت بپردازند.

## ۱-۴- مقایسه رباتیک جمعی و سیستم‌های چندرباته

سیستم‌هایی مانند شبکه‌ی حسگرها و یا سیستم‌های چند عامله[۱] با الهام از طبیعت و حیوانات ممکن است با رباتیک جمعی اشتراکاتی داشته باشند؛ ولی ویژگی‌های خاص رباتیک جمعی تمایز محسوسی بین این سیستم و بقیه‌ی سیستم‌های چندرباته ایجاد می‌کند. علاوه بر ویژگی‌هایی که پیش‌تر توضیح داده شد، جدول ۱-۲ سیستم رباتیک جمعی را با سیستم‌های چندرباته، شبکه حسگرها و سیستم‌های چند عامله مقایسه می‌نماید [۲۰]. استقلال موجود در رباتیک جمعی و توانایی تعامل آن‌ها با محیط برگرفته از اجتماعات حیوانات است. مسئله‌ای که در شبکه‌ی حسگرها وجود ندارد. ارتباطات درون رباتیک جمعی، محلی و محدود است. ارتباطات سراسری باعث کاهش مقیاس‌پذیری و انعطاف گشته و با افزایش تعداد ربات‌ها، هزینه‌ی ارتباطات به صورت نمایی افزایش می‌یابد. مراجع [۲۱, ۲۲] دو نمونه از سیستم‌های چندرباته هستند که دارای تفاوت‌های ذکر شده با رباتیک جمعی می‌باشند. در سیستم‌های توسعه یافته توسط این مراجع، جمعیت ربات‌ها کم و توان پردازشی ربات‌ها بیش از حد قابل قبول در رباتیک جمعی است. از طرفی مراجع [۲۳, ۲۴] فاصله بسیار نزدیکی با رباتیک جمعی دارد. این دو مرجع که به پیاده‌سازی کنترل تشکیل گروه ربات‌ها پرداخته‌اند، تنها قید جمعیت ربات‌ها را رعایت نکرده‌اند.

**جدول ۱-۲: مقایسه رباتیک جمعی و دیگر سیستم‌ها**

| | رباتیک جمعی | سیستم‌های چندرباته | شبکه حسگرها | سیستم‌های چند عامله |
|---|---|---|---|---|
| اندازه جمعیت | متفاوت در محدوده بزرگ | کوچک | ثابت | متفاوت در محدوده کوچک |
| کنترل | غیر متمرکز و مستقل | متمرکز یا از راه دور | متمرکز یا از راه دور | متمرکز یا سلسله مراتبی یا شبکه |
| همگن/ناهمگن | همگن | معمولا ناهمگن | همگن | همگن یا ناهمگن |
| انعطاف | بالا | پایین | پایین | متوسط |
| مقیاس‌پذیری | بالا | پایین | متوسط | متوسط |

[۱] Multi Agent Systems

| محیط | ناشناخته | شناخته یا ناشناخته | شناخته | شناخته |
|---|---|---|---|---|
| حرکت | بله | بله | نه | به ندرت |
| کاربردهای معمول | عملیات خطرناک، عملیات نظامی، عملیات جستجو و نجات | حمل و نقل، حسگر، فوتبال ربات‌ها | نظارت، مراقبت پزشکی، محافظت از محیط | مدیریت منابع شبکه، کنترل توزیع شده |

## ۱-۵- معرفی ربات‌ها و محیط‌های شبیه‌سازی

استفاده از ربات‌های فیزیکی در تحقیقات امر دشوار و بعضاً پرهزینه‌ای است. به همین دلیل محیط‌های شبیه‌سازی جهت آزمون ساختار و الگوریتم‌های مورد استفاده طراحی شده‌اند. آزمون در محیط‌های شبیه‌سازی ارزان‌تر و سریع‌تر از محیط‌های واقعی و با استفاده از ربات‌های فیزیکی است. سکوهای شبیه‌سازی زیادی برای علم رباتیک وجود دارند. در این بخش چند مورد از آن‌ها مختصرا معرفی می‌گردند.

**شبیه‌ساز Player/Stage** یکی از معروف‌ترین محیط‌های شبیه‌سازی است [۲۵, ۲۶]. بخش Player، سروری است که دسترسی کامل به حسگرها و محرک[۱]های ربات را فراهم می‌آورد. بخش Stage نیز شبیه‌سازی است که می‌تواند مقیاس مسئله را تا ۱۰۰۰ ربات افزایش دهد. محیط مورد استفاده در Stage دو بعدی است.

**Gazebo** شبیه‌ساز توسعه یافته‌ی Stage است که بر روی محیط‌های سه بعدی عمل می‌کند [۲۷]. این محیط علاوه بر واسط کاربری خود، از واسط کاربری Player نیز پشتیبانی می‌نماید. بنابراین کنترل-کننده‌های نوشته شده در Stage قابل استفاده در Gazebo است و برعکس.

**Enki** به صورت متن باز و در محیط دو بعدی با زبان C++ نوشته شده است [۲۸]. این محیط توانایی شبیه‌سازی حسگرها، حرکات، تصادم[۲] و دوربین‌ها را در سطح صاف دارد. شبیه‌سازی Enki صدها برابر سریع‌تر از ربات‌های دنیای واقعی است. این محیط برای شبیه‌سازی ربات‌هایی چون e-puck [۲۹] و swarmbot [۳۰] نوشته شده است. با این حال کاربران می‌توانند از این محیط برای ربات‌های دیگر استفاده نمایند.

**Webots** محیطی است برای مدل‌سازی، برنامه‌ریزی و شبیه‌سازی ربات‌ها که بیش از ده سال از ایجاد آن می‌گذرد [۳۱]. کاربر در این محیط می‌تواند تنظیمات پیچیده‌ای روی ربات انجام دهد. می‌توان یک

---

[۱] Actuators

[۲] Collision

یا چندین ربات به صورت همگن یا ناهمگن تعریف کرد. تنظیمات اولیه و از پیش تعریف شده‌ی وسیعی بر روی حسگرها و اجزای حرکتی امکان‌پذیر می‌باشد. اشیای محیط می‌توانند توسط کاربر سفارشی-سازی شوند. امروزه بیش از ۱۰۰۰ دانشگاه از این محیط شبیه‌سازی استفاده می‌کنند. یکی از معایب این محیط کاهش سرعت زیاد در هنگام استفاده از بیش از ۱۰۰ ربات است.

**Breve** محیطی سه بعدی است [۳۲]. رفتار و تعاملات ربات‌ها در این محیط توسط زبان پایتون نوشته می‌شود. کاربر می‌تواند محیط شبیه‌سازی را از هر جهت و موقعیتی ببیند.

**V-REP** محیطی سه بعدی و متن باز است [۳۳]. این محیط ماژول‌های محاسباتی زیادی همچون حرکت‌سنجی روبه‌جلو و عقب، مسیریابی و محاسبه کمترین فاصله دارد.

**ARGoS** محیطی سه بعدی و متن باز است [۳۴]. سکوی آرگوس یک شبیه‌ساز برای سیستم‌های چندرباته است که تاکید ویژه‌ای بر مقیاس جمعیت ربات‌ها با حفظ سرعت دارد. یکی از قابلیت‌های منحصر بفرد آرگوس، امکان کامپایل کد کنترلی برای ربات‌های واقعی است؛ بنابراین نیازی به پیاده‌سازی دو برنامه متفاوت، یکی به هدف شبیه‌سازی و یکی به هدف ربات واقعی وجود ندارد. آرگوس به زبان C++ نوشته شده است.

در جدول ۱-۳ نیز به معرفی برخی از پروژه‌ها و ربات‌های پرکاربرد پرداخته شده است؛ نام، اندازه، حسگرها، راه ارتباطی و سازنده مرتبط با هر یک از ربات‌ها در این جدول ارائه شده است. هر یک از ربات‌های معرفی شده در جدول ۱-۳ در ادامه تشریح شده‌اند.

**ربات Khepera** با گذشت زمان در نسخه‌های متفاوتی ارائه شده است [۳۵]. نسخه‌ی اولیه‌ی آن در گذشته بسیار پرکاربرد بوده ولی امروزه از آن استفاده نمی‌شود. جدیدترین نسخه‌ی این ربات Khepera IV است که حسگرهای مادون قرمز آن جهت تشخیص موانع، چهار حسگر دیگر مادون قرمز برای دنبال کردن خط یا جلوگیری از افتادن و حسگرهای مافوق صوت آن برای تشخیص اشیا قرار داده شده‌اند. ویژگی‌های جدیدی چون شتاب‌سنج، دوربین رنگی، مجاورت‌سنج، میکروفن و سه ال‌ای‌دی، دقت این نسخه را بسیار بالاتر برده‌اند. باتری این ربات تا پنج ساعت دوام می‌آورد. محیط‌های شبیه‌سازی همچون V-REP و Webot برای این ربات مناسب هستند.

**ربات e-puck** دارای حسگر مجاورت‌سنج برای تشخیص نزدیکی اجسام، میکروفن برای تشخیص صداهای محیط و دوربین رنگی است [۳۶]. باتری این ربات تا ۱۰ ساعت عمر می‌کند و محیط‌های شبیه‌سازی چون Enki، WEbots و ARGoS برای این ربات مناسب هستند.

**ربات Alice** نیز از IR برای تشخیص موانع استفاده می‌کند [۳۷]. باتری این ربات، عمر ۲۰ ساعته داشته و محیط شبیه‌سازی پیشنهادی برای این ربات Webot است.

**ربات Jasmine** ارزان و قابل اعتماد است و می‌تواند با تعداد بیش از ۱۰۰ ربات، اهداف رباتیک جمعی را تحقق بخشد [۳۸]. عمر باتری این ربات یک تا دو ساعت است. محیط شبیه‌سازی LaRoSim برای این ربات پیشنهاد می‌گردد.

**ربات‌های I-swarm** بسیار کوچک هستند و می‌توانند اجتماعی با بیشتر از ۱۰۰ عضو داشته باشند [۳۹]. این ربات‌ها می‌تواند ارتباطی بین میکرو رباتیک و سیستم‌های چندرباته ایجاد نمایند (شکل ۱-۳-الف).

**ربات S-bot** با حسگر IR و راه ارتباطی WiFi، دارای یک باتری با عمر یک ساعت است [۴۰]. محیط شبیه‌سازی سفارشی Bot3D نیز برای این ربات استفاده می‌گردد (شکل ۱-۳-ب).

**ربات Kobot** با حسگر IR از برخورد با موانع جلوگیری کرده و از طریق بی‌سیم با اطراف ارتباط برقرار می‌کند [۴۱]. عمر مفید باتری آن ۱۰ ساعت و محیط شبیه‌سازی آن نیز توسط سازنده‌هایش سفارشی‌سازی شده است.

**ربات Swarmbot** به منظور خودسازماندهی و خودچینشی اجسام در محیط دو بعدی ساخته شده است [۳۰]. پروژه‌ی Swarmanoid ادامه‌ی راه این ربات را در محیط سه‌بعدی به عهده گرفته است. باتری این ربات‌ها عمری سه ساعته دارند. Swarmbot3D و Enki محیط‌های شبیه‌سازی مناسب برای این ربات می‌باشند.

**ربات KiloBot** با هدف آزمون رفتارهای تجمعی بر روی صدها یا هزاران ربات ساخته شده است [۴۲]. باتری این ربات از سه تا ۲۴ ساعت عمر دارد. محیط‌های شبیه‌سازی V-REP و ARGoS برای این ربات پیشنهاد می‌گردند.

**الف** **ب**

**شکل ۱-۳: الف) میکرو ربات I-Swarm [۴۳] ب) همکاری در ربات‌های S-bot [۴۴].**

**جدول ۱-۳: معرفی برخی از پروژه‌ها و ربات‌ها**

| نام | قطر (میلی متر) یا طول × عرض | حسگرها | ارتباط | سازنده |
|---|---|---|---|---|
| Khepera | 55 | هشت حسگر مادون قرمز[۱] | ارتباط سیمی | دانشگاه EPFL |
| Khepera III [۳۵] | 120 | ۱۱ حسگر مادون قرمز، پنج حسگر فراصوت[۲] | بلوتوث، وای‌فای | K-Team و دانشگاه EPFL |
| Khepera IV | 140 | ۱۲حسگر مادون قرمز، پنج حسگر فراصوت، دوربین رنگی | بلوتوث، وای‌فای | K-Team و دانشگاه EPFL |
| e-puck [۳۶] | 75 | ۱۱ حسگر مادون قرمز، دوربین رنگی، میکروفون | بلوتوث | دانشگاه EPFL |
| Alice [۳۷] | 20×20 | حسگر مادون قرمز، چراغ، دوربین | امواج رادیویی | دانشگاه EPFL |
| Jasmine [۳۸] | 23×23 | هشت حسگر مادون قرمز | امواج مادون قرمز | تولید شده در پروژه‌ی I-swarm |
| I-swarm robot [۳۹] | 3×3 | حسگر نوری[۳] | بی‌سیم | تولید شده در پروژه‌ی I-swarm |
| S-bot [۴۰] | 120 | میکروفون، حسگر دما، دوربین | وای‌فای | تولید شده در پروژه‌ی Swarm- bots |
| Kobot [۴۱] | 120 | هشت حسگر مادون قرمز، دوربین رنگی | زیگ‌بی[۴] | دانشگاه Middle East Technical |
| Swarmbot [۳۰] | 127×127 | حسگر مادون قرمز، حسگر نوری، دوربین | امواج مادون قرمز | شرکت i-Robot دانشگاه IRIDIA |
| KiloBot [۴۲] | 33 | حسگر نوری، حسگر فاصله | امواج مادون قرمز | دانشگاه Harvard |

۱ Infrared
۲ Ultrasound
۳ Optical Sensor
۴ ZigBee

## ۱-۶- تعریف مسئله و اهداف

تحقیقات در زمینه‌ی رباتیک جمعی همچنان راه درازی در پیش دارد. به دلیل عدم وجود تعریف دقیق از سیستم رباتیک جمعی و مسائل آن، اغلب الگوریتم‌های موجود برای کاربرد خاص طراحی شده‌اند و الگوریتمی که قابلیت استفاده مجدد در چندین کاربرد را داشته باشد، هنوز ارائه نشده است. در اغلب مطالعات مروری در این زمینه، الگوریتم‌ها را بر اساس وظایف و کاربردها دسته‌بندی می‌نمایند [۹, ۴۵]. برخی از وظایف ارائه شده ممکن است به دلیل اشتراکات زیاد به چند دسته تعلق داشته باشند؛ هدف از این دسته‌بندی، جداسازی کامل وظایف نیست. «گردهمایی[۱]»، «تشکیل الگو[۲]»، «آرایش زنجیره‌ای[۳]» و «خود اتصالی» در دسته‌ی رفتارهای سازماندهی قرار دارند. رفتارهای «حرکتی و جهت یابی[۴]» شامل وظایفی چون «کاوش تجمعی[۵]»، «حرکت هماهنگ[۶]» و «حمل و نقل تجمعی[۷]» می‌باشند. «اتفاق نظر[۸]» و «تخصیص وظایف[۹]» نیز در دسته‌ی «تصمیم‌گیری تجمعی[۱۰]» قرار داده شده‌اند.

وظیفه‌ی کاوش هدف[۱۱] الهام گرفته از عملیات جستجوی غذای دسته جمعی زنبورها و مورچه‌ها است [۴۶–۴۸]. این موجودات با استفاده از تعاملات محلی بین خود به جستجوی منابع غذا می‌پردازند. در سیستم‌های رباتیک جمعی محل خاصی به عنوان لانه در نظر گرفته شده و وظیفه‌ی ربات‌ها کاوش در محیط و آوردن اهداف به لانه است. عملیات مین روبی، استخراج معادن، کاوش فضایی و عملیات جستجو و نجات از کاربردهای این نوع وظیفه هستند. کاوش هدف یک وظیفه‌ی پیچیده بشمار می‌رود و می‌توان آن را ترکیبی از وظایف ساده‌تر دانست. به عنوان مثال، کاوش تجمعی، حمل‌ونقل تجمعی و تخصیص وظایف در این مسئله گنجانده شده‌اند. برخی از سیستم‌ها وظایفی چون پراکنده شدن در محیط و یافتن کوتاه‌ترین مسیر را نیز جزئی از مسئله‌ی کاوش هدف تعریف می‌نمایند.

---

۱ Aggregation
۲ Pattern Formation
۳ Chain Formation
۴ Navigation Behaviors
۵ Collective Exploration
۶ Coordination Motion
۷ Collective Transportation
۸ Consensus Achievement
۹ Tasks Allocation
۱۰ Collective Decision Making
۱۱ Foraging

استفاده از تئوری کنترل سوپروایزری[۱] (SCT) در حوزه‌ی رباتیک جمعی اولین بار توسط یوری لوپز و سایرین در سال ۲۰۱۶ مطرح شد [۴۹]. مزایای استفاده از SCT و ابزارهای آن در مقابل استفاده از روش‌های فرمال[۲] موجود به شرح زیر است:

- کاهش راه‌حل‌های تک‌منظوره[۳]. پیش‌تر اشاره شد که در حوزه‌ی رباتیک جمعی، راه‌حل‌ها به صورت تک‌منظوره هستند.
- تولید خودکار کدهای کنترلی به منظور مدل‌سازی ضابطه[۴]های سیستم. در اغلب سیستم‌های فرمال کدها به صورت دستی تولید می‌شوند.
- به جای طراحی یک ماشین حالت متناهی پیچیده، می‌توان رفتار سیستم را به بخش‌های کوچک و ساده تجزیه نمود.
- اثبات وجود ویژگی‌ها[۵] در کد کنترلی.
- قابلیت استفاده‌ی مجدد[۶] کنترل‌کننده‌های طراحی شده در سکوهای رباتیک مختلف.

مسئله‌ی هدف در این پایان‌نامه، مدل‌سازی وظیفه‌ی کاوش هدف با استفاده از تئوری کنترل سوپروایزری است. علت انتخاب وظیفه‌ی کاوش هدف، پیچیدگی و ترکیبی از چند وظیفه بودن آن می‌باشد. با مدل کردن این وظیفه، در واقع چندین وظیفه‌ی رباتیک جمعی مدل شده‌اند. از طرفی دامنه‌ی کاربردهای این وظیفه در صنعت، نجوم، امور نظامی و عملیات نجات بسیار گسترده است.

## ۱-۷- کاربردها

تکنولوژی‌های پیچیده‌ای همچون رباتیک جمعی به دلیل فاکتورهایی مانند خودسازماندهی، مقاومت و انعطاف، دامنه‌ی کاربردهای وسیعی دارند. در ادامه چند مورد از این کاربردها نام برده خواهد شد. در هر حوزه، کاربردهای مشابهی نیز وجود دارد که از بیان موردی آن‌ها خودداری شده است.

- **وظایفی که یک ناحیه را پوشش می‌دهند**. به دلیل توزیع شدگی سیستم‌های رباتیک جمعی، در مواقعی که نیاز به پوشش یک منطقه است می‌توان از آن‌ها استفاده نمود. به عنوان مثال می‌توان به سیستم‌های نظارت محیط و تشخیص سریع خطراتی مانند نشتی مواد شیمیایی اشاره نمود. مزیت رباتیک جمعی در مقایسه با شبکه‌ی حسگرها، در مواجهه با نشتی این است که اولا به دلیل توانایی حرکت ربات‌ها، محل نشتی به صورت دقیق قابل تشخیص

---

[۱] Supervisory Control Theory (SCT)
[۲] Formal
[۳] Ad Hoc
[۴] Specification
[۵] Properties
[۶] Reusability

است. دوما ربات‌ها می‌توانند چینش خود را به شکلی تغییر دهند که جلوی نشتی را بگیرند. از دیگر مثال‌های این حوزه، نقشه‌برداری[1] منطقه، کاوش سیاره‌های دیگر، جستجوی سنگ معادن و یا یافتن یک منبع خاص مانند بو است.

- **وظایفی که خطرناک هستند**. یکی از قابلیت‌های سیستم‌های رباتیک جمعی، غیرضروری بودن هر یک از اعضاست. در صورتی که یکی از اعضا از بین برود، بقیه‌ی اجتماع به درستی به کار خود ادامه می‌دهند. این ویژگی باعث استفاده از رباتیک جمعی در کاربردهای خطرناکی همچون استخراج معدن و مین‌روبی می‌گردد. در حالی که استفاده از یک ربات پیچیده و گران قیمت در این کاربرد عاقلانه نیست.
- **وظایفی که مقیاس آن‌ها در طول زمان بزرگ یا کوچک می‌شود**. نشت نفت از یک کشتی غرق شده را در نظر بگیرید. در صورتی که مخزن کشتی شکسته شود، مقیاس نشتی بزرگ‌تر می‌گردد. رباتیک جمعی در این مواقع می‌تواند با اعزام تعداد زیادی نیرو به راحتی نشتی را کنترل و محیط را پاک‌سازی کند.
- **وظایفی که نیاز به تعداد زیادی نیرو دارد**. حضور در میادینی همانند میدان جنگ، علاوه بر خطرناک بودن به تعداد زیادی نیرو نیاز دارد. در صورتی که تعدادی از نیروها از بین بروند، سایر اعضا می‌توانند به وظیفه‌ی سراسری‌شان ادامه دهند.

برخی از کاربردها، اشتراکی از وظایف بالا هستند و در دسته‌ی خاصی قرار نمی‌گیرند. به عنوان مثال می‌توان به کنترل هواپیماهای بدون سرنشین (UAV)، عملیات نجات، کاربردهای نظامی، حمل و نقل تجمعی، شکار، مهاجرت، پراکندگی[2] ربات‌ها در منطقه به شکلی که بیشترین فضای ممکن پوشش داده شود، تشکیل الگو یا تغییر موقعیت ربات‌ها، حرکت تجمعی[3]، تخصیص وظایف و خوشه‌بندی اشاره کرد. رفتارهایی چون گردهمایی و حرکت تجمعی، پایه و اساس رفتارهای پیچیده‌تر هستند. به منظور انجام وظایفی چون حرکت تجمعی، تشکیل یک شکل و تبادل اطلاعات، ابتدا ربات‌ها باید گرد هم آیند. در بخش مروری بر کارهای گذشته به صورت جامع‌تر به معرفی هر وظیفه پرداخته شده است.

## ۱-۸- چالش‌ها

به دلیل جدید بودن مطالعات رباتیک جمعی، کاربردها و چالش‌های زیادی وجود دارد که محققان در تلاش برای مقابله با آن‌ها هستند. هنوز سیستم رباتیک کاملی در کاربردهای دنیای واقعی ارائه نشده

[1] Cartography
[2] Dispersion
[3] Collective Movement

است و مطالعات همگی در سطح آکادمیک صورت گرفته‌اند [۴۵]. در این بخش به چند مورد از مسائل باز و چالش‌های این حوزه اشاره می‌شود.

- ایجاد سیستمی که قادر به تغییر وظیفه در طول زمان باشد نیز از دیگر موضوعات جالب توجه است. پیاده‌سازی سیستم رباتیکی که بتواند در زمان‌های مختلف بین وظایفی چون گردهمایی، پراکندگی، تشکیل الگو و حمل و نقل گروهی اشیا تغییر وضعیت دهد، امر دشواری است.
- هنوز هم در کاربردهای دنیای واقعی که نیاز به تعداد ربات‌های زیاد وجود دارد، همچون مین‌روبی، نظارت بر ناحیه‌ای بزرگ و پاک‌سازی ناحیه‌ای که نفت در آن نشت کرده است، سیستم‌های رباتیک کارایی مناسبی ندارد. این مسئله به دلیل هزینه‌ی ربات‌ها و محدودیت‌های سخت‌افزاری است.
- سیستم رباتیک جمعی به کنترل‌کننده‌ها و مکانیزم مناسب برای حسگرها نیاز دارد؛ بنابراین برای محققان وجود سکوهای مناسب به عنوان بستر آزمون مهم است. تا به حال بستر آزمون مشترکی برای همه‌ی وظایف رباتیک جمعی ارائه نشده است.
- ارائه‌ی روش مشترک برای ارزیابی عملکرد سیستم نیز از جمله مواردی است که در آینده می‌تواند به مقایسه‌ی بهتر سیستم‌ها کمک کند. تاکنون نرم‌افزارهای کنترلی به شیوه‌ی تک-منظوره ارائه شده‌اند. به همین دلیل استفاده از این سیستم‌ها در دنیای واقعی و نگهداری، تحلیل و اعتبارسنجی آن‌ها دشوار است. روش‌های فرمال برای مقابله با این مشکلات ارائه شده‌اند ولی در اغلب این روش‌ها به دلیل تولید دستی کدهای کنترلی، اثباتی برای انطباق پیاده‌سازی ارائه شده و مشخصات سیستم وجود ندارد.
- مطالعه بر روی امنیت در سطح ربات و در سطح اجتماع نیز از موضوعات باز در زمینه‌ی رباتیک جمعی است.

دسته‌ای از چالش‌ها با عنوان چالش‌های امنیتی شناخته می‌شوند که برخی از آن‌ها در ادامه تشریح شده‌اند [۵۰].

- حمله‌ی فیزیکی به ربات‌ها. ربات‌ها ممکن است در مناطقی که حیوانات وجود دارند، دچار خسارت شوند. از طرفی در کاربردهای نظارتی ممکن است ربات‌ها توسط انسان‌ها گرفته شده و از بین بروند. به عنوان مثال نظارت بر شکار غیر قانونی در جنگل‌های انبوه.
- تشخیص هویت ربات‌ها. ربات‌ها باید در هنگام تعامل متوجه شوند که ربات دیگر از دسته‌ی آن‌ها است یا نه. تداخل دو دسته ممکن است باعث اختلال در عملکرد ربات‌ها شود. حتی ممکن است رباتی که وارد دسته می‌شود یک ربات متهاجم بوده و تعامل با او خطرناک باشد. به عنوان مثال هواپیماهای بدون سرنشین و خطر تعامل با هواپیمای دشمن.

- ارتباط و انتقال اطلاعات بین ربات‌ها ممکن است دچار حمله‌ی امنیتی شود. به عنوان مثال سیستم ربات‌ها در کاربردهای نظامی مانند میدان جنگ یا ورود ربات‌های جاسوسی.

مسئله‌ی هدف نیز چالش‌هایی دربردارد که در ادامه به تعدادی از آن‌ها اشاره می‌شود. این چالش‌ها با توجه به چند رباته بودن سیستم، ویژگی‌های رباتیک جمعی، محدودیت‌ها و مشخصات وظیفه‌ی کاوش هدف و استفاده از کنترل سوپروایزری برای مدل‌سازی انتخاب شده‌اند.

- همانطور که گفته شد، ایجاد سیستمی که قادر به تغییر وظیفه در طول زمان باشد امر دشواری است. در مسئله‌ی کاوش هدف، ربات‌ها باید بتوانند پس از انجام وظایفی چون پراکنده شدن در محیط و تقسیم وظایف، به جستجوی اهداف پرداخته و سپس آن را به محل مورد نظر حمل نمایند. در این راستا یکی از چالش‌های مهم مسئله، تغییر وظیفه در طول زمان خواهد بود.
- جلوگیری از برخورد ربات‌ها به یکدیگر و ایجاد تداخل در عملکرد ربات‌ها
- جلوگیری از برخورد با موانع موجود در محیط
- جلوگیری از وقوع بن‌بست
- در انجام وظیفه‌ی کاوش هدف، ربات‌ها انرژی زیادی صرف می‌کنند و باید زمانی را برای استراحت آن‌ها در نظر گرفت. مدیریت حالت‌های کار کردن و استراحت ربات‌ها به گونه‌ای که میزان انرژی ربات‌ها در هر زمان بیشترین حد ممکن بوده و از طرفی قید مدیریت زمان نیز در نظر گرفته شود بسیار دشوار است.
- ویژگی‌های محدود کننده‌ی رباتیک جمعی از جمله ارتباط محدود ربات‌ها با یکدیگر، توانایی محاسباتی اندک و عدم وجود رهبر مرکزی بر دشواری مسئله می‌افزاید.

استفاده از تئوری کنترل سوپروایزری در مدل‌سازی وظیفه‌ی پیچیده‌ای چون کاوش هدف به دلیل مشخصات و توانایی‌های زیاد و پیچیده‌ی مسئله امر دشواری است.

## ۱-۹- مفروضات

برای تمرکز بیشتر در پیاده‌سازی هر چه بهتر سیستم پیشنهادی، مفروضات زیر در نظر گرفته شده‌اند. برخی از این مفروضات به دلیل ویژگی‌های ذاتی سیستم‌های رباتیک جمعی هستند.

- هنگامی که از SCT برای مدل‌سازی سیستم استفاده می‌شود، فرض بر این است که سیستم به شکل رخداد گسسته است.
- فرض می‌شود که برای انجام وظیفه‌ی هدف، نیاز به همکاری ربات‌ها نیست.

- توانایی‌های هر یک از ربات‌ها باید کم باشد؛ به گونه‌ای که توانایی انجام وظایف بزرگ را به تنهایی نداشته باشند.
- هماهنگی بین ربات‌ها توزیع شده است و هر ربات حسگرها و ارتباطات محدود و محلی دارد.

## ۱-۱۰- نوآوری‌های رساله

استفاده از تئوری کنترل سوپروایزری در حوزه‌ی رباتیک جمعی در سال ۲۰۱۶ توسط [۴۹] مطرح شده و تنها در مدل کردن چهار وظیفه‌ی پایه‌ای گردهمایی، تفکیک، خوشه‌بندی اشیا و تشکیل گروه به همراه تولید خودکار کد استفاده شده است. کاوش هدف به دلیل دامنه‌ی کاربرد وسیع در صنعت، امور نظامی، نجوم و عملیات نجات از مسائل چالشی و مورد توجه محققان می‌باشد. مدل‌سازی وظیفه‌ی پیچیده‌ای چون کاوش هدف، مشکلات و چالش‌های خاص خود را دارد که در بخش چالش‌ها به تعدادی از آن‌ها اشاره شد. ماهیت هدف در هر یک از کاربردهای این وظیفه متفاوت است. سنگ معدن، انواع محصولات خطوط تولید، انسان، حیوانات، مین، مواد موجود در دیگر سیارات تعدادی از اهداف ممکن برای وظیفه‌ی کاوش هدف بشمار می‌روند. از طرفی این وظیفه یک وظیفه‌ی ترکیبی به شمار می‌آید؛ زیرا برای انجام آن نیاز به مجموعه‌ای از سایر وظایف از قبیل «دوری از موانع»، «جستجو در محیط»، «برداشتن و حمل هدف» و «بازگشت به خانه» است.

عدم امکان تولید خودکار کد، عدم اثبات تحقق ویژگی‌های سیستم در کد تولید شده و دیگر محدودیت-های استفاده از روش‌های فرمال که در بخش تعریف مسئله مطرح شد، هدف اصلی استفاده از SCT برای مدل‌سازی این وظیفه می‌باشد. برای رسیدن به این هدف، مطالعه جامعی بر روی روش‌های موجود صورت گرفت که در قالب یک مقاله گزارش شده است:

۱. **فائزه میرزائی، علی‌اکبر پویان، "مروری بر رباتیک جمعی و جایگاه آن در سیستم‌های چندرباته"، مجله مهندسی برق و الکترونیک ایران، پذیرفته شده، آبان ۹۶**

پس از انتخاب SCT برای مدل‌سازی وظیفه کاوش هدف، مطالعات بر روی این موضوع متمرکز شد. در گام اول، ربات‌های مناسب برای رباتیک جمعی انتخاب گشت. سپس محیط شبیه‌سازی مناسب با در نظر گرفتن فاکتورهای مورنیاز بررسی و انتخاب گشت. آنگاه، ابزاری برای خودکارسازی عملیات موردنیاز از طراحی مدل‌ها تا تولید کد کنترلی طراحی شد. با توسعه‌ی مرحله به مرحله‌ی سیستم، این نرم‌افزار نیز توسعه داده شده است. در حال حاضر، این ابزار پس از دریافت مدل‌های رفتار آزاد[۱] و ضابطه‌های کنترلی[۲]، مراحلی از قبیل اعتبارسنجی ساختار کلی طراحی صورت گرفته، کمینه‌سازی آتاماتاها، سنتز[۳]

---

[۱] Free Behaviour Models

[۲] Control Specifications

[۳] Synthesis

کنترل کننده‌ها، حذف حالت‌های بد و غیرقابل دسترس و تولید کد کنترل کننده‌ها را به صورت خودکار انجام می‌دهد. برای بررسی عملکرد این ابزار، استراتژی گردهمایی (تجمع) ربات‌ها با استفاده از SCT پیاده‌سازی و شبیه‌سازی شد. نتایج حاصل از این آزمایش در قالب یک مقاله گزارش شد:

۲. **فائزه میرزائی، علی‌اکبر پویان، سعیده فردوسی، "پیاده‌سازی و شبیه‌سازی استراتژی تجمع ربات‌ها با استفاده از تئوری کنترل سوپروایزری در رباتیک جمعی"، کنفرانس پردازش سیگنال و سیستم-های هوشمند، پذیرفته و ارائه شده، آذر ۹۶**

با بررسی پیش‌نیازهای طراحی یک مدل برای پیاده‌سازی وظیفه کاوش هدف و مطالعات زمینه‌ای بیشتر، لزوم استفاده از مولفه‌هایی مانند احتمالات و زمان در طراحی فرمال روشن شد. از این رو، تئوری کنترل سوپروایزری احتمالاتی و زمان‌دار در رباتیک جمعی مورد بررسی قرار گرفت و در نهایت پیشنهاد شد. نتیجه این پیشنهاد در قالب یک مقاله منتشر شد:

۳. **فائزه میرزائی، علی‌اکبر پویان، سعیده فردوسی، "پیاده‌سازی تئوری کنترل سوپروایزری احتمالی و زمان‌دار در رباتیک جمعی"، کنفرانس ملی فناوری‌های نوین در مهندسی برق و کامپیوتر، پذیرفته و ارائه شده، دی ۹۶**

با استفاده از دو مولفه‌ی احتمالات و زمان می‌توان مدل‌های قدرتمندتری را با پیچیدگی کمتر در مقایسه با عدم استفاده از این دو مولفه، ارائه کرد. برای نمایش این موضوع، دو وظیفه دوری از موانع و همگام‌سازی ربات‌ها در رباتیک جمعی یک مرتبه با استفاده از تئوری کنترل سوپروایزری احتمالاتی و زمان‌دار و یک مرتبه بدون آن پیاده‌سازی و شبیه‌سازی شدند. این مقایسه‌ها در قالب یک مقاله جدید ارائه شدند:

۴. **فائزه میرزائی، علی‌اکبر پویان، سعیده فردوسی، "شبیه‌سازی و پیاده‌سازی وظایف دوری از موانع و همگام‌سازی با استفاده از تئوری کنترل سوپروایزری احتمالی و زمان‌دار در رباتیک جمعی"، مجله مدل‌سازی در مهندسی، اسفند ۹۶**

با داشتن دو مولفه‌ی احتمالات و زمان، به توسعه ابزار جامع طراحی پیشنهادی پرداخته شد و همه قابلیت‌های نرم‌افزار برای استفاده بهینه از این دو مولفه نیز گسترس پیدا کرد. ابزار پیشنهادی قادر به سنتز کنترل کننده‌ها به دو شکل یکپارچه[۱] و ماژولار محلی[۲] می‌باشد. از آنجا که طراحی و سنتز به صورت فرمال صورت می‌گیرد، قابلیت اعتبارسنجی مدل‌ها نیز به ابزار پیشنهادی اضافه گشت. در نهایت با استفاده از تمامی امکانات پیاده‌سازی شده، به طراحی وظیفه کاوش هدف پرداخته شد. پس از اتمام طراحی، آزمایش‌های متعددی برای بررسی صحت عملکرد سیستم در مورد شبیه‌سازی ترتیب داده شد

---

[۱] Monolithic

[۲] Local Modular

و جوانب مختلف عملکرد سیستم مورد بررسی قرار گرفت. نتایج نهایی در قالب یک مقاله‌ی جامع ارائه گشت:

5. **Faezeh Mirzaei, Ali Akbar Pouyan, Saideh Ferdowsi, "Automatic Controller Design for Swarm Robotics using Probabilistic Timed Supervisory Control Theory (ptSCT)", Autonomous Robots, Under Review, June 2019**

برای سادگی و خوانایی بهتر موارد ذکر شده در بالا، نوآوری‌های رساله در ادامه به اختصار فهرست شده است:

- استفاده از تئوری کنترل سوپروایزری برای تولید فرمال کنترل‌کننده‌های رباتیک جمعی
- پیشنهاد تئوری کنترل سوپروایزری احتمالاتی و زمان‌دار
- طراحی، مدل‌سازی و شبیه‌سازی وظیفه‌ی کاوش هدف
- تولید خودکار کدهای کنترلی متناظر با طراحی فرمال
- اعتبارسنجی پیاده‌سازی‌های انجام شده با استفاده از آزمایش‌های متعدد
- پیاده‌سازی و شبیه‌سازی وظیفه‌ی کاوش هدف در رباتیک جمعی
- ارائه یک ابزار برای انجام خودکار تمامی مراحل موردنیاز از طراحی تا شبیه‌سازی

## ۱-۱۱- ساختار رساله

رساله جاری در هفت فصل ارائه شده است. مقدمه‌ای در رابطه با موضوع رساله، اهداف و چالش‌های آن در فصل اول ارائه گشته است. در فصل دوم، کارهای پیشین به طور مفصل مورد بررسی قرار گرفته‌اند. تئوری کنترل سوپروایزری به عنوان مبنای روش پیشنهادی در فصل سوم به صورت مجزا مورد بحث قرار گرفته است. روش پیشنهادی در قالب دو فصل چهارم و پنجم ارائه شده است. فصل چهارم به توضیح ابزار جامع پیشنهادی برای طراحی، مدل‌سازی و تولید خودکار پرداخته است؛ و فصل پنجم به تشریح طراحی صورت گرفته برای وظیفه کاوش اختصاص داده شده است. فصل ششم شامل ارزیابی و نتایج آزمایش‌های انجام شده بر روی سیستم پیشنهادی می‌باشد. در پایان و در فصل هفتم به نتیجه-گیری مباحث مطرح شده پرداخته شده است.

## ۱-۱۲- جمع‌بندی

در این فصل ابتدا به معرفی سیستم‌های تک‌رباته، چندرباته پرداخته شد. سپس رباتیک جمعی معرفی و تفاوت‌های آن با سیستم‌های چندرباته تشریح شد. پس از آن، مسئله هدف در این رساله و اهداف تحقیق توضیح داده شد. علاوه‌براین، کاربردهای متنوع رباتیک جمعی و چالش‌های آن مورد اشاره قرار گرفت. در ابتدا سعی شد فهرست جامعی از چالش‌های موجود ارائه شود و سپس چالش‌های هدف در رساله جاری مورد اشاره قرار گرفتند. در ادامه مفروضات مسئله مطرح گردید و در پایان، خلاصه‌ای از

نوآوری‌های صورت گرفته در این رساله و مسیر طی شده برای نگارش مقالات حاصل از روش پیشنهادی بیان شد.

# فصل

# ۲- مروری بر کارهای پیشین

## ۲-۱- مقدمه

در سال‌های اخیر، تلاش محققان بر الهام از طبیعت و تولید الگوریتم‌هایی بوده است که بتواند رفتار تجمعی را به خوبی شبیه‌سازی نماید. به طور کلی الگوریتم‌های رباتیک جمعی دارای پنج ویژگی می‌باشند [۲۰]. «سادگی»، «مقیاس‌پذیری»، «عدم تمرکز»، «محلی بودن» و «موازی بودن» پنج ویژگی الگوریتم‌های موردنیاز رباتیک جمعی است که در ادامه شرح داده می‌شوند. محققان با توجه به این ویژگی‌ها، الگوریتم‌های زیادی ارائه داده‌اند.

- **سادگی:** با توجه به توانایی اندک ربات‌ها، الگوریتم‌ها نیز باید در حد ممکن ساده باشند. الگوریتم ساده به کاهش هزینه‌ی ربات کمک می‌کند. حتی احتساب رفتارهای تجمعی بهینه و پیچیده می‌تواند با طراحی صحیح الگوریتم همکاری ممکن شود. در اغلب موارد ربات‌ها، ماشین حالت متناهی با تعداد حالت اندک در نظر گرفته می‌شوند.
- **مقیاس‌پذیری:** به منظور تحقق ویژگی مقیاس‌پذیری اجتماع، الگوریتم نیز باید در همه‌ی اندازه‌های جمعیت مقیاس‌پذیر باشد. طراح باید پیوستن و ترک پویای هر ربات به جمعیت را در نظر بگیرد.
- **عدم تمرکز:** ربات‌ها در سیستم رباتیک جمعی خودمختار و مستقل هستند، در نتیجه الگوریتم نیز باید این ویژگی‌ها را داشته باشد. الگوریتم باید از هر گونه کنترل مرکزی و خارجی اجتناب نماید؛ حتی در صورتی که رباتی از رفتار ربات دیگر متاثر شد، باید به صورت شخصی تصمیم خود را بگیرد. یک الگوریتم غیرمتمرکز قابلیت مقیاس‌پذیری دارد.
- **محلی بودن:** از دیگر ویژگی‌های رباتیک جمعی تعاملات و ارتباطات محلی است. الگوریتم طراحی شده نیز به منظور ایجاد قابلیت مقیاس‌پذیری باید ویژگی محلی بودن را دارا باشد.
- **موازی بودن:** اجتماع شامل تعداد زیادی ربات است؛ بنابراین الگوریتم باید تا حد ممکن موازی باشد تا ربات‌ها بتوانند به صورت همزمان با چندین هدف مواجه شوند.

ایجاد یک سیستم رباتیک جمعی، نیاز به مراحل چندگانه زیر دارد:

۱. طراحی و مشخص نمودن نیازهای مسئله
۲. تحلیل و مدل‌سازی
۳. پیاده‌سازی

برخی از محققان، بخش تحلیل و مدل‌سازی را انجام نداده و مستقیما بعد از تعیین نیازهای مسئله به سراغ پیاده‌سازی می‌روند. در این حالت، ارزیابی سیستم، اشکال‌یابی و تصحیح طراحی کاری دشوار و زمانبر خواهد بود و به تبحر طراح وابستگی شدید دارد. در نظر گرفتن همه‌ی ابعاد مسئله و همه‌ی

حالت ممکن در سناریوها، امکان بروز خطا را افزایش می‌دهد. مرحله‌ی مدل‌سازی به نوعی با کاهش ابعاد مسئله، تحلیل سیستم را آسان می‌کند. در مدل‌سازی می‌توان انتخاب کرد که از زمان یا مکان و به صورت پیوسته یا گسسته استفاده شود. در ادامه به تشریح دو مرحله‌ی طراحی و مدل‌سازی پرداخته شده است.

## ۲-۲- طراحی

نیازهای یک سیستم رباتیک جمعی در سطح اجتماع مطرح می‌شود ولی طراح باید سخت‌افزار و رفتار سیستم را در سطح اعضای اجتماع تعریف نماید؛ بنابراین طراحی سیستم رباتیک جمعی دشوار است. طراحی ربات‌ها باید به گونه‌ای باشد که رفتار تجمعی آن‌ها با نیازهای مسئله مطابقت داشته باشد. روش‌های طراحی رباتیک جمعی را می‌توان به دو دسته‌ی «طراحی دستی[۱]» و «طراحی خودکار[۲]» تقسیم نمود. تفاوت این دو دسته، در نقش طراح است. در دسته‌ی اول، طراح با استفاده از تبحر و تجربه‌ی خود سیستم رباتیک جمعی را طراحی و تولید می‌نماید. در دسته‌ی دوم، طراح، تنظیمات فرایند تولید را انجام داده و سیستم موردنظر به صورت خودکار تولید می‌شود.

روش‌های طراحی دستی به دو نوع «پایین‌به‌بالا[۳]» و «بالابه‌پایین[۴]» تقسیم می‌شوند. در روش پایین‌به‌بالا، طراح، رفتار اعضا را تا زمانی که به رفتار تجمعی مطلوبی دست یابد، به صورت آزمون و خطا[۵] تولید و آزمایش کرده و ارتقا می‌دهد. در روش بالابه‌پایین، طراح ابتدا رفتار سیستم را در سطح اجتماع تحلیل کرده و سپس رفتار اعضای اجتماع را استنتاج می‌نماید [۵۱]. محدودیت اصلی روش پایین‌به‌بالا وابستگی کیفیت سیستم رباتیک جمعی طراحی شده به تبحر و تجربه‌ی طراح است. از طرفی روش سعی و خطا تضمینی بر کیفیت نتایج نخواهد داشت. بااین‌حال اغلب از روش‌های پایین‌به‌بالا استفاده می‌گردد زیرا تا کنون روش بالابه‌پایین کلی ارائه نشده است.

رایج‌ترین ابزار طراحی پایین‌به‌بالا در رباتیک جمعی ماشین حالت متناهی احتمالاتی[۶] است. در روش‌های احتمالاتی، رفتار ربات‌ها تا حدی به صورت تصادفی و تا اندازه‌ای با توجه به تعامل ربات با محیطش تعیین می‌گردد. به عنوان مثال در کاربرد گردهمایی، ماشین حالت متناهی با دو حالت حرکت و توقف مورد استفاده قرار می‌گیرد [۵۲, ۵۳]. تصمیم جابجایی بین حالت‌ها ممکن است کاملا به صورت تصادفی و یا با استفاده از نشانه‌های محلی باشد. این نشانه‌ها ممکن است به سادگی حضور یک ربات دیگر در

---

۱ Manual Design
۲ Automatic Design
۳ Bottom-Up
۴ Top-Down
۵ Trial-And-Error
۶ Probabilistic Finite State Machine (PFSM)

نزدیکی و یا به شکل‌های پیچیده‌تری باشد [۵۴]. پارامترهای ماشین حالت مانند احتمال جابجایی بین حالت‌ها معمولا به صورت دستی توسط طراح تعیین می‌گردد. با این حال اخیرا روش‌هایی جهت تعیین خودکار پارامترها ارائه شده است [۵۵]. تاکنون از PFSM برای تولید رفتارهای جمعی چون گردهمایی، آرایش زنجیره‌ای و تخصیص وظایف استفاده شده است [۵۶–۵۸]. از مزایای استفاده از PFSMها می‌توان به موارد زیر اشاره نمود:

- اَعمال[۱] و شروط[۲] به سادگی و به صورت ساخت‌یافته قابل ترکیب شدن هستند. حالات ماشین، اعمال ممکن ربات و گذارها، شروط موجود را تعیین می‌کنند.
- با استفاده از PFSM می‌توان رفتارها را به زیررفتارهایی تجزیه[۳] نمود.
- خواندن، تحلیل و فهم یک PFSM به راحتی صورت می‌گیرد.
- با توجه به ماژوله بودن PFSMها، قابلیت استفاده‌ی مجدد از رفتارها وجود دارد.

از دیگر روش‌های پایین‌به‌بالا، استفاده از نیروهای مجازی[۴] در حوزه‌ی فیزیک مصنوعی[۵] است. در فیزیک مصنوعی، رفتار عامل‌ها توسط نیروهای مجازی مدل می‌شوند. این نیروها، حرکت عامل‌ها و تعاملات آن‌ها با محیط را تعیین می‌کنند. آرایش موجود در بسیاری از حیوانات را می‌توان با استفاده از نیروهای جاذبه (عامل‌ها تمایل دارند در نزدیکی یکدیگر بمانند) و نیروهای دافعه (از تصادم و برخورد عامل‌ها جلوگیری می‌کند) مدل نمود. هر ربات بر اساس نیرویی که از همسایه‌هایش بر او اعمال می‌شود حرکت می‌کند. این نیروها بر اساس فاصله‌ی بین ربات‌ها تعیین می‌شوند. اگر فاصله‌ی دو ربات از حد مشخصی بالاتر رود، نیروی جاذبه و اگر فاصله از حد مشخصی پایین‌تر باشد نیروی دافعه خواهیم داشت.

ربات‌ها به دلیل توانایی حس محلی، دید محدودی نسبت به محیط خود دارند. از طرفی تعیین جهت قرار گرفتن همسایه‌ها هنگامی که از تکنولوژی‌هایی چون مادون قرمز استفاده می‌شود با خطای بالایی مواجه است. به همین دلیل مطالعات محدودی از قوانین فیزیک مصنوعی برای حرکت ربات‌ها استفاده کرده‌اند. مرجع [۵۹] گردهمایی ربات‌ها را با استفاده از فیزیک مصنوعی پیاده‌سازی و آزمایش کرده است. در این مقاله، فاصله و جهت نسبی همسایه‌ها با استفاده از فرکانس رادیویی و تکنولوژی مادون قرمز به دست می‌آیند. علاوه بر آن، یک مکانیزم اجتناب از مانع ارائه شده است که در آن موانع به شکل ربات مجازی تعریف شده‌اند. به منظور اجتناب از نیروی جاذبه بین مانع و ربات، ربات مجازی تنها در صورتی فعال می‌شود که فاصله‌اش با ربات واقعی به گونه‌ای باشد که نیروی دافعه از نیروی جاذبه بیشتر

---

۱ Actions
۲ Conditions
۳ Decomposition
۴ Virtual Forces
۵ Artificial Physics

باشد. از نیروی مجازی در انجام وظایفی چون تشکیل الگو، کاوش تجمعی و حرکت هماهنگ استفاده شده است. از مزایای استفاده از نیروی مجازی می‌توان به موارد زیر اشاره نمود:

- به جای استفاده از چندین قانون و رفتار کنترلی، می‌توان با تعریف یک قانون ریاضی فضای ورودی حسگر را به خروجی اجزای حرکتی انتقال داد.
- امکان ترکیب رفتارها به وسیله‌ی عملگرهای برداری وجود دارد.
- برخی ویژگی‌ها مانند تحمل‌پذیری خطا[۱] و پایداری[۲] را می‌توان توسط ابزارهای تئوری موجود در فیزیک، تئوری کنترل و تئوری گراف‌ها اثبات نمود.

در روش بالابه‌پایین، ابتدا رفتار ربات‌ها در سطح اجتماع طراحی شده و سپس رفتار اعضای اجتماع استنتاج می‌گردد. روش اثبات شده‌ای به منظور استنتاج ذکر شده وجود ندارد؛ بنابراین روش‌های بالابه‌پایین بسیار دشوار بوده و معمولا به حل معادلات پیچیده‌ی ریاضی نیازمند هستند. از طرفی به منظور ایجاد نتایج قابل اعتماد در این معادلات، تعداد ربات‌ها باید بسیار زیاد باشد. باخراخ[۳] [۶۰] یک زبان اسکریپتی به نام Protoswarm پیشنهاد داده است که در آن می‌توان اسکریپتی در سطح اجتماع نوشته که رفتار ربات را به صورت فردی تعریف نماید. کازادی[۴] [۶۱] بر اساس بردار همیلتون، طراحی بالا به پایین را انجام داده است. با شروع از یک توصیف ریاضی در سطح اجتماع، به صورت خودکار قوانین در سطح اعضای اجتماع استخراج می‌شود. مشکل اصلی استفاده از روش همیلتون این است که تنها در زمینه‌هایی مانند تشکیل الگو کاربرد دارد.

در طراحی خودکار، روش حل مسئله به صورت جعبه سیاه[۵] است. یکی از پرکاربردترین روش‌های طراحی خودکار، روش‌های تکاملی هستند و به کاربرد آن‌ها در رباتیک، رباتیک تکاملی گفته می‌شود [۶۲]. در این حالت، معمولا یک شبکه‌ی عصبی که پارامترهایش توسط روش‌های تکاملی تعیین می‌شود، ربات‌ها را کنترل می‌نماید [۶۳, ۶۴]. مرجع [۶۵] به جای اعمال شبکه عصبی به یک ربات، نمونه‌های مختلف یک شبکه عصبی یکسان را به کل اجتماع اعمال می‌کند. در صورتی که هر ربات شبکه‌ی عصبی متفاوتی داشته باشد، چالش افزایش می‌یابد.

تنظیمات دشوار و زمانبر، عدم وجود ضمانت همگرایی[۶] و خاصیت جعبه سیاه بودن شبکه‌های عصبی در روش‌های تکاملی از مهم‌ترین محدودیت‌های آن‌هاست. به جای شبکه‌ی عصبی می‌توان از روش‌های

---

۱ Fault Tolerance
۲ Stability
۳ Bachrach
۴ Kazadi
۵ Black Box
۶ Convergence

توصیفی دیگر نظیر ماشین حالت استفاده نمود. مرجع [۶۶] وظیفه‌ی جهت‌یابی را با استفاده از این روش انجام داده است.

یکی از روش‌های دیگر طراحی خودکار، یادگیری تقویتی[۱] است [۶۷]. یادگیری تقویتی از دسته یادگیری‌های بدون نظارت است که نیازی به پایگاه داده‌ی آموزشی ندارد و تنها با استفاده از یک تابع پاداش، حالات مطلوب و نامطلوب را تعیین می‌کند. این روش به دلیل محدودیت‌هایش در حوزه‌ی رباتیک جمعی به ندرت استفاده شده است. مهم‌ترین محدودیت این روش، دشواری تجزیه‌ی پاداش سراسری[۲] به پاداش در سطح اعضای اجتماع است. از طرفی، حجم اطلاعاتی که ربات باید ذخیره کند با افزایش جمعیت، بسیار زیاد خواهد شد.

روش‌های دیگری نیز برای طراحی خودکار ارائه شده است. به عنوان مثال فرانسسکا[۳] و سایرین در [۵۵] با استفاده از الگوریتم‌های بهینه‌سازی و ترکیب رفتارها و شروط ساده‌ی از پیش تعریف شده، یک PFSM را به صورت خودکار تولید کرده‌اند.

## ۲-۳- تحلیل و مدل‌سازی

در فاز تحلیل سیستم، رفتار تجمعی مورد انتظار، ارضای نیازهای مسئله و تحقق ویژگی‌های آن مورد بررسی قرار می‌گیرد. یکی از روش‌های تحلیل شناخته شده، روش بررسی مدل[۴] نام دارد [۶۸]. با استفاده از این تکنیک، می‌توان به صورت فرمال وجود ویژگی‌هایی چون عدم وجود بن‌بست[۵] را اثبات نمود. برخلاف روش‌های تحلیلی چون شبیه‌سازی که تنها زیرمجموعه‌ای از تمامی سناریوهای خروجی ممکن سیستم را بررسی می‌کنند؛ روش بررسی مدل به صورت خودکار تمامی خروجی‌ها را در نظر گرفته و وجود ویژگی موردنظر را بررسی می‌نماید.

ویژگی‌های رفتارهای تجمعی معمولا توسط مفهوم مدل تحلیل می‌شوند. انواع روش‌های مدل‌سازی[۶] در ادامه شرح داده شده است. مدل‌سازی روشی است که در برخی از فیلدهای تحقیقاتی جهت فهم بهتر عملکرد درون سیستم مورد استفاده قرار می‌گیرد. در حوزه‌ی رباتیک جمعی نیز مدل‌سازی می‌تواند مفید باشد؛ زیرا مسئله تا صدها یا هزاران ربات قابل گسترش است و هزینه و زمان برای آزمایش‌های در این سطح محدود هستند. مدل‌سازی مولفه‌ی اصلی الگوریتم همکاری ربات‌ها است که رفتار و

---

[۱] Reinforcement Learning
[۲] Global Reward
[۳] Francesca
[۴] Model Checking
[۵] Deadlock
[۶] Modeling

تعاملات بین ربات‌ها را کنترل می‌نماید. با توجه به مشخصات رباتیک جمعی، روش‌های مدل‌سازی را می‌توان به چهار دسته تقسیم نمود [۶۹]:

۱. مدل‌سازی مبتنی بر حسگر[۱]
۲. مدل‌سازی میکروسکپی[۲]
۳. مدل‌سازی ماکروسکوپی[۳]
۴. مدل‌سازی بر اساس الگوریتم‌های هوش جمعی

در مدل‌سازی مبتنی بر حسگر، حسگرها و محرک‌های ربات به عنوان مولفه‌های اصلی سیستم در کنار اشیای موجود در محیط مدل می‌شوند. آنگاه عمل مدل‌سازی تعاملات ربات‌ها به ساده‌ترین شکل ممکن و نزدیک به واقعیت صورت می‌گیرد. این روش مدل‌سازی، قدیمی‌ترین و پرکاربردترین روش به شمار می‌رود. تحقیقات پیشین محدودیت‌های فیزیکی ربات‌های واقعی را در نظر نمی‌گرفتند [۷۰]. امروزه به این محدودیت‌ها بیشتر توجه می‌شود [۷۱, ۷۲].

در مدل‌سازی میکروسکوپی، ربات‌ها و تعاملاتشان به شکل ماشین حالت متناهی مدل می‌شوند. رفتار هر یک از ربات‌ها با تعدادی حالت تعریف می‌گردد و شرط گذار را ورودی حسگرها یا ارتباطات تعیین می‌نماید. به منظور توصیف رفتار اجتماع، باید توصیف رفتار اعضا به تعداد اعضا تکرار شود. این مسئله پیچیدگی محاسباتی را افزایش خواهد داد. ازاین‌رو، در اغلب موارد، تحلیل مدل‌های میکروسکوپی با استفاده از شبیه‌ساز انجام می‌شود. از آنجا که مدل بر اساس رفتار هر یک از ربات‌ها است، شبیه‌سازی باید چندین مرتبه اجرا شود تا رفتار میانگین اجتماع بدست آید. در اغلب تحقیقات از مدل میکروسکوپی احتمالاتی استفاده می‌شود؛ زیرا نویز را می‌توان به شکل یک احتمال مدل کرد [۷۳].

مدل‌سازی ماکروسکوپی نقطه‌ی مقابل مدل‌سازی میکروسکوپی است. در این نوع از مدل‌سازی یک حالت از سیستم، میانگین حالت ربات‌ها در آن حالت در زمان مشخص را نشان می‌دهد. به طور کلی می‌توان گفت مدل‌سازی میکروسکوپی در سطح ربات‌ها و مدل‌سازی ماکروسکوپی در سطح اجتماع است. بنابراین مدل‌سازی ماکروسکوپی دید سراسری نسبت به اجتماع و مدل‌سازی میکروسکوپی جزئیات رفتار اجتماع را نشان می‌دهند [۷۴]. رایج‌ترین روش مدل‌سازی، روش ماکروسکوپی است.

---

[۱] Sensor-based Modeling
[۲] Microscopic
[۳] Macroscopic

مدل‌سازی ماکروسکوپی به چند روش قابل انجام است؛ استفاده از معادله ارزیابی(rate equation) [۷۵, ۷۶]، معادله لانگوین[۱] و فاکر-پلانک[۲] [۷۷] از رایج‌ترین روش‌ها هستند.

ماریتونی[۳] و سایرین در [۷۸] فریمورکی برای مدل کردن رباتیک جمعی در دو سطح میکروسکوپی و ماکروسکوپی ارائه نموده‌اند. این فریمورک از ماشین حالت متناهی احتمالاتی (PFSM) استفاده می‌کند. نویسندگان این مقاله مطالعه موردی برای برداشتن چوب از زمین توسط ربات‌ها انجام داده‌اند. ماسینک و سایرین در [۷۹] اظهار داشتند که قرار گرفتن دو سطح میکروسکوپی و ماکروسکوپی در کنار هم تناقضاتی دارد. آن‌ها روشی به نام Bio-PEPA[۴] ارائه نمودند که برای مدل‌سازی واکنش‌های بیوشیمی استفاده می‌شود. این روش رباتیک جمعی را تنها در سطح میکروسکوپی مدل کرده و اعتبارسنجی و آنالیز ویژگی‌های ماکروسکوپی را با استفاده از روش بررسی مدل انجام می‌دهد.

PSO[۵] رایج‌ترین الگوریتم هوش جمعی است که در مدل‌سازی استفاده می‌شود. این الگوریتم با الهام از پرواز دسته جمعی پرندگان طراحی شده است [۸۰]. علاوه بر این الگوریتم، الگوریتم‌های دیگری نیز مانند کلونی مورچه‌ها در حوزه‌ی رباتیک جمعی استفاده شده‌اند [۸۱].

در ادامه تعدادی از روش‌های دیگر مدل‌سازی معرفی شده‌اند.. برامبیلا[۶] و سایرین در [۵۱] روشی با عنوان طراحی ویژگی محور[۷] ارائه نمودند. این روش از چهار مرحله تشکیل می‌شود. ابتدا نیازها[۸] به صورت فرمال تعیین می‌گردند. در بخش دوم مدل ماکروسکوپی با استفاده از زنجیره‌ی مارکو[۹] طراحی و توسط روش بررسی مدل ارزیابی می‌شود. سپس از مدل برای راهنمایی پیاده‌سازی و شبیه‌سازی استفاده می‌گردد. در بخش پایانی، پیاده‌سازی و شبیه‌سازی صورت می‌گیرد.

در [۸۲] وظیفه‌ی گردهمایی با استفاده از روش زنجیره‌ی مارکو مدل‌سازی و با استفاده از شبیه‌ساز ارزیابی شده است. در [۶۸] سیستم‌های رباتیک جمعی با استفاده از بررسی مدل احتمالاتی تحلیل شده‌اند. که به صورت فرمال ارضای ویژگی‌های مسئله را بررسی می‌نماید. مرجع [۸۳] وظیفه‌ی

---

۱ Langevin

۲ Fokker–Planck Equation

۳ Martinoli

۴ Biochemical Performance Evaluation Process Algebra

۵ Particle Swarm Optimization

۶ Brambilla

۷ Property-driven Design

۸ Requirements

۹ Markov Chain

گردهمایی را با استفاده از ماشین حالت احتمالاتی مدل کرده است. لابلا[۱] [۸۴] از ماشین حالت احتمالاتی برای کاوش هدف استفاده کرده است.

ماشین حالت متناهی احتمالاتی می‌تواند به صورت خودکار تولید گردد. مرجع [۵۵] روشی به نام AutoMoDe-Vanilla[۲] برای این منظور ارائه داده است. در روش AutoMoDe از ماژول‌های از پیش تعریف شده‌ای برای طراحی استفاده می‌گردد. این ماژول‌ها رفتارها و شروط سیستم را در بردارند. پارامترهای ماشین حالت متناهی احتمالاتی با استفاده از الگوریتم‌های بهینه‌سازی تعیین می‌شوند.

شبکه پتری[۳] روشی دیگر برای مدل کردن نرم‌افزار کنترلی رباتیک جمعی است [۸۵]. کنترل‌کننده‌های مبتنی بر شبکه پتری توسط کامپیوتری مرکزی با ربات‌ها در ارتباط هستند؛ در صورتی که در رباتیک جمعی نباید کنترل‌کننده‌ی مرکزی وجود داشته باشد. در [۸۶] نیز، کاوش هدف با استفاده از شبکه پتری و زنجیره مارکو مدل شده است. با استفاده از این روش، پارامترهای کارایی در بازه‌ی زمانی مشخص قابل پیش‌بینی هستند.

تئوری کنترل سوپروایزری روشی دیگر برای تولید کنترل‌کننده‌ها است که اغلب در خطوط تولید استفاده می‌شود [۸۹, ۹۰]. در این روش، سنتز کنترل‌کننده بر اساس توصیف فرمال سیستم و مشخصاتش صورت می‌گیرد. مطالعاتی نیز درباره‌ی استفاده از SCT در سیستم‌های چند رباته برای انجام وظایفی چون تحویل اشیا انجام شده است [۹۱, ۹۲]. البته تمرکز اصلی این مطالعات بر روی طراحی و تحلیل کنترل‌کننده‌ها است؛ نه بر روی پیاده‌سازی و اعتبارسنجی با استفاده از ربات‌های واقعی. از SCT برای سیستم‌های حمل و نقل انسان یا اشیا نیز استفاده شده است [۹۳].

مرجع [۴۹] اولین مقاله‌ای است که از تئوری کنترل سوپروایزری (SCT) برای مدل کردن رباتیک جمعی استفاده کرده است. در این مقاله چهار وظیفه‌ی گردهمایی، تفکیک ربات‌ها[۴]، تشکیل گروه و خوشه‌بندی اشیا با استفاده از SCT مدل شده و سپس پیاده‌سازی شده‌اند. استفاده از SCT مزایایی همچون تولید خودکار کد از مشخصات مدل شده را دارد. در دیگر روش‌های فرمال استفاده شده در رباتیک جمعی، ضمانتی بر این‌که پیاده‌سازی ارائه شده ویژگی‌های مسئله را دربردارد وجود ندارد. این مشکل در SCT با تولید خودکار نرم‌افزار کنترلی حل خواهد شد. نرم‌افزار کنترلی با استفاده از زبان‌های منظم، فرمال می‌گردد. کنترل‌کننده‌های مدل شده با این روش را می‌توان در سکوهای مختلف استفاده نمود.

---

[۱] Labella

[۲] Automatic Modular Design

[۳] Petri Net

[۴] Segregation

## ۲-۴- کاوش هدف

وظیفه‌ی کاوش هدف الهام گرفته از جستجوی غذای حیواناتی چون مورچه است [۴۷, ۴۸]. ربات‌ها در محیط پراکنده شده و به جستجوی اهداف خاصی می‌پردازند. سپس اهداف جمع‌آوری شده را به منطقه‌ی مشخصی (لانه) منتقل می‌کنند. عملیات مین روبی، استخراج معادن، کاوش فضایی و عملیات جستجو و نجات چند نمونه از کاربردهای این وظیفه می‌باشند.

با توجه به کاربردهای مختلف این وظیفه، سناریوهای مختلفی برای انجام آن مطرح شده‌اند. به عنوان مثال کاوش هدف چند منظوره نوعی از سیستم‌های کاوش هدف است که ربات‌ها به دنبال اشیای مختلف رفته و هر کدام را به لانه‌ی مخصوص آن می‌برند؛ عملیات مین روبی و عملیات جستجو و نجات از این نوع وظایف هستند. ممکن است اهداف در محیط پراکنده باشند و یا در محل خاصی به نام منبع وجود داشته باشند. در این بخش مروری بر روش‌های استفاده شده برای مدل کردن و طراحی این وظیفه انجام شده است.

الگوریتم‌های متفاوتی برای کاوش هدف ارائه شده است. بر اساس دسته‌بندی انجام شده در [۹۴]، سه دسته‌ی کاوش مبتنی بر گرادیان[۱]، کاوش پوشش محیط[۲] و کاوش وفقی[۳] قابل تعریف می‌باشد. در کاوش مبتنی بر گرادیان، ربات‌ها ممکن است با استفاده از مواد شیمیایی یا ارتباطات، یک راهنمایی گرادیانی تا هدف ایجاد کنند. استفاده و پیاده‌سازی فرومون مسئله دشواری است و از آنجا که ارتباط سراسری نداریم باید از ارتباطات محلی استفاده شود. یا ربات‌ها از ردپای فیزیکی مانند الکل، بو، گرما یا علامت‌های مجازی استفاده نمایند. قرار دادن علامت‌های فیزیکی دائم در محیط قابل قبول نیست و پیاده‌سازی علامت‌های موقت نیز دشوار هستند. در روش پوشش محیط، از نیروهای فیزیک بین ربات‌ها برای تصمیم حرکت استفاده می‌شود. در روش وفقی، ربات باید برای جابجایی بین وظایفی چون جمع‌آوری غذا، دوری از موانع و استراحت تصمیم‌گیری کند. یکی از راه‌حل‌های تصمیم‌گیری، رای‌گیری است که امری زمانبر خواهد بود.

حمل شی در وظیفه‌ی کاوش هدف معمولا به دو شکل صورت می‌گیرد. در حالت اول ربات بدون هماهنگی و کمک گرفتن از سایرین، شی را به لانه منتقل می‌کند. در حالت دوم ربات‌ها برای جابجایی اشیا، هماهنگ عمل می‌کنند. هماهنگی برای حمل یک شی، الهام گرفته از حمل گروهی غذا توسط مورچه‌ها می‌باشد [۹۵]. در صورتی که ربات‌ها برای بردن غذا به لانه همکاری کنند، از ایجاد تداخل و برخورد آن‌ها کاسته می‌شود. به عنوان مثال ممکن است ربات‌ها بخواهند جعبه‌ای را هل دهند که یک ربات به تنهایی قادر به هل دادن آن نیست. یکی از نکات قابل توجه در این وظیفه، جهت هل دادن

---

[۱] Gradient-based

[۲] Area-sweeping

[۳] Adaptive

ربات‌ها است که اگر هماهنگ نباشد ممکن است نیروی یکدیگر را خنثی نمایند. ممکن است یک جهت از پیش تعیین شده به ربات‌ها داده شود یا اینکه از نیروهای جاذبه‌ی تعریف شده استفاده گردد. در مرجع [۹۶] در مکان مقصد، لامپی روشن است و ربات‌ها قادرند این لامپ روشن را دیده و به سمت آن حرکت نمایند. در مرجع [۹۷] ربات‌ها سیستم بینایی دارند که آن‌ها را قادر به دیدن مکان مقصد می‌کند؛ ارتفاع جعبه به گونه‌ای است که جلوی دید ربات‌ها را نمی‌گیرد.

ربات‌ها می‌توانند با ایجاد یک آرایش زنجیره‌ای و دست به دست کردن غذا، آن را به لانه برسانند[۱]. در این حالت، ربات‌های حامل غذا با نشانه‌ای مثل چراغ روشن، وضعیت خود را به بقیه اطلاع می‌دهند [۹۸]. در مرجع [۹۹] اگر رباتی که باری بر دوش دارد با رباتی روبرو شود که باری ندارد، غذا را بر زمین گذاشته و در خلاف جهت قبلی شروع به حرکت می‌نماید؛ در واقع از ربات دیگر می‌خواهد که کار او را ادامه دهد. راه دیگر همکاری، تقسیم وظایف است؛ ربات‌ها را می‌توان به دو گروه تقسیم نمود: گروهی در اطراف لانه فعالیت کنند و گروهی خارج از محدوده‌ی اطراف لانه. ربات‌های گروه دوم پس از یافتن غذا آن را به محدوده‌ی مرزی آورده و دسته‌ی اول آن را به لانه می‌برند [۱۰۰]. ممکن است هر ربات به محدوده‌ی خاصی انتساب داده شده و تنها در آن محدوده به دنبال غذا بگردد [۱۰۱].

روش‌های مدل‌سازی رایج وظیفه‌ی کاوش هدف، ماشین حالت متناهی، طراحی مبتنی بر فیزیک مجازی و مسیریابی شبکه[۲] است. به منظور ارزیابی عملکرد سیستم در کاوش هدف می‌توان از تعداد اشیای جمع شده در زمان مشخص [۱۰۲–۱۰۴] یا زمان لازم برای جمع آوری تعداد اشیای مشخص [۹۴] استفاده نمود. تعداد ربات‌ها، اندازه‌ی محیط، چگالی ربات‌ها، چگالی اهداف و محدوده‌ی ارتباطی بین ربات‌ها تعدادی از پارامترهایی هستند که بر نتیجه‌ی بدست آمده تاثیر خواهند گذاشت. در اغلب مطالعات انجام شده، به منظور تحلیل کارایی سیستم کاوش هدف از ربات‌های واقعی یا شبیه‌سازی‌های کامپیوتری استفاده شده است. در صورتی که معمولا به دلیل پیچیدگی ذاتی این وظیفه، مدل‌سازی ریاضی کاوش هدف صورت نگرفته است. با این حال تعدادی استثنا نیز وجود دارد. به عنوان مثال مرجع [۸۶]، سناریوی مطرح شده توسط [۹۸] را با استفاده از شبکه‌ی پتری آماری و زنجیره‌ی مارکو مدل کرده است. مرجع [۵۱] نیز از روش زنجیره‌ی مارکو برای مدل کردن وظیفه‌ی کاوش هدف در سطح ماکروسکپی استفاده کرده است.

مرجع [۱۰۵] یک فریمورک کلی برای مدل کردن کاوش جانوران با استفاده از تابع ارزیابی تولید کرده است. مراجع [۱۰۶, ۱۰۷] یک مدل ماکروسکوپی برای مسئله کاوش هدف در دو حالت بدون ارتباط و با ارتباط ارائه داده‌اند. مرجع [۷۵] نیز یک مدل احتمالاتی ماکروسکوپی برای کاوش هدف ارائه داده

---

[۱] Bucket Brigading

[۲] Network Routing

است. در این مقاله از یک ماشین حالت احتمالاتی و چند معادله ارزیابی استفاده شده است. معادلات مشتق جزیی برای مدل‌سازی کاوش هدف نیز در [۱۰۸] مورد استفاده قرار گرفته‌اند.

## ۲-۵- جمع‌بندی

در این فصل به کارهای پیشین انجام شده در فازهای تحلیل، طراحی و مدل‌سازی پرداخته شد. سپس کارهای ارائه شده برای پیاده‌سازی وظیفه کاوش هدف مرور شد. اغلب سیستم‌های موجود، روال واحد و پیوسته‌ای از تحلیل تا پیاده‌سازی ارائه نکرده‌اند. آن‌ها برای هر مرحله از توسعه سیستم از یک ابزار مجزا بهره برده‌اند. این امر پیچیدگی توسعه سیستم را افزایش می‌دهد که خود منجر به افزایش هزینه نیز خواهد شد. برای مثال اعتبارسنجی روش‌های فرمال با استفاده از روش‌هایی مثل بررسی مدل یک هزینه اضافی به سیستم تحمیل می‌کند. گذار از مرحله مدل‌سازی به مرحله پیاده‌سازی در روش‌های فرمال نیز یک چالش جدی به حساب می‌آید. روشی برای اعتبارسنجی و همخوانی کدهای تولید شده با مدل‌های پایه وجود ندارد. از طرفی روش‌های غیرفرمال، هیچ تضمینی برای درستی تحلیل و پیاده‌سازی ارائه نمی‌کنند و تنها راه بررسی درستی عملکرد سیستم، انجام آزمون‌های تجربی است؛ برای یک سیستم بزرگ و پیچیده، آزمون‌های تجربی در صورت امکان‌پذیر بودن، کافی نبوده و ریسک بالایی دارد.

بنابراین وجود ابزاری که بتوان در آن تمامی مراحل توسعه سیستم از طراحی تا مدل‌سازی و سپس تولید کد را به صورت یکپارچه و واحد صورت داد، کمبود بزرگی است که احساس می‌شود. این ابزار باید قادر به اعتبارسنجی فرمال طراحی‌های صورت گرفته و تولید خودکار کدهای متناظر با طراحی‌ها باشد.

# فصل

# ۳- تئوری کنترل سوپروایزری

## ۳-۱- مقدمه

روش پیشنهادی در این رساله، تئوری کنترل سوپروایزری را در رباتیک جمعی به کار گرفته است. بنابراین در این فصل به تشریح کامل SCT پرداخته شده است؛ پس از معرفی SCT و با تاکید بر کمبودهای آن، نسخه احتمالاتی و زمان‌دار SCT پیشنهاد گشته است. همچنین در این بخش، به مراحل مورد نیاز برای طراحی یک سیستم با استفاده از SCT اشاره شده و انواع روش‌های سنتز تشریح گشته است.

## ۳-۲- تئوری کنترل سوپروایزری

تئوری کنترل سوپروایزری یک فریمورک تئوری برای سنتز کنترل‌کننده‌ها است که به آن‌ها سوپروایزر می‌گویند. فرض می‌شود که سیستم مورد بررسی را می‌توان به صورت یک سیستم رخداد گسسته[۱] (DES) بیان کرد. یک سیستم رخداد گسسته از تعدادی حالت گسسته تشکیل شده و جابجایی بین حالات توسط رخدادها صورت می‌گیرد. جابجایی بین حالت‌ها گذار نام دارد. SCT بین رخدادهای قابل‌کنترل و رخدادهای غیرقابل‌کنترل تمایز قائل می‌شود. رخدادهای غیرقابل‌کنترل، سیگنال‌های واکنشی همچون سیگنال ارسال شده توسط حسگرها را نشان می‌دهند. رخدادهای قابل‌کنترل در واقع همان سیگنال‌های فرمان هستند که توسط کنترل‌کننده ایجاد می‌شود (مانند فرمان حرکت به جلو و یا چرخش به چپ).

در SCT طراح دو مسئله را مدل می‌کند:

۱. سیستم چه کاری می‌تواند انجام دهد
۲. سیستم چه کاری را باید انجام دهد

در بخش اول توانایی‌های سیستم توسط تعدادی مدل رفتار آزاد مشخص می‌شوند. در بخش دوم ضابطه‌های کنترلی تعیین می‌گردند. هر دو بخش مذکور توسط یک زبان فرمال تعریف می‌شوند. زبان مجموعه‌ای از کلمات است و کلمات نتیجه‌ی الحاق الفبای زبان می‌باشند. هر حرف الفبا به یک رخداد در DES نسبت داده می‌شود؛ بنابراین دنباله‌ی مطلوب رخدادها، کلمات زبان را تشکیل می‌دهند. SCT همه‌ی مدل‌های رفتار آزاد و مشخصات کنترلی را در یک زبان ترکیب می‌نماید. ناظر SCT با محدود کردن مجموعه‌ای از رخدادهای قابل‌کنترل که سیستم انتخاب می‌نماید؛ تضمین می‌کند که در هر زمان تنها کلمات معتبر یا پیش‌وند کلمات معتبر اتفاق بیفتد.

[۱] Descrete Event System

فرض کنید رباتی می‌خواهد یک شیشه شیر را از یخچال بردارد. ربات در ابتدا رخداد قابل‌کنترل "باز کردن در یخچال" را انتخاب می‌کند. حال اگر رخداد غیرقابل‌کنترل "شیر در یخچال وجود دارد" اتفاق بیفتد؛ SCT مجموعه رخدادها را به "شیر را از یخچال بردار" و "درب یخچال را ببند" محدود می‌کند. در صورتی که رخداد غیرقابل‌کنترل "شیر در یخچال وجود ندارد" اتفاق بیفتد؛ SCT مجموعه رخداد را به "درب یخجال را ببند" محدود می‌کند. در هر دو سناریو، ربات تا بسته شدن درب یخچال کار دیگری نمی‌تواند انجام دهد [۴۹].

### ۳-۲-۱- مولدها

زبان‌های منظم[۱] رایج‌ترین زبان‌های استفاده شده در SCT هستند که تحت عنوان زبان‌های نوع سوم نیز شناخته می‌شوند. کلماتِ زبان منظم را می‌توان توسط مولد[۲] ایجاد نمود. مولد، ساختاری شبیه به آتاماتای متناهی داشته و ماشین حالت متناهی[۳] خوانده می‌شود. با این‌حال آتاماتا کلمات یک زبان را با «پذیرش» یا «رد» آن‌ها تشخیص می‌دهد ولی مولد کلمات مورد پذیرش یک زبان را تولید می‌نماید. یک مولد $G$ به صورت $(Q, \Sigma, \delta, q_0, Q_m)$ تعریف می‌شود که در آن $Q$ مجموعه‌ی حالات، $\Sigma$ الفبای زبان، $\delta$ تابع انتقال است که به صورت ( $\delta : Q \times \Sigma \to Q$) تعریف می‌شود. $q_0 \in Q$ حالت اولیه و $Q_m \subseteq Q$ مجموعه حالات علامت‌دار هستند. حالات علامت‌دار به حالت‌های امن سیستم گفته می‌شود. به عنوان مثال پایان یک وظیفه را می‌توان با حالت علامت‌دار نشان داد ولی این حالت‌ها به معنی پایان عملیات نبوده و ممکن است سیستم پس از رسیدن این حالات به کار خود ادامه دهد. زبان تولید شده توسط مولد $G$ را به صورت $L(G)$ نمایش می‌دهند. رخدادهای سیستم که معادل الفبای زبان هستند به دو نوع قابل‌کنترل ($\Sigma_c$) و غیرقابل‌کنترل ($\Sigma_u$) تقسیم می‌شوند، به صورتی که $\Sigma = \Sigma_c \cup \Sigma_u$ و $\Sigma_c \cap \Sigma_u = \emptyset$ . رخداد قابل‌کنترل $e_c \in \Sigma_c$ در حالت $q \in Q$ تنها در صورتی فعال می‌شود که تابع $\delta(q, e_c)$ تعریف شده باشد. $\Sigma^*$ مجموعه‌ی تمامی کلمات ممکن (دنباله‌ی رخدادهای ممکن) ساخته شده از الفبای زبان سیستم است. $\Sigma^+$ برابر با تمامی کلمات ممکن به جز رشته‌ی تهی است ($\Sigma^+ = \Sigma^* - \{\lambda\}$).

### ۳-۲-۲- مدل رفتار آزاد

در تئوری کنترل سوپروایزری، هر یک از توانایی‌های فیزیکی سیستم با استفاده از یک مدل رفتار آزاد بیان می‌شود. در نتیجه برای یک سیستم، $m$ مولد $G_i$ ( $i \in \{1,2,..m\}$) تعریف شده که هر مولد یک مدل رفتار آزاد را توصیف می‌کند. به صورت پیش‌فرض، مدل‌های رفتار آزاد را مستقل از یکدیگر در نظر می‌گیرند. نوار نقاله‌ای را در یک کارخانه‌ی تولیدی در نظر بگیرید که محصولات را جابجا می‌کند.

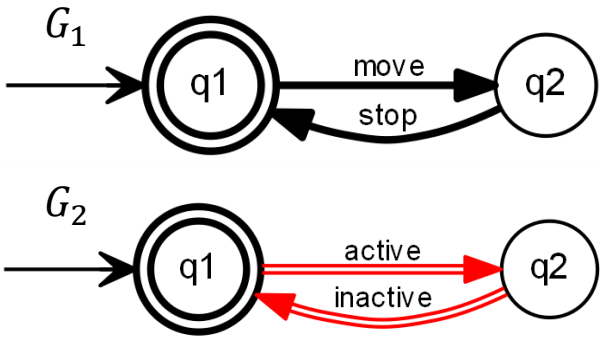
---

[۱] Regular Languages

[۲] Generator

[۳] Finite State Machine

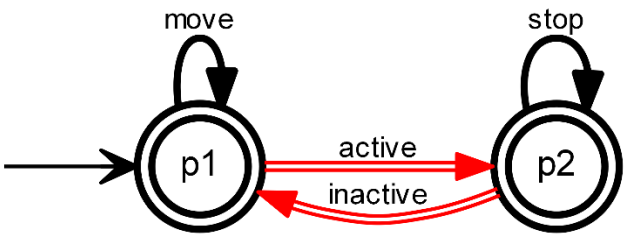
در کنار نوار نقاله، حسگری وجود دارد که وجود و عدم وجود محصول روی نوار نقاله را تشخیص می‌دهد. شکل ۳-۱ مدل رفتار آزاد برای نوار نقاله و حسگر را نشان می‌دهد. رخدادهای *move* و *stop* قابل‌کنترل و رخدادهای *active* و *inactive* غیرقابل‌کنترل هستند. معمولا تنها حالت اولیه‌ی مدل را علامت‌دار قرار می‌دهند. این کار به معنی برگشت سوپروایزر به شرایط اولیه است. کمان‌های یک‌خطی و دوخطی (قرمز رنگ) به ترتیب نشانگر رخدادهای قابل‌کنترل و غیرقابل‌کنترل هستند. حالت‌های علامت‌دار با دایره‌های دوخطی مشخص شده‌اند.




**شکل ۳-۱: مثالی از مدل رفتار آزاد نوار نقاله (G1) و حسگر (G2).**

### ۳-۲-۳- ضابطه‌های کنترلی

رفتار مطلوب سیستم با $n$ ضابطه کنترلی نمایش داده می‌شود. هر ضابطه‌ی کنترلی $E_j$ $(j \in \{1,2,..n\})$ ارتباط دو یا چند مدل رفتار آزاد را نشان می‌دهد. شکل ۳-۲ ضابطه‌ی تعیین کننده‌ی ارتباط بین مدل‌های نوار نقاله و حسگر را نمایش می‌دهد. قانون موجود در این ضابطه به صورت «هنگامی که یک محصول در مقابل حسگر قرار گیرد، نوار نقاله متوقف شده و در غیر این‌صورت باید حرکت نماید» است. SCT از وقوع برخی رخدادهای قابل‌کنترل در بعضی از حالت‌ها جلوگیری می‌کند. در مثال نوار نقاله و حسگر، در حالت p1 رخداد *stop* غیرفعال و رخداد *move* فعال است. بنابراین هنگامی که حسگر غیرفعال است، نوار حرکت می‌نماید.




**شکل ۳-۲: ضابطه کنترلی که موجب می‌شود نوار نقاله تنها در صورتی حرکت کند که محصولی در مقابل حسگر نباشد.**

### ۳-۲-۴- محاسبه سوپروایزر

برای بدست آوردن سوپروایز، مولدهای مربوط به مدل‌های رفتار آزاد و ضابطه‌های کنترلی با هم ترکیب می‌شوند. در فرآیند ترکیب، مولد مربوط به مدل رفتار آزاد با توجه به ضابطه‌های کنترلی محدود می‌گردد. در واقع ربات تنها قادر به اجرای اعمالی است که توسط ضابطه‌های کنترلی مجاز اعلام گشته است.

برای ترکیب دو مولد $G_a$ و $G_b$ با الفبای $\sum_i,\ i \in \{a,b\}$ از عملگر ترکیب موازی[۱] به صورت نشان داده شده در فرمول (۱) استفاده می‌شود. این عملگر با نماد $\|$ نمایش داده شده است.

$$G_a \parallel G_b = (Q_a \times Q_b,\ \sum_a \cup \sum_b, \delta_{a\|b}, (q_{0_a}, q_{0_b}), Q_{m_a} \times Q_{m_b}) \qquad (۱)$$

تابع انتقال $\delta_{a\|b}$ به صورت زیر محاسبه می‌گردد.

$$\delta_{a\|b}\big((q_a,q_b),e\big) = \begin{cases} \big(\delta_a(q_a,e),\delta_b(q_b,e)\big) & if\ \delta_a(q_a,e)\ and\ \delta_b(q_b,e)\ defined \\ (\delta_a(q_a,e),q_b) & if\ \delta_a(q_a,e)\ defined\ and\ e \notin \sum_b \\ \big(q_a,\delta_b(q_b,e)\big) & if\ \delta_b(q_b,e)\ defined\ and\ e \notin \sum_a \\ undefined & otherwise \end{cases} \qquad (۲)$$

فرمول (۲) اجازه می‌دهد رخدادهایی که بین دو مولد مشترک نیستند، به صورت غیرهمزمان عمل کنند. با ترکیب موازی مدل رفتار آزاد $G$ و ضابطه کنترلی $E$، مولد زبان هدف $(K)$ یا سوپروایزر حاصل می‌شود:

$$K = G \parallel E \qquad (۳)$$

در این مرحله زبان $L(K)$ لزوما کنترل‌شونده نیست. زبان $K$ با الفبای $\sum$ و مجموعه رخدادهای غیرقابل کنترل $\sum_u \subseteq \sum$ در صورتی کنترل‌شونده است که [۱۰۹]:

$$\forall s \in \overline{L(K)}, \forall e_u \in \sum_u, se_u \in L(G) \rightarrow se_u \in \overline{L(K)} \qquad (۴)$$

نماد $\bar{L}$ بیانگر زبان پیشوند بسته شده‌ی زبان $L$ است و به صورت زیر تعریف می‌شود:

$$\bar{L} = \{s \in \sum^* : \exists t \in \sum^* \wedge st \in L\} \qquad (۵)$$

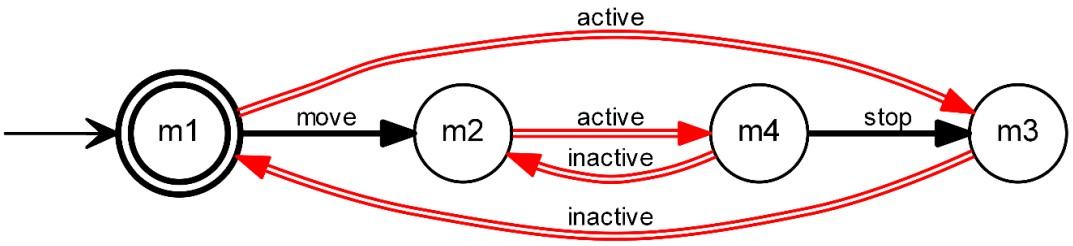
به بیانی دیگر، اگر $s$ پیشوند یکی از رشته‌های تولید شده توسط مولد $K$ باشد و $e_u$ یک رخداد غیرقابل کنترل که از نظر فیزیکی امکان رخ دادن آن وجود دارد $(se_u \in L(G))$، آنگاه زبان $L(K)$ در صورتی کنترل شونده است که رشته $se_u$ نیز پیشوند یکی از رشته‌های تولید شده توسط مدل $K$ باشد. در

[۱] Parallel Composition

صورتی که اینچنین نباشد، زبان $L(K)$ غیرقابل کنترل و حالتی که در آن رخداد غیرقابل کنترل $e_u$ رخ داده است، حالت بد نامیده می‌شود. برای استخراج زیرمجموعه‌ای از زبان $L(K)$ که کنترل‌شونده است، باید همه حالت‌های بد و سایر حالت‌هایی که توسط رخدادهای غیرقابل کنترل به حالت‌های بد می‌رسند، حذف گردند. به زبان حاصل، بزرگترین زیرمجموعه کنترل‌شونده زبان $L(K)$ گفته می‌شود [۱۱۰].

پس از بدست آوردن زبان کنترل شونده، نوبت به حذف بن‌بست‌ها می‌رسد. در صورتی که زبان دارای یکی از دو حالت بن‌بست و حلقه بینهایت[۱] باشد، بلاک‌شونده است. برای حذف بن‌بست‌ها، حالت‌های غیرقابل دسترس از حالت ابتدایی باید حذف شوند. علاوه‌براین، حالت‌هایی که مسیری بین آن‌ها و حداقل یکی از حالت‌های علامت‌دار وجود ندارد نیز باید حذف شوند. حذف این دو مجموعه از حالت‌ها، بلاک‌شونده نبودن زبان $L(K)$ را تضمین می‌کند [۱۱۰]. بنابراین، سوپروایزر نهایی به صورت نشان داده شده در فرمول (۶) بدست می‌آید. عملگر $SupC$، با توجه به مدل رفتار آزاد $G$، ابتدا حالت‌های بد و سپس حالت‌های غیرقابل دسترس را از مولد $K$ حذف می‌کند. مولد نهایی حاصل $S$ کنترل‌شونده و بلاک‌نشونده است. شکل ۳-۳، سوپروایزر محاسبه شده از سنتز مدل‌های رفتار آزاد شکل ۳-۱ و ضابطه‌ی کنترلی شکل ۳-۲ را با توجه به مراحل تشریح شده، به نمایش گذاشته است. همانطور که مشاهده می‌شود، سوپروایزر بدست آمده، توانایی‌های فیزیکی سیستم را با ضابطه‌های کنترلی موردنظر بدرستی ترکیب کرده است و اجازه تولید دنباله‌های نادرست رخدادها را نمی‌دهد.

$$S = SupC(K, G) \qquad (۶)$$




**شکل ۳-۳: سوپروایزر محاسبه شده از سنتز مدل‌های رفتار آزاد شکل ۳-۱ و ضابطه‌ی کنترلی شکل ۳-۲.**

تا اینجا، نحوه محاسبه سوپروایزر تشریح شد. در صورتی که تعداد مولدهای مدل رفتار آزاد و یا ضابطه‌های کنترلی بیش از یکی باشد، سه رویکرد برای محاسبه سوپروایزر مطرح می‌شود [۹۰]. در ادامه سه روش مربوطه شرح داده شده است. بر اساس نتایج و مقایسه‌های ارائه شده در مرجع [۴۹]،

[۱] Livelock

روش سوم از جهت اندازه ماشین حالت سوپروایزر، بهتر از دو روش دیگر عمل می‌کند. به همین دلیل، در این رساله نیز از این روش برای سنتز سوپروایزر استفاده شده است.

### ۳-۲-۴-۱- سوپروایزر یکپارچه

اگر همه مولدهای مربوط به مدل‌های رفتار آزاد و ضابطه‌های کنترلی باهم ترکیب شده و نتیجه یک سوپروایزر واحد گردد؛ سوپروایزر حاصل، یکپارچه نامیده می‌شود. این روش در چهار مرحله صورت می‌گیرد. ابتدا همه $m$ مدل رفتار آزاد با هم ترکیب شده و یک مولد مدل رفتار آزاد واحد $G^{mon}$ را تولید می‌کنند (فرمول (۷)). سپس، $n$ ضابطه کنترلی با هم ترکیب شده و یک مولد ضابطه کنترلی واحد $E^{mon}$ را تولید می‌کنند (فرمول (۸)). آنگاه مولد $K^{mon}$ محاسبه شده (فرمول (۹)) که پس از اعمال عملگر $SupC$ منجر به تولید سوپروایزر نهایی $S^{mon}$ خواهد شد (فرمول (۱۰)).

$$G^{mon} = G_1 \parallel \cdots \parallel G_m \tag{۷}$$

$$E^{mon} = E_1 \parallel \cdots \parallel E_n \tag{۸}$$

$$K^{mon} = G^{mon} \parallel E^{mon} \tag{۹}$$

$$S^{mon} = SupC(K^{mon}, G^{mon}) \tag{۱۰}$$

### ۳-۲-۴-۲- سوپروایزر ماژولار

به دلیل استفاده از عملگر ترکیب موازی، تعداد حالت‌های مولد حاصل از ترکیب، ممکن است بسیار بزرگ باشد. در نتیجه، انتقال چنین ماشین حالتی به ربات‌ها، نیاز به مقدار زیادی حافظه دارد که در بسیاری از موارد، امکان‌پذیر نیست. برای رفع این مشکل، ایده سوپروایزرهای ماژولار مطرح شد [۱۱۰]. این روش به ازای هر ضابطه‌ی کنترلی، یک سوپروایزر بدست می‌آورد که به صورت موازی اجرا می‌شوند. برای اعمال این روش، مانند روش قبل، ابتدا همه $m$ مدل رفتار آزاد با هم ترکیب شده و یک مولد مدل رفتار آزاد واحد $G^{mod}$ را تولید می‌کنند (فرمول (۱۱)). سپس مولدهای $K_i^{mod}$ به ترتیب برای تک‌تک ضابطه‌های کنترلی $E_i$ محاسبه می‌شود (فرمول (۱۲)). در نهایت، پس از اعمال عملگر $SupC$ به هر مولد $K_i^{mod}$، سوپروایزر نهایی $S_i^{mod}$ حاصل خواهد شد. (فرمول (۱۳)).

$$G^{mod} = G_{mon} = G_1 \parallel \cdots \parallel G_m \tag{۱۱}$$

$$K_i^{mod} = G^{mod} \parallel E_i \quad \forall\, i \in \{1, \dots, n\} \tag{۱۲}$$

$$S_i^{mod} = SupC\left(K_i^{mod}, G^{mod}\right) \quad \forall\, i \in \{1, \dots, n\} \tag{۱۳}$$

برای استفاده از این روش، ضابطه‌های کنترلی نباید با یکدیگر تداخل داشته باشند. دو مولد $S_1$ و $S_2$ را متداخل می‌نامیم اگر مولد حاصل از ترکیب موازی آن‌ها دارای یکی از انواع حالت‌های غیرقابل دسترس باشد. یک روش برای رفع تداخل، حذف دو مولد مربوطه و جایگزین کردن مولد $S_1 \parallel S_2$ با آن دو است [۹۰].

### ۳-۲-۴-۳- سوپروایزر ماژولار محلی

این روش هم در سطح مدل‌های رفتار آزاد و هم در سطح ضابطه‌های کنترلی به صورت ماژولار عمل می‌کنند. در روش قبل، هر ضابطه‌ی کنترلی با کل مدل‌های رفتار آزاد ترکیب می‌شد. در این روش، هر ضابطه کنترلی تنها با مدل‌های رفتار آزاد مرتبط با خودش، ترکیب می‌شود (اشتراک مجموعه رخدادهایشان تهی نباشد). استفاده از این روش نیز به عدم تداخل ضابطه‌های کنترلی وابسته است. برای محاسبه سوپروایزرهای ماژولار محلی، ابتدا مدل‌های رفتار آزاد مرتبط با هر ضابطه‌ی کنترلی $E_i$ با هم ترکیب شده و مدل رفتار آزاد واحد $G_i^{loc}$ را بدست می‌دهند (فرمول (۱۴)). سپس مولدهای $K_i^{loc}$ به ترتیب برای تک‌تک ضابطه‌های کنترلی $E_i$ محاسبه می‌شوند (فرمول (۱۵)). در نهایت، پس از اعمال عملگر $SupC$ به هر مولد $K_i^{loc}$، سوپروایزر نهایی $S_i^{loc}$ بدست خواهد آمد. (فرمول (۱۶)).

$$G_i^{loc} = G_{k_1} \parallel \cdots \parallel G_{k_T} \quad \forall i \in \{1, \dots, n\}, \forall k_i \in \{1, \dots, m\} \wedge (\sum_{G_{k_i}} \cap \sum_{E_i}) \neq \emptyset \tag{۱۴}$$

$$K_i^{loc} = G_i^{loc} \parallel E_i \quad \forall i \in \{1, \dots, n\} \tag{۱۵}$$

$$S_i^{loc} = SupC\left(K_i^{loc}, G_i^{loc}\right) \quad \forall i \in \{1, \dots, n\} \tag{۱۶}$$

این روش نیاز به یک اعتبارسنجی اضافه نسبت به روش یکپارچه دارد؛ نتیجه ترکیب موازی همه سوپروایزرهای ماژولار محلی باید با سوپروایزر یکپارچه بدست آمده از روش قبل، برابر باشد (فرمول (۱۷)) [۴۹]. در غیراینصورت باید طراحی صورت گرفته را بازنگری کرد تا به نتیجه دلخواه رسید.

$$S^{mon} = S_1^{loc} \parallel \cdots \parallel S_i^{loc} \quad \forall i \in \{1, \dots, n\} \tag{۱۷}$$

## ۳-۳- تئوری کنترل سوپروایزری احتمالاتی و زمان‌دار

در طراحی ایده‌آل یک سیستم، هیچ حالتی نباید دارای بیش از یک رخداد قابل‌کنترل فعال[۱] باشد. در این صورت، به ازای هر ورودی، تنها یک خروجی داریم. در طراحی واقعی، چنین چیزی همیشه برقرار

[۱] Enabled

نیست، زیرا ضابطه‌های کنترلی لزوما همه ترکیبات ممکن رخدادها را محدود نمی‌کنند [۱۱۱]. بنابراین در عمل، ماشین‌های حالتی با بیش از یک رخداد قابل کنترل فعال در یک حالت خواهیم داشت. در این مواقع کدامیک از رخدادها باید اجرا گردند؟ این مورد یکی از محدودیت‌های ماشین‌های حالت متناهی مورد استفاده در مولدها است. عدم پشتیبانی از رخدادهای احتمالاتی از تعیین اولویت بین رخدادها جلوگیری می‌کند؛ برای مثال، ممکن است در یک مسئله نیاز به اعمال اولویت بیشتر به حرکت روبه‌جلو نسبت به چرخش به طرفین وجود داشته باشد؛ و یا چرخش به یکی از طرفین بیشتر از دیگری رخ دهد. با استفاده از رخدادهای احتمالاتی می‌توان در مرحله طراحی و به صورت فرمال، عدم قطعیت رخدادها را در نظر گرفته و اعمال کرد.

محدودیت دوم ماشین‌های حالت متناهی، عدم پشتیبانی از زمان است. با اضافه کردن زمان به رخدادها، می‌توان دسته‌ی بزرگتری از مسائل را حل نمود. در تئوری کنترل سوپروایزری زمان‌دار، یک ساعت سراسری در سیستم وجود دارد که مستقل از کنترل کننده‌ها عمل می‌کند [۱۱۲]. این ساعت برای اندازه‌گیری گذشت زمان مورد استفاده قرار می‌گیرد. برای هر گذار دو زمان قابل تعیین است: «کران پایین» و «کران بالا». کران پایین، تعیین کننده مقدار تاخیر اعمال شده به زمان جاری سیستم برای اجرای رخداد قابل‌کنترل متعلق به گذار موردنظر است. در ماشین حالت متناهی زمان‌دار، به هر رخداد یک کران پایین و یک کران بالای زمان (حتی بی‌نهایت) نسبت داده می‌شود. رخداد زمان‌دار نمی‌تواند قبل از فرارسیدن کران پایینِ زمانش، فعال شود. کران بالا، زمان انقضای رخداد را مشخص می‌کند. یک رخداد پس از رسیدن به کران بالای زمانش، قابل فعال شدن نیست (منقضی می‌شود). احتمال و زمان تنها مختص به رخدادهای قابل‌کنترل هستند.

در ادامه به تعریف فرمال تئوری کنترل سوپروایزری احتمالاتی و زمان‌دار (ptSCT) پرداخته شده است. مولد مربوط به ptSCT به صورت $(Q, \Sigma, \delta, q_0, Q_m, P, T)$ تعریف می‌شود. تابع $P$ بیانگر احتمال رویداد رخدادها در هر گذار و تابع $T$ بیانگر کران پایین و بالای زمان مربوط به رخدادها هستند: $P : Q \times \Sigma \to [0,1]$ و $T : Q \times \Sigma \to [l,u]$. $l$ کران پایین و $u$ کران بالای گذار مربوطه را نشان می‌دهد: $l \in \mathbb{N}, u \in \mathbb{N} \cup \{\infty\}$. مجموعه رخدادها نیز شامل یک رخداد جدید است که همان تیک ساعت سراسری سیستم می‌باشد؛ بنابراین $\Sigma = \Sigma_{SCT} \cup \{tick\}$. این رخداد مستقل از کنترل‌کننده‌ها در هر واحد زمانی رخ می‌دهد. درضمن، مجموع احتمالات خروجی از هر حالت برابر با یک است: $\forall q \in Q, \sum_{e\in\Sigma} p(q,e) = 1$.

استفاده از کران بالا، علاوه‌بر افزایش توانایی حل مسائل دشوارتر، پیچیدگی را نیز افزایش می‌دهد. از طرفی، استفاده از کران بالا در مسائلی که مبتنی بر فعالیت‌ها هستند، از نظر معنایی منطقی است؛ زیرا یک فعالیت قابل منقضی شدن است؛ و ممکن است اگر در یک بازه زمانی مشخص اجرا نگردد، ارزش

خود را از دست بدهد. در حالی که رخدادها اینچنین نیستند [۱۱۳]. با توجه به دلایل نامبرده شده، در روش پیشنهادی، با در نظر گرفتن $u = \infty$ برای همه گذارها، با پذیرش کاهش نسبی توان حل مسئله، پیچیدگی تا حد بسیار زیادی کاهش پیدا می‌کند. ptSCT یک نسخه عمومی‌تر از SCT است که به ترتیب با صفر و یک قرار دادن کران پایین و احتمال همه رخدادها حاصل می‌شود. با ساده‌سازی صورت گرفته، پیچیدگی زمانی محاسبه ptSCT برابر با SCT خواهد بود.

هر گذار دارای یک شمارنده‌ی داخلی است که از زمان ورود به حالت مربوطه، شروع به شمارش تعداد تیک‌های ساعت سراسری می‌نماید. در صورتی که کران پایین گذار موردنظر از مقدار شمارنده داخلی بیشتر (یا مساوی) باشد، گذار مربوطه قابل اجرا شدن خواهد بود. شمارنده‌ی داخلی هر گذار در دو حالت، از نو راه‌اندازی می‌شود: اول) حالت جاری تغییر کند و دوم) گذار مربوطه اجرا گردد.

### ۳-۳-۱- محاسبه سوپروایزر

در این بخش، محاسبه سوپروایزر احتمالاتی و زمان‌دار تشریح شده است؛ بنابراین از تکرار مباحث مشترک مطرح شده برای محاسبه سوپروایزر پایه (بدون احتمالات و زمان) اجتناب شده است. برای ترکیب دو مولد احتمالاتی و زمان‌دار $G_a$ و $G_b$ با الفبای $\sum_i,\ i \in \{a,b\}$ از عملگر ترکیب موازی به صورت نشان داده شده در فرمول (۱۸) استفاده می‌شود.

$$G_a \parallel G_b = (Q_a \times Q_b,\ \sum_a \cup \sum_b, \delta_{a\parallel b}, (q_{0_a}, q_{0_b}), Q_{m_a} \times Q_{m_b}, P_{a\parallel b}, T_{a\parallel b}\ ) \quad (۱۸)$$

نحوه محاسبه احتمال و زمان رخداد گذارهای مربوط به رخدادهای قابل‌کنترل به ترتیب در فرمول‌های (۱۹) و (۲۰) ارائه شده‌اند.

$$P_{a\parallel b}((q_a,q_b),e_c) = \begin{cases} P_a(q_a,e_c) \times P_b(q_b,e_c) & if\ \delta_a(q_a,e)\ and\ \delta_b(q_b,e)\ defined \\ P_a(q_a,e_c) & if\ \delta_a(q_a,e)\ defined\ and\ e \notin \sum_b \\ P_b(q_b,e_c) & if\ \delta_b(q_b,e)\ defined\ and\ e \notin \sum_a \\ 0 & otherwise \end{cases} \quad (۱۹)$$

$$T_{a\parallel b}((q_a,q_b),e_c) = \begin{cases} max(T_a(q_a,e_c),T_b(q_b,e_c)) & if\ \delta_a(q_a,e)\ and\ \delta_b(q_b,e)\ defined \\ T_a(q_a,e_c) & if\ \delta_a(q_a,e)\ defined\ and\ e \notin \sum_b \\ T_b(q_b,e_c) & if\ \delta_b(q_b,e)\ defined\ and\ e \notin \sum_a \\ 0 & otherwise \end{cases} \quad (۲۰)$$

محاسبه تابع انتقال مولد احتمالاتی و زمان‌دار و سایر مراحل محاسبه سوپروایزر مشابه با نسخه‌ی پایه می‌باشد. یک فاکتور مهم در محاسبه سوپروایزر احتمالاتی، نرمال‌سازی احتمالات خروجی از یک حالت است. وقتی سنتز به صورت یکپارچه صورت می‌گیرد، تنها یک سوپروایزر خواهیم داشت. بنابراین می‌توان در زمان محاسبه سوپروایزر، احتمالات خروجی از همه حالات را نرمال‌سازی کرد. این روش نرمال‌سازی را روش برون‌خط[1] نامگذاری می‌کنیم. اما روال نرمال‌سازی برای سنتز ماژولار محلی متفاوت است. نرمال‌سازی هر سوپروایزر محلی کافی نیست؛ بلکه سوپروایزرهایی که با یک رخداد مشترک عمل می‌کنند نیز باید نسبت به هم نرمال‌سازی شوند. این روش نرمال‌سازی را روش برخط[2] می‌نامیم. این روش باید در مرحله پیاده‌سازی سیستم اعمال گردد که در بخش مربوطه به جزئیات آن اشاره شده است. روش برون‌خط، در مرحله محاسبه سوپروایزر به صورت نشان داده شده در فرمول (۱۹) اعمال می‌گردد. فرمول (۲۰) نحوه محاسبه کران پایین زمان را برای سوپروایزر ارائه می‌کند. زمانی که دو ضابطه‌ی کنترلی، کران‌های پایین متفاوتی را برای یک رخداد یکسان تعیین کرده باشند، کران پایین بزرگتر برای سوپروایزر انتخاب می‌شود. به این ترتیب، حد تعیین شده توسط هر دو ضابطه‌ی کنترلی رعایت خواهد شد.

## ۳-۴- جمع‌بندی

در این فصل به تشریح تئوری کنترل سوپروایزری پرداخته شد. مفهوم مولد توضیح داده شد و شکل توصیف فرمال مدل‌های رفتار آزاد و ضابطه‌های کنترلی بیان شد. عملیات ترکیب موازی معرفی گشت؛ آنگاه پس از معرفی رخدادهای غیرقابل کنترل، عملیات سنتز و تفاوت آن با عملیات ترکیب موازی به روشنی بیان شد. پس از معرفی سنتز، روش‌های مختلف سنتز مورد بررسی قرار گرفت. در پایان، با اضافه کردن دو فاکتور احتمالات و زمان به تئوری کنترل سوپروایزری، نسخه احتمالاتی و زمان‌دار SCT، با نام ptSCT معرفی گشت. با استفاده از ptSCT می‌توان طراحی قدرتمندتر و در عین حال ساده‌تری نسبت به SCT انجام داد. برای نمایش قدرت ptSCT در مقایسه با SCT، در فصل نتایج آزمایش‌ها، آزمایش‌های متعددی ترتیب داده شده است.

[1] Offline
[2] Online

# فصل

# ۴- ابزار جامع پیشنهادی برای طراحی، مدل‌سازی و تولید خودکار کد

## ۴-۱- مقدمه

در این فصل به تشریح ابزار جامع پیشنهادی برای طراحی، مدل‌سازی و تولید خودکار کد پرداخته شده است. این فصل مقدمه فصل بعدی است که در آن، با استفاده از ابزار تولید شده در این بخش، به طراحی و پیاده‌سازی وظیفه کاوش هدف پرداخته خواهد شد.

برای تشریح ابزار جامع پیشنهادی، باید ابتدا به نحوه‌ی انتخاب محیط و ربات مناسب برای شبیه‌سازی پرداخته شود. بنابراین، در ابتدای فصل، مسیر طی شده تا رسیدن به گزینه‌های مناسبِ این رساله، توضیح داده شده است. پس از آن، به جزئیات گام اول برای پیاده‌سازی ابزار پیشنهادی پرداخته شده است؛ یعنی پیاده‌سازی ptSCT بر روی محیط شبیه‌سازی منتخب (سکوی آرگوس[۱]). در گام دوم و پس از پیاده‌سازی ptSCT که در واقع نزدیک‌ترین لایه به سخت‌افزار محسوب می‌شود؛ نوبت به لایه‌ی بالایی آن، یعنی ابزاری برای طراحی مدل‌های رفتار آزاد و ضابطه‌های کنترلی می‌رسد. این ابزار باید بتواند تمامی مراحل موردنیاز برای رسیدن از طراحی به پیاده‌سازی را به صورت خودکار انجام دهد. در پایان این فصل، یک وظیفه ساده در رباتیک جمعی، با استفاده از ابزار جامع پیشنهادی، پیاده‌سازی و در محیط شبیه‌سازی اجرا شده است؛ به این ترتیب، با نحوه عملکرد نرم‌افزار آشنا شده و از صحت عملکرد آن نیز اطمینان حاصل خواهد شد.

## ۴-۲- معرفی محیط شبیه‌سازی ARGoS

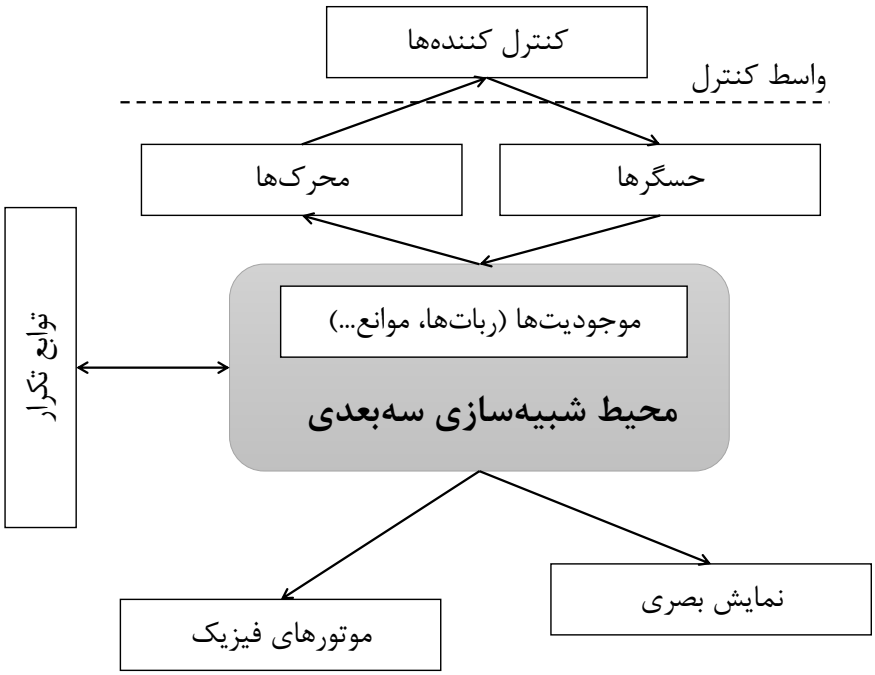
محیط‌های شبیه‌سازی متعددی برای رباتیک جمعی ارائه شده است؛ هرچند همه آن‌ها برای روش پیشنهادی مناسب نیستند. برای مثال برخی از نمونه‌های معرفی شده در فصل ۱، رایگان نبوده [۳۱] و برخی دارای گروه توسعه‌دهندگان فعال نبودند [۳۲]. از طرفی، برخی از محیط‌ها با افزایش تعداد ربات‌ها که لازمه یک سیستم رباتیک جمعی است، دچار کاهش سرعت قابل توجهی می‌شدند [۳۱].

یکی از کم‌نقص‌ترین محیط‌های رایگان ارائه شده برای رباتیک جمعی، سکوی ARGoS می‌باشد [۳۴]. این سکو به صورت متن‌باز ارائه شده و دارای گروه توسعه‌دهندگان فعالی است؛ به شکلی که با گزارش یک اشکال و یا درخواست یک ویژگی مهم، به سرعت به آن رسیدگی کرده و نسخه جدیدی ارائه می‌کنند. این سکو با دو هدف اصلی سرعت بالا و انعطاف‌پذیری عرضه شده است؛ بنابراین با افزایش تعداد ربات‌ها، می‌توان سرعت اجرای شبیه‌سازی را بهینه‌سازی کرد. ویژگی انعطاف‌پذیری اجازه می‌دهد تقریبا همه بخش‌های ARGoS را سفارشی‌سازی کرد؛ از طراحی محیط شبیه‌سازی گرفته، تا عملکرد ربات‌ها و شکل اجرای آزمایش‌ها. ARGoS از انواع مختلفی از ربات‌ها از قبیل E-puck و Kilobot پشتیبانی می‌کند. همچنین می‌توان پشتیبانی از ربات‌های جدید را در قالب پلاگین به آن افزود. یکی

[۱] ARGoS

از قابلیت‌های منحصربفرد ARGoS، امکان تقسیم محیط شبیه‌سازی به بخش‌های مختلف است؛ به شکلی که می‌توان هر بخش را توسط یک موتور فیزیک[1] متفاوت مدیریت کرد.

شکل ۴-۱ معماری ARGoS را به تصویر کشیده است. در ARGoS نه تنها ربات‌ها و موانع قابلیت سفارشی‌سازی دارند، بلکه حتی موتورهای فیزیک نیز قابل ویرایش و حذف و اضافه هستند. یکی از قابلیت‌های منحصربفرد ARGoS عدم‌نیاز به دو پیاده‌سازی مختلف برای شبیه‌سازی و اجرای واقعی است. کد نوشته شده برای شبیه‌سازی، بدون تغییر قابل کامپایل برای ربات‌های واقعی است. این امر یک امتیاز بزرگ به حساب می‌آید؛ زیرا پس از بدست آوردن نتایج دلخواه از مرحله شبیه‌سازی، نیازی به پیاده‌سازی مجدد منطق طراحی شده برای اجرای واقعی نخواهد بود.




**شکل ۴-۱: معماری محیط شبیه‌سازی ARGoS. تمامی جعبه‌های سفید رنگ قابل سفارشی‌سازی هستند [۳۴].**

پیاده‌سازی و اجرای یک پروژه توسط ARGoS از دو بخش تشکیل می‌شود:

۱. **طراحی و تنظیم محیط شبیه‌سازی**: ابعاد محیط، نوع و تعداد ربات‌ها در محیط، حسگرها و محرک‌های موردنیاز، شکل و تعداد موانع در محیط، شکل توزیع ربات‌ها و موانع در محیط و موتورهای فیزیک مورد استفاده برای مدیریت محیط، بخشی از مواردی است که در ARGoS می‌توان برای طراحی محیط مورد استفاده قرار داد.

[1] Physics Engine

۲. **پیاده‌سازی نرم‌افزار کنترل کننده ربات‌ها**: نرم‌افزار کنترل کننده ربات‌ها به زبان C++ پیاده‌سازی می‌شود. برای نوشتن چنین نرم‌افزاری، باید با نحوه عملکرد ربات موردنظر آشنا بوده و ورودی و خروجی حسگرها و محرک‌ها را به خوبی بشناسیم.

در ادامه یک نمونه فایل مربوط به تنظیمات محیط که با فرمت argos. شناخته می‌شود، ارائه شده است (شکل ۴-۲). در این فایل، ابتدا تعدادی حسگر و محرک معرفی شده‌اند؛ سپس ابعاد محیط و دیوارها تعیین شده‌اند؛ آنگاه تعداد ۴۰ ربات و ۲۰ مانع به صورت تصادفی در محیط جاگذاری شده‌اند؛ در پایان موتور فیزیک و نحوه نمایش بصری محیط مشخص شده‌اند. در بخش پیاده‌سازی تنها می‌توان از موجودیت‌ها (ربات، موانع و ...)، حسگرها و محرک‌هایی استفاده کرد که در فایل تنظیمات معرفی شده‌اند.

```
<?xml version="1.0" ?>
<argos-configuration>

  <!-- ************************ -->
  <!-- * General configuration * -->
  <!-- ************************ -->
  <framework>
    <system threads="0" />
    <experiment length="0"
                ticks_per_second="20"
                random_seed="124" />
  </framework>

  <!-- *************** -->
  <!-- * Controllers * -->
  <!-- *************** -->
  <controllers>

    <epuck_obstacle_avoidance_sct_prob_controller id="fdc"
            library="build/controllers/
libepuck_obstacleavoidance_sct.so">
      <actuators>
        <epuck_wheels implementation="default" noise_std_dev="0"/>
```

```
      </actuators>
      <sensors>
        <epuck_proximity implementation="default" show_rays="true"/>
      </sensors>
      <params velocity="12.8"/>
    </epuck_obstacle_avoidance_sct_prob_controller>

  </controllers>

  <!-- ********************** -->
  <!-- * Arena configuration * -->
  <!-- ********************** -->
  <arena size="3, 3, 1" center="0,0,0.5">

    <box id="wall_north" size="3,0.1,0.5" movable="false">
      <body position="0,1.5,0" orientation="0,0,0" />
    </box>
    <box id="wall_south" size="3,0.1,0.5" movable="false">
      <body position="0,-1.5,0" orientation="0,0,0" />
    </box>
    <box id="wall_east" size="0.1,3,0.5" movable="false">
      <body position="1.5,0,0" orientation="0,0,0" />
    </box>
    <box id="wall_west" size="0.1,3,0.5" movable="false">
      <body position="-1.5,0,0" orientation="0,0,0" />
    </box>

    <distribute>
      <position method="uniform" min="-1,-1,0" max="1,1,0" />
      <orientation  method="gaussian"  mean="0,0,0"  std_dev="360,0,0"
/>
      <entity quantity="40" max_trials="100">
        <e-puck id="fb">
```

```
          <controller config="fdc" />
        </e-puck>
      </entity>
    </distribute>


   <!--We distribute 5 boxes uniformly in position and rotation around
Z.-->
    <distribute>
      <position method="uniform" min="-1.5,-1.5,0" max="1.5,1.5,0" />
      <orientation method="uniform" min="0,0,0" max="360,0,0" />
      <entity quantity="10" max_trials="100">
        <box id="b" size="0.1,0.1,0.4" movable="false" />
      </entity>
    </distribute>


    <!-- We distribute cylinders uniformly in position and with
constant
         rotation (rotating a cylinder around Z does not matter) -->
    <distribute>
      <position method="uniform" min="-1.5,-1.5,0" max="1.5,1.5,0" />
      <orientation method="constant" values="0,0,0" />
      <entity quantity="10" max_trials="100">
        <cylinder id="c" height="0.4" radius="0.1" movable="false" />
      </entity>
    </distribute>


  </arena>


  <!-- ****************** -->
  <!-- * Physics engines * -->
  <!-- ****************** -->
  <physics_engines>
    <dynamics2d id="dyn2d" />
  </physics_engines>
```

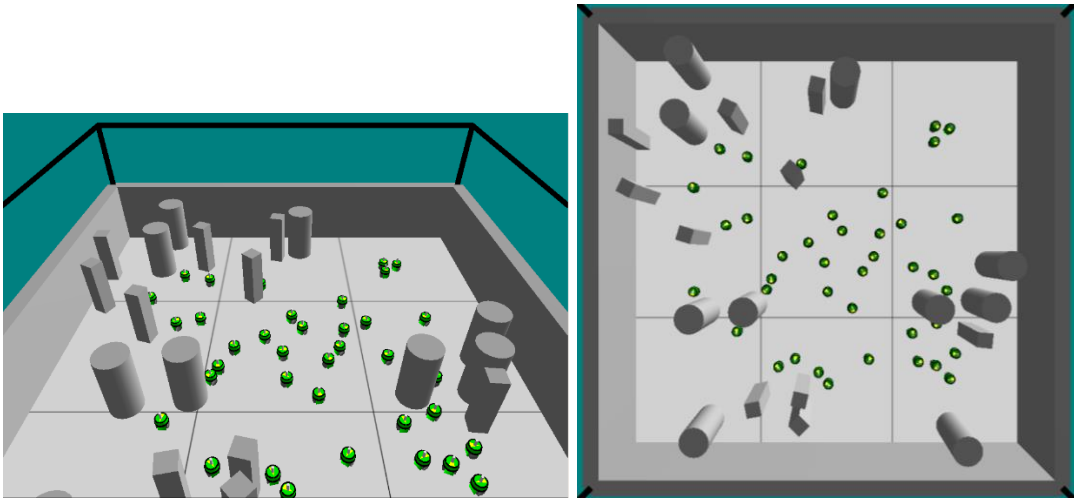
```
  <!-- ********* -->
  <!-- * Media * -->
  <!-- ********* -->
  <media />

  <!-- ****************** -->
  <!-- * Visualization * -->
  <!-- ****************** -->
  <visualization>
    <qt-opengl>
      <camera>
        <placement idx="0"
                   position="0,0,3"
                   look_at="0,0,0"
                   lens_focal_length="20" />
      </camera>
    </qt-opengl>
  </visualization>

</argos-configuration>
```

شکل ۴-۲: یک نمونه فایل تنظیمات ARGoS

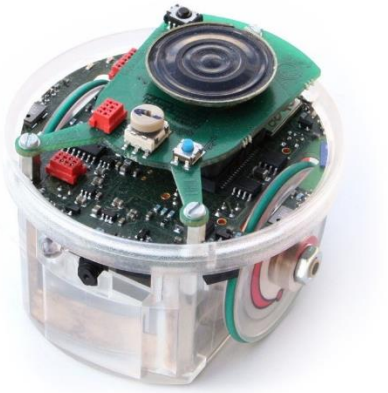
شکل ۴-۳، نتیجه شبیه‌سازی تنظیمات شکل ۴-۲ توسط سکوی ARGoS را از دو زاویه مختلف به تصویر کشیده است. در بخش camera از فایل تنظیمات، می‌توان موقعیت و زاویه دوربین را سفارشی‌سازی کرد.



**شکل ۴-۳: نمایش محیط شبیه‌سازی سکوی ARGoS حاصل از تنظیمات شکل ۴-۲ از دو زاویه‌ی مختلف.**

## ۴-۳- معرفی ربات E-puck

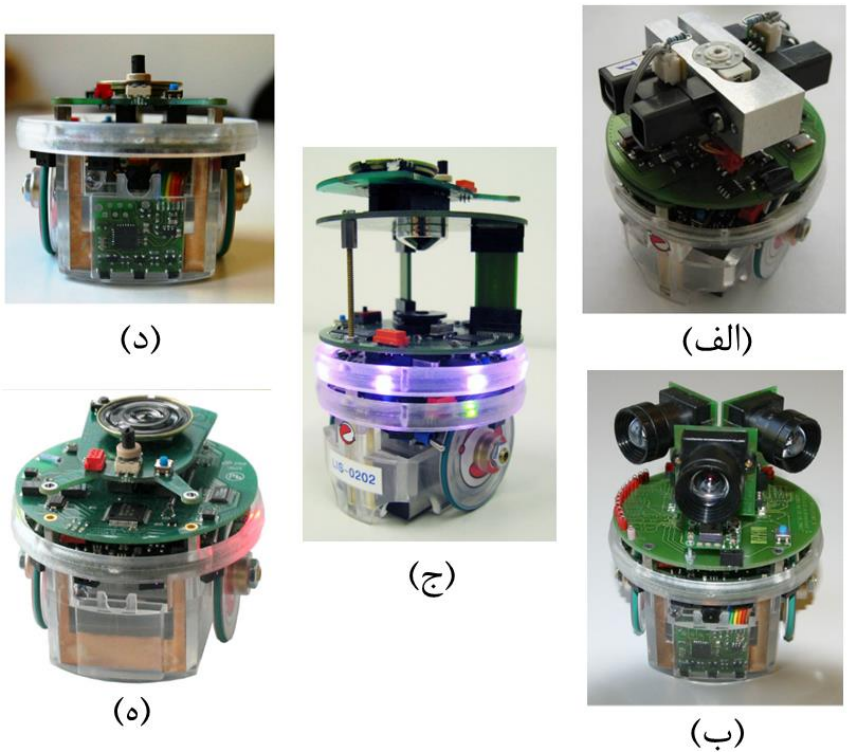
ربات E-puck رباتی است که با هدف مطالعات مهندسی در دانشگاه‌ها توسط دانشگاه EPFL خلق شد (شکل ۴-۴) [۲۹]. قبل از E-puck، ربات‌های دیگری نیز با این هدف ارائه شدند، ولی آن‌ها یا ربات‌های بهینه‌ای نبودند و یا ابزار آموزشی مناسبی نبودند. یک ربات بهینه و در عین حال یک ابزار آموزشی مناسب باید دارای شرایطی چون اندازه مناسب، کاربرپسندی، توانایی مناسب، هزینه‌ی کم و متن‌باز باشد [۲۹]. همگی ربات‌های ارائه شده قبل از E-puck حداقل یکی از این ضابطه‌ها را نداشتند؛ یا بزرگ بودند، یا دارای قابلیت‌های محدود بودند و یا گران بودند. و تنها تعداد بسیار کمی از آن‌ها متن‌باز بودند. ربات E-puck همه این ویژگی‌ها را داراست. این ویژگی‌ها منجر به ایجاد یک ربات مناسب برای رباتیک جمعی نیز شده است. اندازه‌ی کوچک و هزینه‌ی کم این ربات، امکان خرید جمعیت بالای این ربات و اجرای آزمایش‌ها در فضای کوچک را می‌دهد (شکل ۴-۵).



**شکل ۴-۴: ربات E-puck [۲۹].**

**شکل ۴-۵: استفاده از تعداد بالای ربات‌های E-puck در فضای محدود [۱۱۴].**

یکی از قابلیت‌های کاربردی ربات E-puck، امکان توسعه بردهای فیزیکی آن است؛ به این ترتیب می‌توان حسگرها، محرک‌ها و قدرت محاسباتی ربات را افزایش داد. به سه شکل می‌توان E-puck را توسعه داد. با اضافه کردن بردها به بالا، پایین و به صورت ساندویچی، می‌توان حسگرها و محرک‌های جدید اضافه کرد. شکل ۴-۶، نمونه‌هایی از توسعه E-puck به هر یک از این سه روش را به نمایش گذاشته است.




**شکل ۴-۶: توسعه ربات E-puck به شکل‌های مختلف. (الف): اضافه کردن برد فاصله‌سنج مادون‌قرمز در بالا، (ب): اضافه کردن دوربین خطی عریض در بالا، (ج): اضافه کردن دو برد کامل برای ارتباطات بصری به صورت ساندویچی، (د): اضافه کردن برد تشخیص رنگ زمین در پایین، (ه): اضافه کردن برد رادویی Zigbee به صورت ساندویچی [۲۹].**

با توجه به قابلیت توسعه آسان در ربات E-puck می‌توان آن را متناسب با وظیفه موردنظر، سفارشی‌سازی کرد. برای مثال برای انجام وظیفه کاوش هدف به حسگرها و محرک‌های زیر نیاز است:

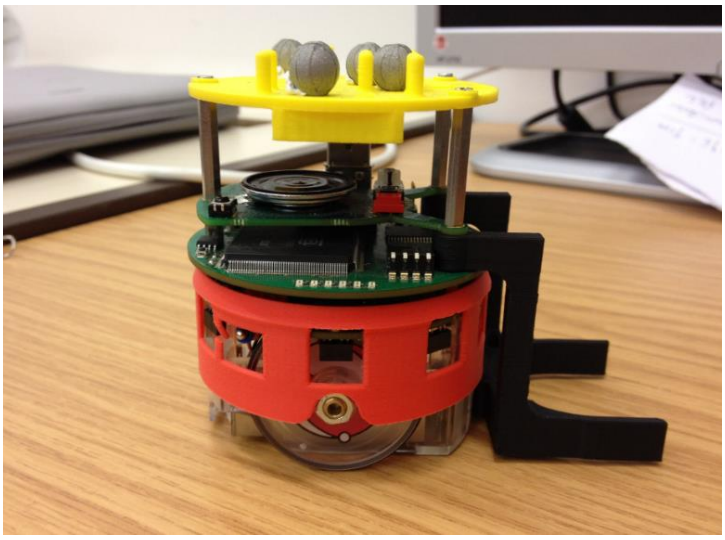
- محرک چرخ‌ها[۱]: برای حرکت و چرخش ربات در محیط با حداکثر سرعت ۱۳ سانتی‌متر بر ثانیه.
- محرک لامپ‌های ال‌ای‌دی رنگی[۲]: برای نمایش وضعیت جاری (جستجو، استراحت و ...) با رنگ‌های متناسب
- محرک دستگیره[۳]: برای برداشتن و حمل اهداف
- حسگر مجاورت‌سنج[۴]: برای تشخیص موانع و سایر ربات‌ها که دارای برد شش سانتی‌متر است.
- حسگر زمین[۵]: برای تشخیص محدوده خانه با بررسی رنگ زمین.

---

۱ Wheels Actuator
۲ RGB LEDs Actuator
۳ Gripper Actuator
۴ Proximity Sensor
۵ Ground Sensor

- حسگر نور[1]: برای تشخیص جهت خانه
- حسگر دوربین ۳۶۰ درجه[2]: برای دیدن اهداف تا فاصله ۳۵ سانتی‌متر.

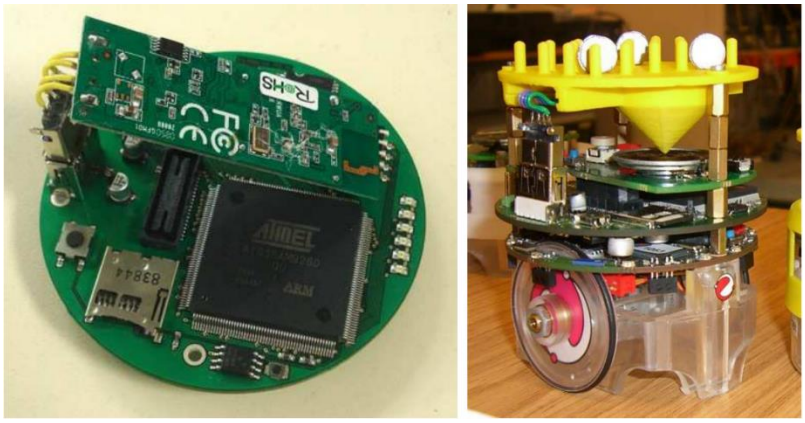
خوشبختانه، غیر از محرک دستگیره، سایر موارد قابل اضافه کردن به E-puck هستند. برای اضافه کردن محرک دستگیره، باید از ترفندهای خاص منظوره استفاده کرد. برای مثال، مرجع [۱۱۵] دستگیره‌هایی را طراحی و به صورت سه‌بعدی پرینت کرده است. طبیعتا این دستگیره قابلیت دریافت دستور (باز و بسته شدن) ندارند و تنها به حمل غذا بر روی زمین توسط ربات کمک می‌کنند (شکل ۴-۷).



**شکل ۴-۷: اضافه کردن دستگیره به E-puck با چاپ سه‌بعدی [۱۱۵].**

اضافه کردن حسگرها و محرک‌های جدید به E-puck به دلیل محدودیت رم (8KB) و قدرت پردازشی آن (16 MIPS) نمی‌تواند از تعداد مشخصی فراتر رود. از طرفی پردازش تصاویر ورودی توسط دوربین‌ها نیز به رم و قدرت پردازشی بالایی نیاز دارد. برای غلبه بر این محدودیت‌ها، مرجع [۱۱۶] یک برد توسعه مبتنی بر سیستم‌عامل لینوکس ارائه کرده است که از میکروپردازنده ARM9 بهره می‌برد و در کنار پردازنده اصلی ربات اجرا می‌شود (شکل ۴-۸). در این رساله نیز از این برد برای توسعه توان پردازشی ربات بهره برده شده است.

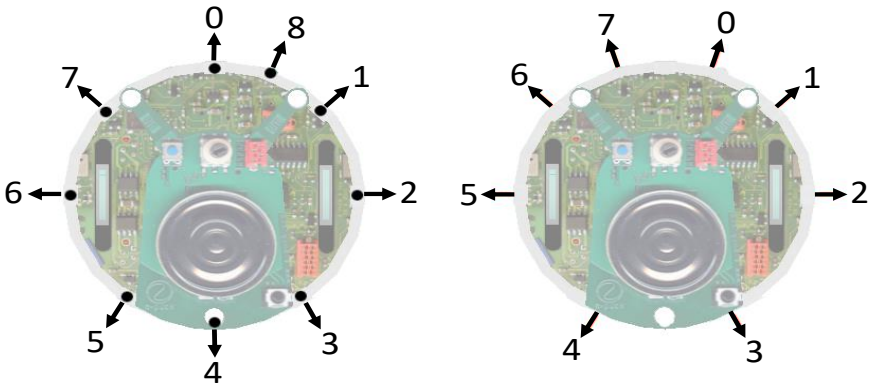
[1] Light Sensor
[2] OmniDirectional Camera Sensor



**شکل ۴-۸: (چپ): برد توسعه مبتنی بر لینوکس (راست): ربات E-puck مجهز به برد توسعه مربوطه که در بالای برد اصلی ربات و در زیر برد بلندگو قرار گرفته است [۱۱۶].**

مکان قرارگیری حسگرها و محرک‌های پرکاربرد ربات E-puck در شکل ۴-۹ نمایش داده شده‌اند. E-puck دارای هشت حسگر مجاورت‌سنج است که مکان دقیق آن‌ها در شکل ۴-۱۰ (الف) به نمایش گذاشته شده است. هر یک از این حسگرها مقدار مشخصی نور مادون‌قرمز را تابش کرده و از میزان انعکاس نور، وجود یا عدم وجود اجسام و فاصله تقریبی آن‌ها را تشخیص می‌دهد. E-puck دارای سه حسگر زمین نیز هست که عملکرد مشابهی دارند. به دلیل حساسیت حسگرها در دریافت انعکاس نور، آن‌ها می‌توانند به عنوان حسگر نور هم مورد استفاده قرار بگیرند. E-puck دارای نه ال‌ای‌دی رنگی است که همگی با هم روشن و خاموش می‌شوند. مکان دقیق این محرک‌ها در شکل ۴-۱۰ (ب) نشان داده شده است.


**شکل ۴-۹: معرفی حسگرها و محرک‌های کاربردی ربات E-puck.**




ب الف

**شکل ۴-۱۰: (الف): مکان حسگرهای مجاورت‌سنج (یا حسگرهای نور) و (ب): مکان محرک ال‌ای‌دی‌های رنگی در ربات E-puck از نمای بالا.**

## ۴-۴- پیاده‌سازی ptSCT بر روی سکوی ARGoS

ساختار پیاده‌سازی شده مبتنی بر ساختار ارائه شده در [۴۹] می‌باشد. این ساختار از سه لایه تشکیل شده که برروی سخت‌افزار قرار می‌گیرند. در بالاترین لایه، سوپروایزر قرار گرفته است. سپس، اجرا کننده مولدها[۱] و در پایینی‌ترین لایه، رویه‌های عملیاتی[۲] قرار گرفته است (شکل ۴-۱۱). وظیفه لایه رویه‌های عملیاتی، تبدیل سیگنال‌ها به رخدادها و برعکس است. این لایه بسته به نوع ربات، متغیر است؛ زیرا هر ربات مجموعه حسگرها و محرک‌های متفاوتی دارد و طبیعتا دارای سیگنال‌های منحصربفرد است. به صورت کلی، این لایه، لایه سخت‌افزار را از سایر لایه‌های نرم‌افزاری پنهان می‌کند. لایه اجرا کننده مولدها، یک یا چند مولد را دریافت کرده و با توجه به رخدادهای ورودی، یک یا چند گام از مولدها را اجرا می‌کند. سوپروایزر در بالاترین لایه، وظیفه مدیریت اجرا کننده‌ها را برعهده دارد. وظایف سطح بالایی چون موارد زیر توسط این لایه صورت می‌گیرد. جزئیات پیاده‌سازی ptSCT در ادامه تشریح شده است.

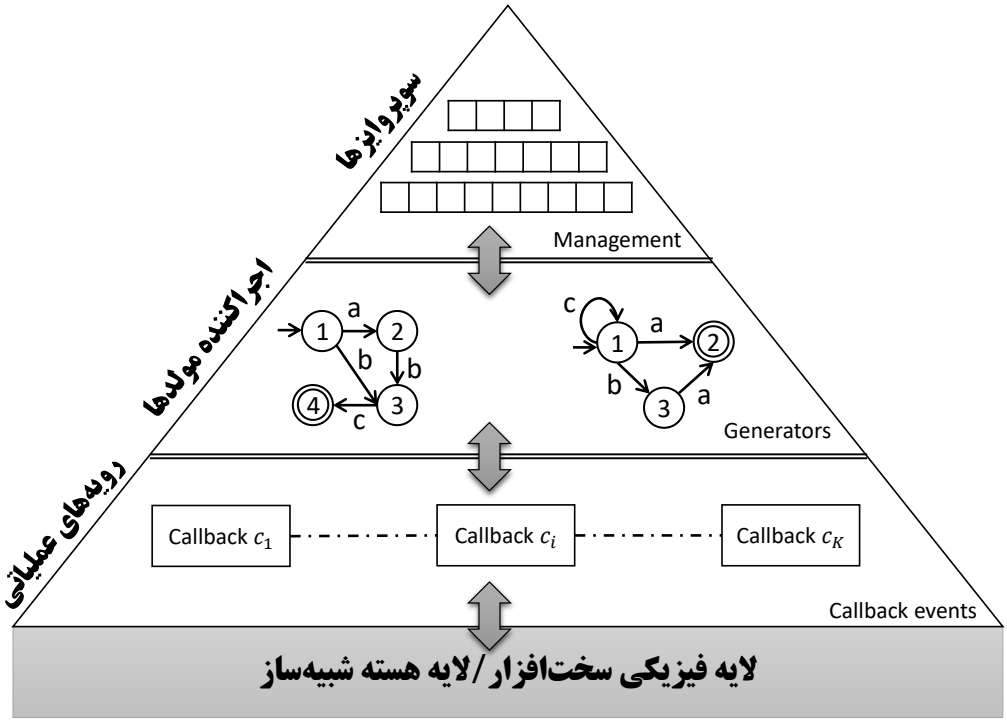
- نرمال‌سازی احتمالات در سوپروایزرهای ماژولار محلی
- اعمال زمان به رخدادها و تعیین فعال بودن/نبودن یک رخداد
- تعیین مولدهای مرتبط به یکدیگر با توجه به اشتراک رخدادهایشان

---

[۱] Generators Player

[۲] Operational Procedures

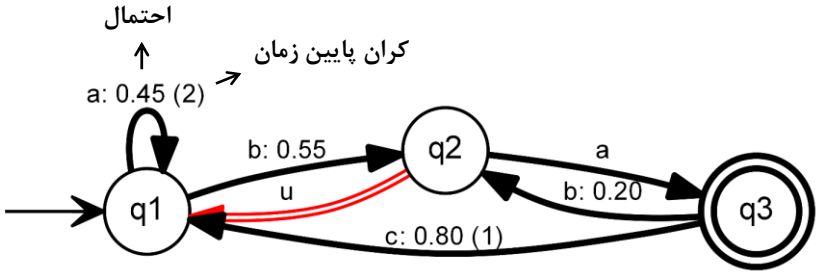
- ایجاد ارتباط بین هر رخداد و تکه‌کدی که باید در زمان اجرای آن رخداد، فراخوانده شود (Callback function)
- تمایز بین رخدادهای قابل کنترل و غیرقابل کنترل
- مدیریت رخداد تیک ساعت سراسری سیستم



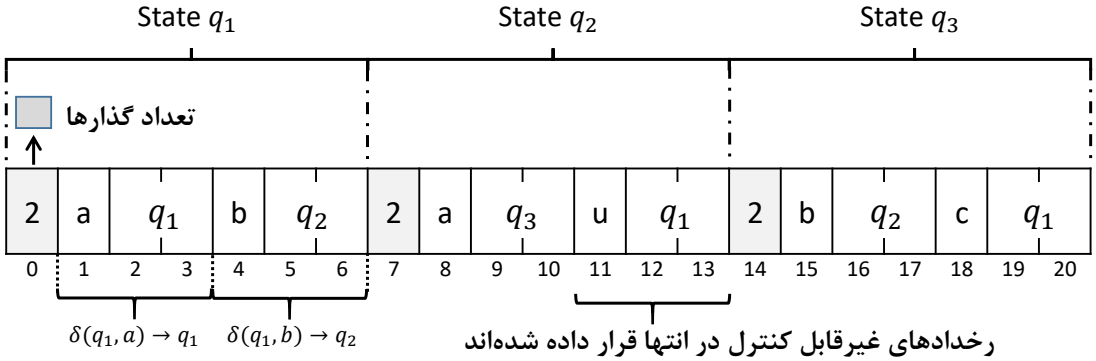

**شکل ۴-۱۱: سه لایه پیاده‌سازی شده بر روی لایه سخت‌افزار در سکوی ARGoS.**

### ۴-۴-۱- ساختار پیشنهادی برای توصیف ptSCT در ARGoS

در ابتدا به توصیف ساختمان داده مورد استفاده برای نگهداری مولدها و جزئیات مربوط به آن‌ها پرداخته شده است. برای سادگی بیشتر، مولد احتمالاتی و زمان‌دار نشان داده شده در شکل ۴-۱۲را درنظر بگیرید. ساختار مورد استفاده برای توصیف تابع انتقال ($\delta$) این مولد در شکل ۴-۱۳ نمایش داده شده است. هر بلوک در این ساختار با یک بایت آغاز می‌شود که بیانگر تعداد گذارهای خارج شده از حالت مربوطه است. سپس به ازای هر گذار، سه بایت وجود خواهد داشت. بایت اول، رخداد مربوط به گذار را نگه داشته و دو بایت بعدی، حالت مقصد را نگه می‌دارند. رخدادهای قابل کنترل در ابتدای بلوک قرار داده می‌شوند. ساختار پیشنهادی می‌تواند مولدهایی را توصیف کند که حداکثر دارای ۲۵۵ گذار خروجی از هر حالت باشند؛ همچنین هر مولد می‌تواند حداکثر ۶۵۵۳۵ حالت داشته باشد. با توجه به شکل سنتز در روش پیشنهادی که ماژولار محلی است؛ بنظر می‌رسد این محدودیت‌ها مشکل‌ساز نخواهد بود. هرچند ساختار پیشنهادی به سادگی قابل توسعه برای مولدهای بزرگتر می‌باشد.

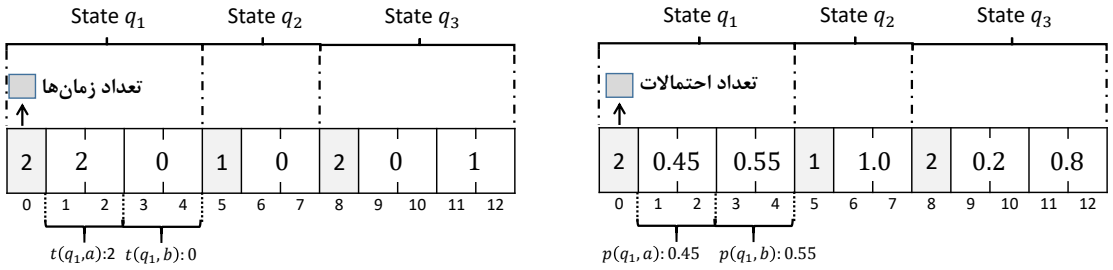



**شکل ۴-۱۲: یک مولد احتمالاتی و زمان‌دار نمونه.**




**شکل ۴-۱۳: شکل توصیف تابع انتقال مربوط به مولد شکل ۴-۱۲ در ARGoS.**

ساختار پیشنهادی برای توصیف تابع احتمالات و زمان برای مولد شکل ۴-۱۲ در شکل ۴-۱۴ به نمایش گذاشته شده است. برای توصیف تابع احتمالات و زمان از یک ساختمان داده مشابه بهره گرفته شده است؛ در این ساختار، هر بلوک با یک بایت آغاز می‌شود که بیانگر تعداد گذارهای خروجی با رخدادهای قابل کنترل است. پس از آن، به ازای هر رخداد، دو بایت قرار می‌گیرد که «احتمال»/«کران پایین زمان» آن رخداد را نگه می‌دارد. در این ساختار، شماره رخدادها حذف شده است تا در مصرف فضا صرفه‌جویی شود؛ از آنجا که در ساختار مربوط به تابع انتقال، رخدادهای قابل کنترل در ابتدای بلوک قرار داده شده‌اند؛ بنابراین، اندیس رخدادهای قابل کنترل بین این سه ساختار می‌تواند مشترک درنظرگرفته شود. در نتیجه، شماره رخدادها در دو ساختار مربوط به احتمالات و زمان، از ساختار مربوط به تابع انتقال قابل استنتاج است.




الف ب

**شکل ۴-۱۴: شکل توصیف تابع احتمالات (الف) و تابع زمان (ب) مربوط به مولد شکل ۴-۱۲ در ARGoS.**

علاوه‌بر سه ساختار بالا، یک آرایه به طول تعداد رخدادها ($e$) نیز تعریف می‌شود که برای هر رخداد، قابل کنترل بودن یا نبودن آن را تعیین می‌کند. یک ماتریس $e \times N$ نیز برای تعیین تعلق یا عدم تعلق هر رخداد به هر سوپروایزر درنظر گرفته شده است. $N$ بیانگر تعداد سوپروایزرها است. یک آرایه به طول $N$ نیز برای نگهداری حالت جاری هر سوپروایزر (مولد) مورد استفاده قرار می‌گیرد که هر خانه آن دو بایت فضا اشغال می‌کند.

اگر تعداد حالت‌های همه‌ی سوپروایزرهای ماژولار محلی را $s$ و تعداد مجموع گذارها را $t$ در نظر بگیریم، مجموع فضای اشغالی توسط ساختار پیشنهادی، رابطه خطی با مجموع تعداد حالت‌ها و گذارها خواهد داشت. مقدار دقیق فضای اشغالی در لایه‌های سوپروایزرها و اجراکننده مولدها، توسط فرمول (۲۱) ارائه شده است. $t_c$ برابر است با تعداد رخدادهای قابل کنترل.

$$\begin{aligned} \text{MemCons} &= (s + 3t) + 2 \times (s + 2t_c) + e + (e + 2) \times N \\ &\in O(s + t) \end{aligned} \qquad (۲۱)$$

برای محاسبه مجموع فضای اشغال شده توسط ساختار پیشنهادی باید فضای اشغالی ارائه شده توسط فرمول (۲۱) را با فضای اشغالی توسط لایه رویه‌های عملیاتی جمع کرد. واحد فضای اشغالی ارائه شده برابر با بایت[۱] می‌باشد.

### ۴-۴-۲- اجرا کننده مولد احتمالاتی و زمان‌دار

در ساختار پیشنهادی، اجرا کننده مولد به کمک لایه سوپروایزرها، قادر به اجرای چندین مولد به صورت همزمان است که هر مولد دارای مولفه‌های احتمالات و زمان نیز می‌باشد. از آنجا که سوپروایزر یکپارچه می‌تواند به شکل سوپروایزر ماژولار محلی نیز توصیف شود، بنابراین اجرا کننده‌ی مولد پیشنهادی قادر به اجرای مولد مربوط به سوپروایزر یکپارچه نیز می‌باشد.

[۱] Byte

اجرا کننده مولد باید کران پایین زمان هر رخداد و همچنین احتمال آن را در نظر بگیرد. همچنین باید ماژول‌هایی که دارای رخدادهای مشترک هستند را در محاسبات خاص مثل نرمال‌سازی احتمالات شرکت دهد. رخداد تیک ساعت سراسری نیز در این لایه مدیریت می‌شود. این موارد از طریق لایه بالایی (لایه سوپروایزرها) برای این لایه ارسال می‌شوند. با توجه به توضیحات داده شده، الگوریتم اجرا کننده مولد احتمالاتی و زمان‌دار در شکل ۴-۱۵ ارائه شده است.

1: **Let** $N$ be the number of supervisors $S_i, i \in \{1,2,\dots,N\}$

2: **procedure** GeneratorPlayer

3: **for all** $i \in \{1,2,\dots,N\}$ **do**

4: set current state $c_i$ of supervisor $S_i$ to initial state $q_{0_j}$

5: **while** $true$ **do**

6: calculate the set of occurred uncontrollable events $\Psi$

7: **for all** $e_u \in \Psi$ **do**

8: **for all** $i \in \{1,2,\dots,N\}$ **do**

9: $c_i \leftarrow \delta_i(c_i, e_u)$

10: update the local counter of the current states of all supervisors

11: calculate the set of enabled controllable events $\Phi$ which satisfy lower bound time constraint

12: **if** $\Phi \neq \emptyset$ **then**

13: normalize probabilities of $\Phi$

14: select one controllable event $e_c \in \Phi$ using roulette wheel algorithm

15: **for all** $i \in \{1,2,\dots,N\}$ **do**

16: $c_i \leftarrow \delta_i(c_i, e_c)$

17: execute callback function of $e_c$

**شکل ۴-۱۵: الگوریتم پیشنهادی اجرا کننده‌ی مولد احتمالاتی و زمان‌دار.**

در خطوط سوم و چهارم از الگوریتم پیشنهادی، سوپروایزرهای ماژولار محلی، مقداردهی اولیه می‌شوند. سپس دو قطعه کد در یک حلقه بی‌نهایت اجرا می‌شوند. قطعه کد اول، همه‌ی رخدادهای غیرقابل کنترل را بررسی کرده و مواردی که اتفاق افتاده‌اند را در یک مجموعه نگه می‌دارد. سپس حالت فعلی همه سوپروایزرها را با توجه به رخدادهای جمع‌آوری شده، بروزرسانی می‌کند. قطعه کد دوم، پس از بروزرسانی شماره‌ی داخلی حالت جاری همه سوپروایزرها، رخدادهای قابل کنترل فعالی که کران پایین

زمان‌شان معتبر باشد را در یک مجموعه قرار می‌دهد (خطوط دهم و یازدهم)؛ آنگاه احتمال رخدادها را با توجه به سوپروایزرهای متناظرشان، نرمال‌سازی می‌کند (خط سیزدهم). سپس، با استفاده از الگوریتم چرخ رولت[۱] یکی از رخدادها به صورت تصادفی و با درنظر گرفتن احتمالات نرمال شده انتخاب می‌شود (خط چهاردهم). پس از آن، حالت فعلی همه سوپروایزرها مجددا بروزرسانی خواهد شد. در پایان، تابع فراخوان[۲] مربوط به رخداد انتخاب شده، اجرا می‌گردد (خط هفدهم). لایه اجرا کننده‌ی مولد تنها از امضای تابع فراخوان مطلع است و از پیاده‌سازی آن اطلاعی ندارد. پیاده‌سازی توابع فراخوان در لایه رویه‌های عملیاتی ارائه می‌گردد.

### ۴-۴-۳- رویه‌های عملیاتی

همان‌طور که پیش‌تر اشاره شد، نزدیک‌ترین لایه به سخت‌افزار در ساختار پیشنهادی، لایه رویه‌های عملیاتی است. این لایه در ابتدا در محیط‌های تولید ارائه شد و وظیفه‌اش ترجمه رخدادها به/از سیگنال‌های ورودی/خروجی کنترل کننده‌های منطقی قابل برنامه‌ریزی[۳] بود [۸۹]. در ساختار پیشنهادی، وظیفه‌ی این لایه، سطح بالاتر است. این لایه به برنامه‌نویس اجازه می‌دهد تا توابع فراخوان متناظر با هر رخداد را به شکل دلخواه پیاده‌سازی کند. این توابع در نهایت توسط لایه‌های بالاتر و در هنگام اجرا شدن رخداد مربوطه فراخوانی خواهند شد. برای مثال، فرض کنید رخدادی به نام $turnRight$ در طراحی درنظرگرفته شده که وظیفه آن چرخش ربات به سمت راست است. عملکرد دقیق این رخداد توسط تابع فراخوان آن مشخص می‌شود. یک نمونه پیاده‌سازی این تابع در شکل ۴-۱۶ ارائه شده است. در این پیاده‌سازی، ابتدا یک اشاره‌گر به محرک چرخ‌های ربات E-puck تعریف شده است (خط سوم)؛ سپس سرعت موتور سمت چپ ربات برابر با یک مقدار مثبت و سرعت موتور سمت راست برابر با صفر تنظیم شده است (خط هفتم)؛ سرعت حرکت بیشتر موتور سمت چپ نسبت به راست، منجر به چرخش ربات به سمت راست خواهد شد. اگر سرعت حرکت موتور مخالف برابر با یک مقدار منفی (بجای صفر) قرار داده شود؛ شدت چرخش بیشتر خواهد شد.

```
1: #include<argos3/plugins/robots/e-
puck/control_interface/ci_epuck_wheels_actuator.h>
2:
3: CCI_EPuckWheelsActuator* m_pcWheels =
4:             GetActuator<CCI_EPuckWheelsActuator> ("epuck_wheels");
5:
6: void callback_turnRight(){
```

[۱] Roulette wheel
[۲] Callback function
[۳] Programmable Logic Controller

```
7:     m_pcWheels->SetLinearVelocity(2.5f, 0);
8: }
```

**شکل ۴-۱۶: یک نمونه پیاده‌سازی تابع فراخوان برای چرخش ربات E-puck به سمت راست.**

عملکرد توابع فراخوان برای رخدادهای غیرقابل کنترل کمی متفاوت است؛ وظیفه این توابع برای رخدادهای غیرقابل کنترل، بررسی وقوع رخداد مربوطه است. برای مثال، اگر قصد داریم ربات به مدت دو ثانیه به سمت راست بچرخد، می‌توان رخداد غیرقابل کنترل moveEnded را به این صورت پیاده‌سازی کرد: آیا زمان چرخش از دو ثانیه تجاوز کرده است؟ توابع فراخوان رخدادهای غیرقابل کنترل معمولا یک خروجی از نوع Boolean دارند که وقوع یا عدم وقوع رخدادها را اطلاع‌رسانی می‌کنند. در صورتی که توابع فراخوان رخدادهای قابل کنترل، مقدار خروجی ندارند و تنها عمل یا مجموعه‌ای از اعمال را اجرا می‌کنند.

در ساختار پیشنهادی، تنها بخشی که برنامه‌نویس (طراح) باید آن را پیاده‌سازی کند، همین بخش رویه‌های عملیاتی است. این لایه، انعطاف‌پذیری در طراحی را به شدت افزایش می‌دهد، زیرا برنامه‌نویس می‌تواند توابع فراخوان را به شکل دلخواه خود پیاده‌سازی کند. هرچند این بخش می‌تواند یک نکته منفی نیز درنظر گرفته شود، زیرا پیاده‌سازی صورت گرفته توسط طراح مصون از باگ‌های نرم‌افزاری نیست.

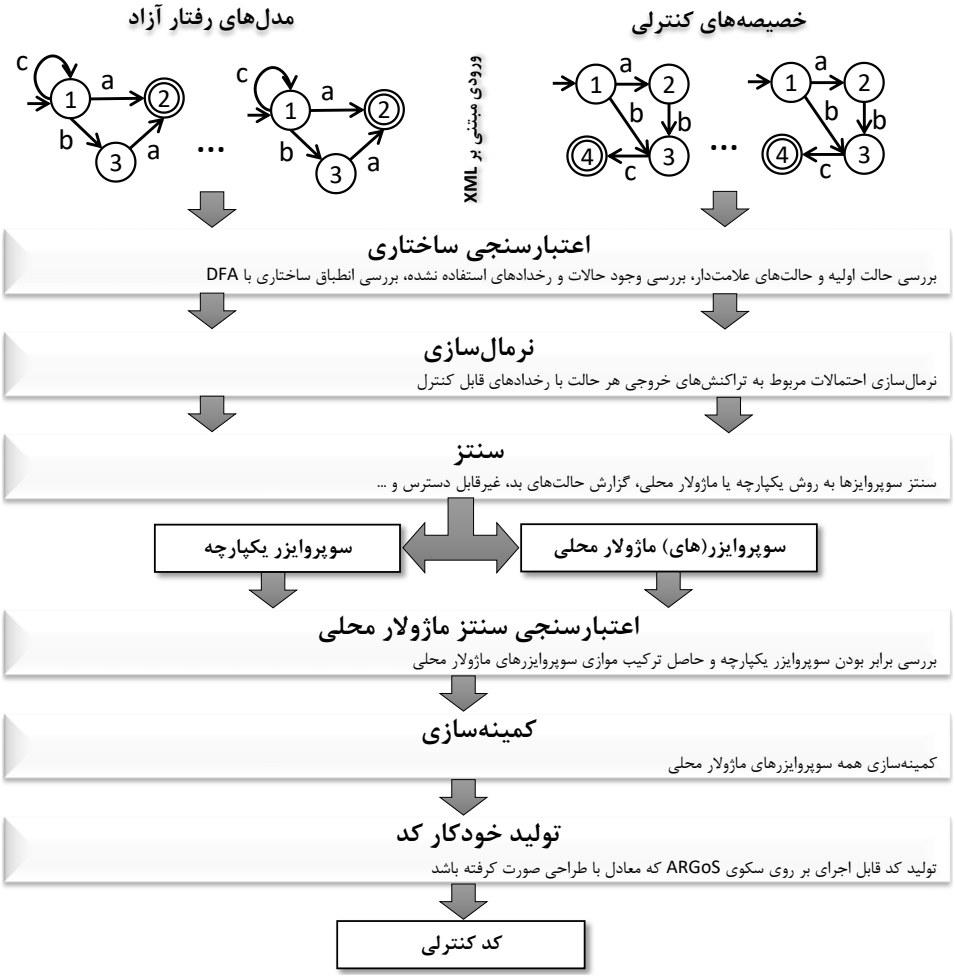
## ۴-۵- پیاده‌سازی نرم‌افزار جامع سنتز و تولید خودکار کد

یکی از کمبودهای حوزه رباتیک جمعی، وجود نرم‌افزاری است که بتواند به صورت جامع دربرگیرنده همه مراحل طراحی یک کنترل کننده از مدل‌سازی تا تولید خودکار کد باشد. در این بخش به تشریح نرم‌افزار پیشنهادی در این رساله پرداخته شده است.

ابزار پیشنهادی توسط زبان برنامه‌نویسی Python پیاده‌سازی شده است و به صورت خلاصه، دارای قابلیت‌های زیر است که هریک در ادامه تشریح خواهند شد. جزئیات و ترتیب مراحل انجام شده توسط ابزار پیشنهادی در شکل ۴-۱۷ به تصویر کشیده شده است.

- دریافت مدل‌های رفتار آزاد و ضابطه‌های کنترلی در قالب XML
- اعتبارسنجی ساختاری مولدها و یادآوری خطاها به طراح
- سنتز مولدها و محاسبه خودکار سوپروایزر به دو روش یکپارچه و ماژولار محلی
- کمینه‌سازی مولدها و حذف حالت‌های غیرقابل دسترس و حالت‌های بد
- رسم گرافیکی مولدها با هدف تسهیل در طراحی و اشکال‌زدایی

- تولید خودکار کدهای کنترلی برای اجرا در محیط ARGoS و یا استفاده بر روی ربات‌های واقعی




**شکل ۴-۱۷: فلوچارت مراحل انجام شده توسط ابزار جامع پیشنهادی.**

ابزار پیشنهادی برای دریافت مدل‌های رفتار آزاد و ضابطه‌های کنترلی از ساختار نشان داده شده در شکل ۴-۱۸ بهره می‌برد. این ساختار متناظر با مدل ارائه شده در شکل ۴-۱۲ است. در این ساختار، چهار نوع المان اصلی وجود دارد:

۱. **المان automata**: این المان، المان ریشه در ساختار مبتنی بر XML پیشنهادی است که خود دارای دو خصیصه است. خصیصه type که نوع مدل را تعیین می‌کند (مدل رفتار آزاد یا ضابطه‌ی کنترلی)؛ و خصیصه‌ی name که نام مدل را مشخص می‌سازد.
۲. **المان states**: این المان دربرگیرنده حالت‌های موجود در ماشین حالت می‌باشد. هر حالت توسط یک المان state مشخص می‌شود که دارای یک خصیصه name است؛ همچنین برای

تعیین حالت ابتدایی می‌توان از خصیصه i بهره برد و مقدار آن را برابر با یک (true) قرار داد. حالات علامت‌دار توسط خصیصه m تعیین می‌شوند.

۳. **المان events**: مجموعه‌ی رخدادها در این المان قرار داده می‌شوند. هر رخداد توسط یک المان event تعریف می‌شود. رخدادها دارای دو خصیصه name و con هستند. خصیصه name، نام رخداد و خصیصه con، قابل کنترل بودن/نبودن آن را مشخص می‌سازد.

۴. **المان transitions**: این المان شامل مجموعه‌ی گذارهای موجود در ماشین حالت است. هر گذار توسط یک المان transition مشخص می‌شود. هر گذار دارای سه خصیصه اجباری و دو خصیصه اختیاری است. سه خصیصه اجباری عبارتند از خصیصه‌های src، dst و event که به ترتیب حالات مبدا و مقصد و رخداد مربوط به گذار را تعیین می‌کنند. دو خصیصه اختیاری برای تعیین احتمال (prob) و کران پایین زمان (time) گذار مورد استفاده قرار می‌گیرند. بازه مورد استفاده برای احتمال مربوط به هر گذار، محدودیتی ندارد و در زمان سنتز، نرمال‌سازی خواهد شد. واحد مورد استفاده برای زمان، تعداد تیک‌های ساعت سراسری سیستم است؛ شروع شمارش تیک‌ها از زمان ورود به حالت جاری درنظر گرفته می‌شود. در صورت عدم تعیین زمان برای یک گذار، مقدار صفر برای آن درنظر گرفته می‌شود. در صورت عدم تعیین احتمال برای یک گذار، مقدار یک برای آن درنظر گرفته می‌شود.

شِمای DTD[۱] مورد استفاده برای ساختار پیشنهادی در شکل ۴-۱۹ ارائه گشته است. این شِما، روابط المان‌های معرفی شده با یکدیگر را تعریف می‌کند. المان automata به عنوان المان ریشه تعریف شده که می‌تواند دارای دو خصیصه name و type باشد. از طرفی، المان automata می‌توان دارای زیرالمان‌های states، events و transitions باشد. هر المان states دربرگیرنده تعدادی (یک یا بیشتر) زیرالمان state است که هریک دارای سه خصیصه name، i و m بوده و دو مورد آخر به صورت اختیاری تعریف گشته‌اند. المان‌های events و transitions نیز به ترتیب دربرگیرنده یک یا بیشتر زیرالمان event و transition هستند. هر المان event دارای خصیصه اجباری name و خصیصه اختیاری con می‌باشد. المان‌های transition نیز باید شامل سه خصیصه event، src و dst بوده و می‌توانند شامل خصیصه‌های prob و time نیز باشند.

[۱] Document Type Definition

```xml
<automata name="G1" type="plant">
    <states>
        <state name="q1" i="1"/>
        <state name="q2" />
        <state name="q3" m="1"/>
    </states>

    <events>
        <event con="1" name="a"/>
        <event con="1" name="b"/>
        <event con="1" name="c"/>
        <event con="0" name="u"/>
    </events>

    <transitions>
        <transition event="a" src="q1" dst="q1" prob="0.45" time="2"/>
        <transition event="b" src="q1" dst="q2" prob="0.55"/>

        <transition event="a" src="q2" dst="q3"/>
        <transition event="u" src="q2" dst="q1"/>

        <transition event="b" src="q3" dst="q2" prob="0.2"/>
        <transition event="c" src="q3" dst="q1" prob="0.8" time="1"/>
    </transitions>
</automata>
```

**شکل ۴-۱۸: ساختار مبتنی بر XML مورد استفاده در ابزار پیشنهادی برای دریافت مدل‌های رفتار آزاد و ضابطه‌های کنترلی. این ساختار متناظر با مدل ارائه شده در شکل ۴-۱۲ می‌باشد.**

```xml
<!DOCTYPE automata [
    <!ELEMENT automata (states, events, transitions)>
        <!ATTLIST automata name CDATA #REQUIRED>
        <!ATTLIST automata type CDATA #REQUIRED>
    <!ELEMENT states (state+)>
        <!ATTLIST state name CDATA #REQUIRED>
        <!ATTLIST state i CDATA #IMPLIED>
        <!ATTLIST state m CDATA #IMPLIED>
    <!ELEMENT events (event+)>
        <!ATTLIST event name CDATA #REQUIRED>
        <!ATTLIST event con CDATA #REQUIRED>
    <!ELEMENT transitions (transition+)>
        <!ATTLIST transition event CDATA #REQUIRED>
        <!ATTLIST transition src CDATA #REQUIRED>
        <!ATTLIST transition dst CDATA #REQUIRED>
        <!ATTLIST transition prob CDATA #IMPLIED>
        <!ATTLIST transition time CDATA #IMPLIED>
]>
```

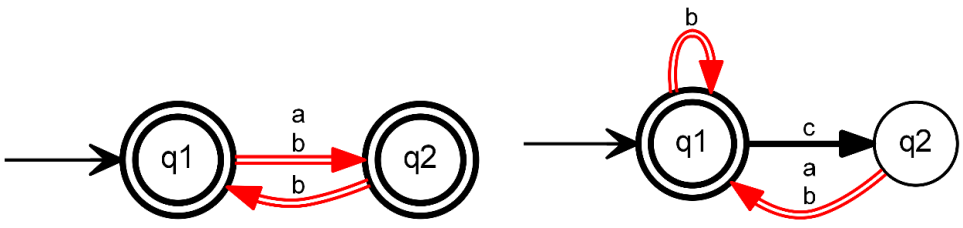
**شکل ۴-۱۹: شمای DTD مورد استفاده برای ساختار XML پیشنهادی.**

در صورتی که اشکالات غیرعمدی در طراحی وجود داشته باشد، ابزار پیشنهادی آن‌ها را تشخیص داده و به طراح گوشزد می‌کند. مواردی از قبیل عدم تعیین حالت آغازین، عدم وجود حداقل یک حالت علامت‌دار، استفاده از رخداد یا حالتی در بخش گذارها که در بخش مربوط به خودش تعریف نشده و یا تعریف حالت و رخدادی که در بخش گذارها مورد استفاده قرار نگرفته شده باشد. طراحی صورت گرفته

حتما باید مطابق با ساختار یک DFA[1] باشد؛ یعنی هر حالت از ماشین حالت حداکثر دارای یک گذار خروجی متناظر با هر رخداد باشد. این مورد نیز توسط ابزار پیشنهادی بررسی می‌شود.

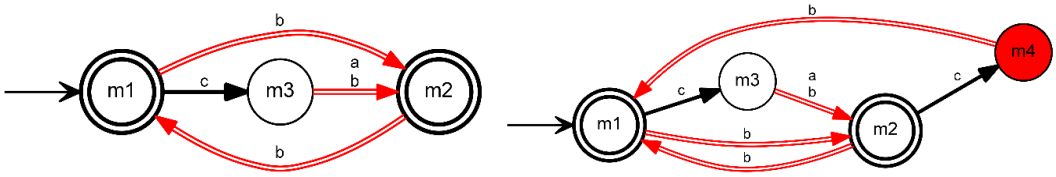
پس از اعتبارسنجی طراحی‌های اولیه، همه مراحل سنتز که در فصل سوم توضیح داده شد، شامل اعمال عملگر ترکیب موازی، کمینه‌سازی، حذف حالت‌های غیرقابل دسترس و حالت‌های بد به صورت خودکار انجام می‌گیرد. در صورت تغییر سوپروایزر پس از اعمال هر یک از مراحل نامبرده شده، نسخه‌ی قبل و بعد از تغییر به اطلاع طراح رسانده می‌شود. حالت‌های غیرقابل دسترس و حالت‌های بد حذف شده نیز به طراح اطلاع‌رسانی می‌شوند. در صورتی که طراح، روش سنتز ماژولار محلی را برگزیده باشد، طراحی مبتنی بر XML مربوط به هر یک از سوپروایزرها نیز در اختیار طراح قرار می‌گیرد. علاوه‌بر این، مولد متناظر با هر فایل مبتنی بر XML به منظور تحلیل و اشکال‌یابی ساده‌تر به صورت گرافیکی رسم می‌شود. حتی کیفیت و فرمت تصویر خروجی نیز توسط طراح قابل تنظیم است. شکل ۴-۲۰، یک نمونه مدل رفتار آزاد و ضابطه‌ی کنترلی را نشان می‌دهد که پس از اعمال عملگر ترکیب موازی، دارای یک حالت بد خواهد بود. ابزار پیشنهادی، حالت مربوطه را با رنگ قرمز نمایش داده است تا طراح از وجود آن مطلع شود. سپس حالت بد را حذف کرده و سوپروایزر نهایی را قبل و بعد از کمینه‌سازی در اختیار طراح قرار می‌دهد.

پس از محاسبه خودکار سوپروایزر(های) نهایی، کد موردنیاز برای اجرای این کنترل کننده به ربات‌ها در سکوی ARGoS در قالب یک فایل به فرمت cpp. تولید می‌شود. این فایل بدون هیچ تغییری، قابل استفاده در سکوی ARGoS خواهد بود. تنها موردی که قبل از استفاده از این فایل به آن نیاز داریم، فراهم آوردن توابع فراخوان متناظر با هر رخداد است. از آنجا که توابع فراخوان، توابع کوچک و معمولا پراستفاده هستند، پیاده‌سازی آن‌ها زمان‌بر نیست.

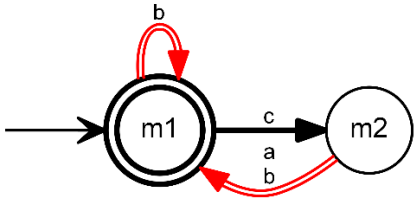



ب) $E$ الف) $G$

[1] Deterministic finite automaton




ب) $K = G \parallel E$ د) $S = SupC(K, G)$




ه) $S_m$

**شکل ۴-۲۰: مثالی از عملکرد ابزار پیشنهادی در نمایش حالت بد به طراح. الف) یک نمونه مدل رفتار آزاد $G$، ب) یک نمونه ضابطه‌ی کنترلی $E$، ج) سوپروایزر محاسبه شده $K$ توسط ابزار پیشنهادی که دارای حالت بد است، د) سوپروایزر نهایی پس از حذف حالت بد و ه) سوپروایزر کمینه شده.**

## ۴-۶- بررسی صحت عملکرد نرم‌افزار

برای بررسی صحت عملکرد نرم‌افزار، از دو آزمایش ساده استفاده شده است. البته انجام تنها دو آزمایش، صحت عملکرد نرم‌افزار را تضمین نمی‌کند؛ ولی به دلیل کوچک بودن هسته نرم‌افزار، می‌توان پس از انجام آزمون‌های استاندارد نرم‌افزاری، از صحت عملکرد آن با اطمینان بالایی آسوده خاطر بود. در آزمایش اول، وظیفه تجمع ربات‌ها پیاده‌سازی خواهد شد؛ و در آزمایش دوم به پیاده‌سازی وظیفه همگام‌سازی ربات‌ها پرداخته خواهد شد.

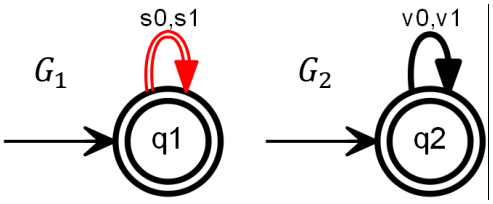
### ۴-۶-۱- پیاده‌سازی وظیفه تجمع ربات‌ها

در این آزمایش، وظیفه‌ی تجمع ربات‌ها پیاده‌سازی خواهد شد. گردهمایی ربات‌ها در یک محیط همگن، هدف این آزمایش است. این وظیفه معمولا در وظایف بزرگتر و پیچیده‌تر مانند تشکیل الگو و خود اتصالی مورد استفاده قرار می‌گیرد. الگوریتم مورد استفاده برای گردهمایی از ایده مطرح شده در [۱۱۷] بهره برده است که نیاز به محاسبات بسیار کمی دارد و برای رباتیک جمعی مناسب است. برای پیاده‌سازی این روش، هر ربات باید به یک حسگر دودویی $I$ مجهز باشد که در صورت مشاهده سایر ربات‌ها در مسیر دید ربات، مقدار یک و در غیراینصورت مقدار صفر را برمی‌گرداند. در صورتی که مقدار حسگر $I$ برابر با صفر باشد، ربات در یک مسیر دایره‌ای شکل، با سرعت $(v_{l0}, v_{r0}) = (-0.7, -1)$ و در جهت ساعتگرد رو به عقب حرکت می‌کند؛ و در صورتی که مقدار حسگر برابر با یک باشد، ربات با سرعت $(v_{l1}, v_{r1}) = (1, -1)$ در جهت ساعتگرد به دور خود می‌چرخد. $v_{l0}$ و $v_{r0}$ به ترتیب،

سرعت‌های موتور مربوط به چرخ‌های چپ و راست ربات هستند. مقادیر گزارش شده برای سرعت به بازه [۱،۱-] نگاشت شده‌اند. این مقادیر در مرحله پیاده‌سازی به بازه واقعی سرعتی که ربات پشتیبانی می‌کند، برگردانده خواهند شد. مرجع [۱۱۷] برای پیاده‌سازی حسگر $I$، از دوربین VGA در E-puck بهره برده است. این دوربین تصاویر را با وضوح ۶۴۰ * ۴۸۰ برمی‌گرداند؛ حافظه E-puck به قدری کوچک است که امکان ذخیره‌سازی یک تصویر با این ابعاد را ندارد. به همین دلیل، مرجع [۱۱۷] پس از کاهش ابعاد تصویر، تنها یک ستون به ارتفاع ۱۵ پیکسل و عرض ۱ پیکسل را استخراج کرده و با رای‌گیری از رنگ پیکسل‌های موجود در این ستون، در مورد وجود یا عدم وجود شیء در مقابل ربات تصمیم می‌گیرد.

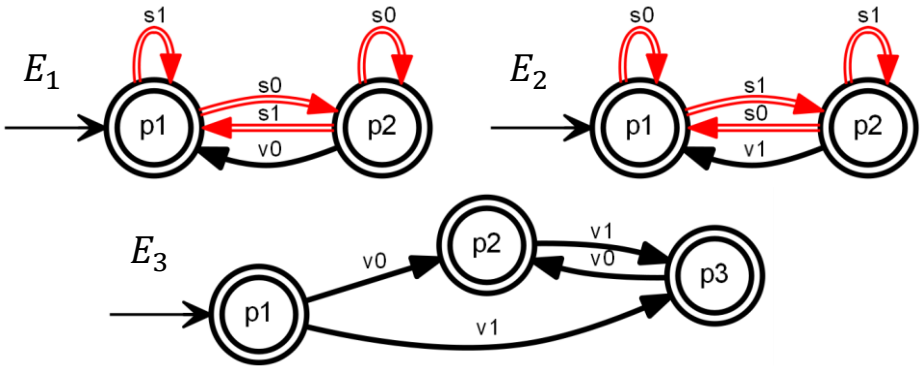
در این آزمایش، روش توضیح داده شده در بالا با مقداری تغییر و با استفاده از ابزار پیشنهادی پیاده‌سازی شده است. روش استفاده شده در این بخش از حسگر مجاورت‌سنج به جای دوربین بهره برده است. محدودیت استفاده از حسگر بسیار بیشتر از دوربین است. زیرا بازه دید دوربین E-puck در حدود ۱۴۰ سانتی‌متر است، در صورتی که حسگر مجاورت‌سنج دارای برد ۸ سانتی‌متری است. علاوه‌براین، با استفاده از دوربین می‌توان وجود اشیاء رنگی و یا خاکستری را تشخیص داد. در صورتی که حسگر مجاورت‌سنج تنها وجود اشیاء در مجاورت ربات را گزارش می‌کند و هیچ اطلاعاتی از نوع شیء مربوطه در اختیار قرار نمی‌دهد. همان‌طور که می‌دانیم، E-Puck دارای هشت حسگر مجاورت‌سنج است. روش پیشنهادی از ترکیب حسگرهای شماره صفر و هفت برای تشخیص سایر ربات‌ها بهره می‌برد. خروجی این حسگرها مقداری در بازه [۱-۰] است؛ صفر به معنی عدم وجود شیء و یک به معنی وجود شیء در نزدیک‌ترین فاصله ممکن به حسگر می‌باشد. برای ترکیب دو حسگر شماره صفر و هفت، بیشینه مقدار خروجی آن‌ها مورد استفاده قرار گرفته است. طراحی سوپروایزر نیز کاملا بر اساس مراحل توضیح داده شده در بخش‌های قبل صورت می‌گیرد.

شکل ۴-۲۱، مدل‌های رفتار آزاد مربوط به وظیفه‌ی تجمع ربات‌ها را نمایش داده است. مدل رفتار آزاد $G_1$ بیانگر عملکرد حسگر دودویی مورد بحث است. رخدادهای غیرقابل کنترل $s_0$ و $s_1$ به ترتیب معادل دو مقدار خروجی $I = 0$ و $I = 1$ می‌باشند. مدل رفتار آزاد $G_2$ حرکت‌های ممکن ربات را توصیف می‌کند. رخدادهای قابل کنترل $v_0$ و $v_1$ به ترتیب معادل دو سرعت موتورهای ربات $(v_{l0}, v_{r0})$ و $(v_{l1}, v_{r1})$ می‌باشند. هر حالت در این دو مولد، هم حالت ابتدایی و هم علامت‌دار است.



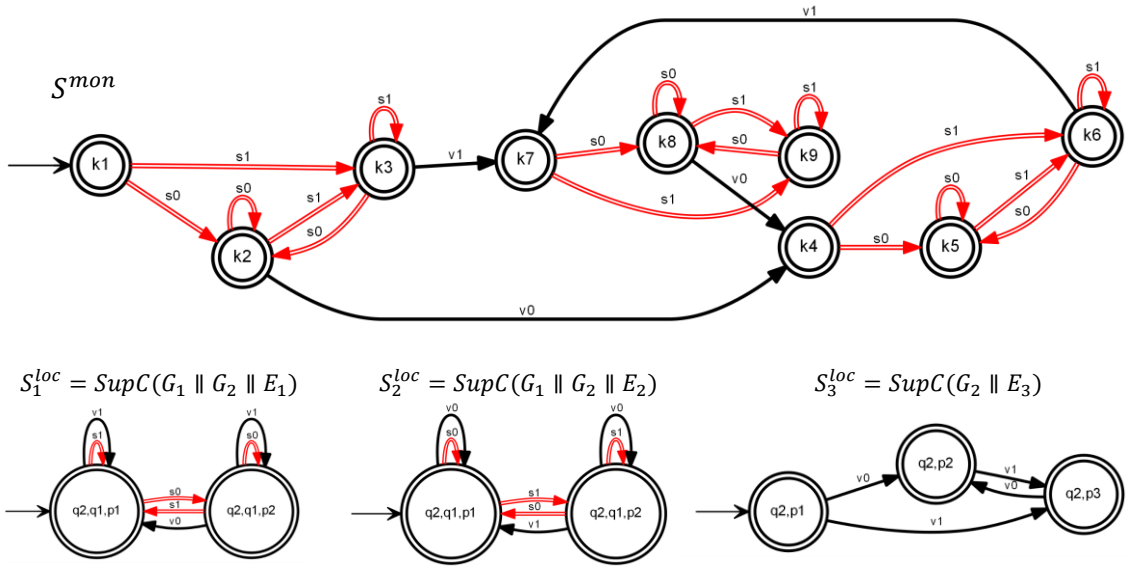

**شکل ۴-۲۱: مدل‌های رفتار آزاد مربوط به وظیفه‌ی تجمع ربات‌ها با استفاده از ptSCT.**

شکل ۴-۲۲، ضابطه‌های کنترلی پیشنهادی برای وظیفه‌ی تجمع ربات‌ها را نمایش داده است. ضابطه کنترلی $E_1$، رخداد $v_0$ را تنها در صورتی فعال می‌کند که هیچ شیءای در معرض دید ربات وجود نداشته باشد. ضابطه کنترلی $E_2$، رخداد $v_1$ را تنها در صورتی فعال می‌کند که شیءای در معرض دید ربات وجود داشته باشد. وقتی سرعت موتور ربات تنظیم می‌شود، ربات تا بی‌نهایت بر اساس آن سرعت حرکت می‌کند. بنابراین نیازی به اتفاق افتادن مداوم رخدادهای $v_0$ یا $v_1$ وجود ندارد. هدف از ضابطه کنترلی $E_3$ همین امر است؛ یعنی اجرای نوبتی رخدادهای $v_0$ و $v_1$. عدم وجود ضابطه کنترلی $E_3$ موجب عملکرد اشتباه ربات‌ها نخواهد شد و تنها از یک امر بی‌اثر جلوگیری می‌کند.




**شکل ۴-۲۲: ضابطه‌های کنترلی مربوط به وظیفه‌ی تجمع ربات‌ها با استفاده از ptSCT.**

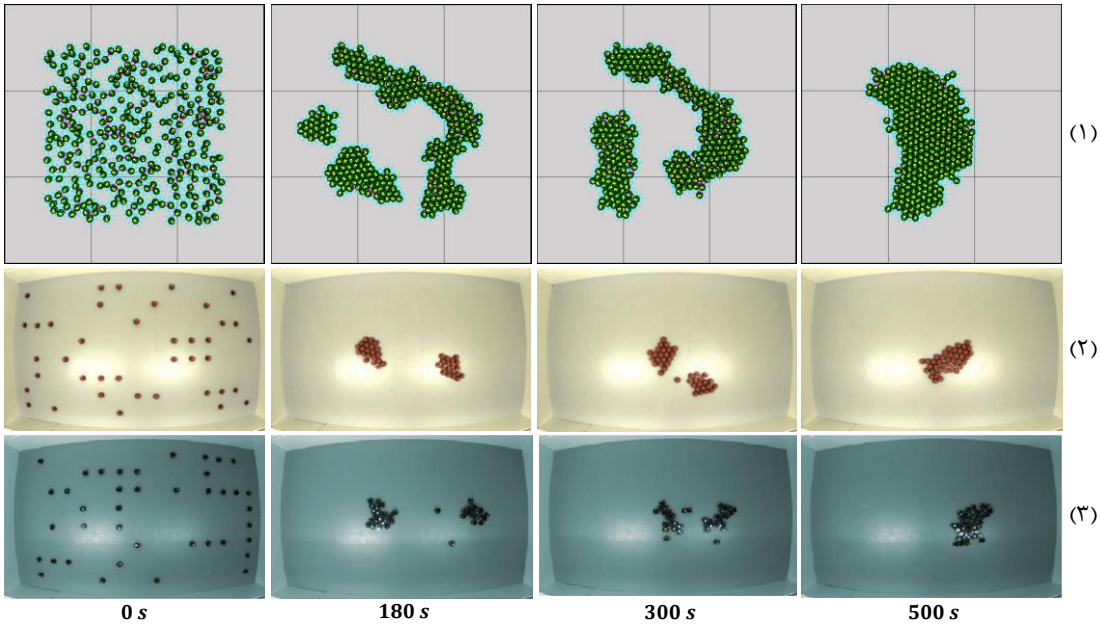
شکل ۴-۲۳، سوپروایزر یکپارچه و سوپروایزرهای ماژولار محلی محاسبه شده توسط ابزار پیشنهادی را به تصویر کشیده است. برای اجرای استراتژی تجمع ربات‌ها در محیط شبیه‌سازی و یا در دنیای واقعی، کافی است که یکی از این دو نوع سوپروایزر به عنوان نرم‌افزار کنترلی ربات‌ها مورد استفاده قرار گیرند. نتیجه استفاده از هر یک از دو نوع سوپروایزر برابر است، هرچند مجموع تعداد حالت‌ها و رخدادهای سوپروایزرهای ماژولار محلی بسیار کمتر است. این تفاوت در مولدهای بزرگتر، بارزتر خواهد شد.




**شکل ۴-۲۳: سوپروایزر یکپارچه ($S^{mon}$) و سوپروایزرهای ماژولار محلی مربوط به وظیفه تجمع ربات‌ها.**

برای انجام شبیه‌سازی‌ها از ۳۰۰ ربات E-puck بهره برده شده است. به ورودی حسگر مجاورت‌سنج، نویز گوسی با انحراف معیار ۰٫۳ اعمال شده است تا به رفتار دنیای واقعی نزدیک‌تر شود. نویز مشابهی به عملکرد موتورهای ربات نیز اعمال گشته است. ابعاد محیط مورد استفاده برای انجام آزمایش‌ها برابر است با ۳۰۰ سانتی‌متر در ۳۰۰ سانتی‌متر و ارتفاع نیم متر. برای افزایش سرعت شبیه‌سازی، در هر ثانیه، ۱۰ دور از الگوریتم اجرا می‌شود. دوربین در ارتفاع دو و نیم متری از محیط شبیه‌سازی قرار داده شده و در زمان‌های مشخص، تصاویری تهیه می‌کند.

نتیجه آزمایش انجام شده در شکل ۴-۲۴ نمایش داده شده است. نتیجه بدست آمده با دو روش از جدیدترین روش‌های موجود مورد مقایسه قرار گرفته است [۴۹, ۱۱۷]. هر دو روش مورد مقایسه از دوربین VGA ربات استفاده کرده‌اند که برد چند ده برابری نسبت به حسگر مجاورت‌سنج ربات دارد. با این حال، مشاهده می‌شود که نتیجه بدست آمده توسط روش پیشنهادی کاملا مناسب است. علاوه‌براین، تعداد ربات‌های مورد استفاده در آزمایش انجام شده تقریبا هفت برابر بیشتر از دو روش مقایسه شده می‌باشد.




**شکل ۴-۲۴: مقایسه دنباله تصاویر گرفته شده در زمان‌های صفر، ۱۸۰، ۳۰۰ و ۵۰۰ ثانیه از آزمایش تجمع ربات‌ها. (۱) روش پیاده‌سازی شده در این بخش با ۳۰۰ ربات، (۲) روش پیشنهادی در [۱۰۸] با ۴۰ ربات و (۳) روش پیشنهادی در [۱۰۶] با ۴۰ ربات.**

همان‌طور که در نتیجه آزمایش مشاهده می‌شود، ربات‌ها ابتدا دسته‌های مجتمع کوچک تشکیل داده و سپس این دسته‌ها در هم ادغام شده و دسته‌های بزرگتر را ساخته‌اند. این رفتار در [۱۰۸] نیز گزارش شده است. این آزمایش ۲۰ مرتبه تکرار شده است تا یک میانگین از رفتار ربات‌ها بدست آید. در ۱۷ مرتبه از ۲۰ آزمایش صورت گرفته، ربات‌ها بعد از ۵۰۰ ثانیه در یک دسته واحد تجمع کردند؛ در سه مرتبه باقیمانده، دو دسته از ربات‌ها تشکیل شده است. به دلیل برد بسیار کم حسگر مجاورت‌سنج، در صورتی که فاصله ربات‌ها از هم زیاد باشد، تجمع آن‌ها نیاز به زمان بسیار بیشتری نسبت به روش‌های ارائه شده در [۴۹, ۱۱۷] دارد.

### ۴-۶-۲- پیاده‌سازی وظیفه همگام‌سازی

همگام‌سازی از وظایف پرکاربرد در رباتیک جمعی است؛ زیرا گام اول در بسیاری از وظایف پیچیده‌تر را تشکیل می‌دهد. برای نمونه، از همگام‌سازی می‌توان برای تشخیص ربات‌های دچار مشکل شده استفاده کرد [۲۵]؛ و یا باکتری‌ها برای تشخیص اندازه جمعیت از همگام‌سازی بهره می‌برند [۲۶]. در این بخش، وظیفه همگام‌سازی هم با استفاده از SCT و هم با استفاده از ptSCT پیاده‌سازی شده است؛ به این ترتیب می‌توان مقایسه‌ای بین عملکرد این دو انجام داد و پیچیدگی فضایی سوپروایزرهای محاسبه شده از این دو روش را سنجید.

در وظیفه پیاده‌سازی شده در این بخش، هر ربات دارای یک شمارنده داخلی $counter$ است. این شمارنده از یک مقدار تصادفی آغاز به شمارش کرده و پس از رسیدن به یک مقدار مشخص $T$، شمارش را از صفر آغاز می‌کند. بنابراین، سقف شمارش برای همه ربات‌ها یکسان، ولی مقدار آغازین برای هر ربات متفاوت است. ربات‌ها باید بتوانند پس از مدتی، شمارنده‌هایشان را با یکدیگر همگام کنند. این مسئله در واقع همان همگام‌سازی فازها[۱] است. هر ربات وقتی به سقف شمارشش می‌رسد، با روشن کردن چراغ‌های ال‌ای‌دی خود برای مدت معین، به سایرین علامت می‌دهد. هر رباتی که چراغ سایر ربات‌ها را می‌بیند، با استفاده از فرمول ۹ مقدار شمارنده خود را تغییر می‌دهد.

$$counter = counter + \alpha \,.\, counter \qquad (۲۲)$$

ثابت سراسری $\alpha$ مقداری در بازه [۰-۱] بوده و برای همه ربات‌ها یکسان است. ثابت سراسری $T$ و شمارنده $counter$ مقادیری صحیح و مثبت هستند. الگوریتم توضیح داده شده در شکل ۴-۲۵ به نمایش گذاشته شده است. تابع $signal$ وظیفه علامت دادن به سایرین را انجام می‌دهد. تابع $rand$ مقداری صحیح و در بازه $[1, T]$ تولید می‌کند. شرط $signal\_received$ در صورتی که ربات، علامتی از سایرین دریافت کرده باشد، برقرار بوده و در غیراینصورت برقرار نخواهد بود.

```
1  Let α be a global constant in the range of [0:1]
2  Let T be a global positive integer constant
3
4  procedure PhaseSynchronization
5      Let counter be the local counter of the current supervisor
6      counter rand(1,T)
7      while true do
8          counter ← counter + 1
9          if counter > T then
10             signal
11             reset counter
12         if signal_received then
13             counter ← counter + a * counter
```

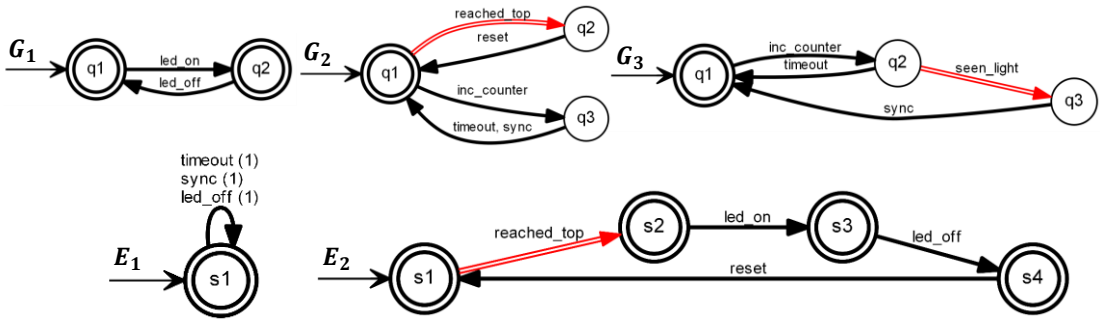
**شکل ۴-۲۵: الگوریتم همگام‌سازی پیاده‌سازی شده.**

در این وظیفه از دو محرک و حسگر ربات E-Puck استفاده شده است. چراغ‌های ال‌ای‌دی و دوربین ۳۶۰ درجه. هر ربات به وسیله دوربینش، روشن بودن چراغ سایرین را تشخیص داده و شمارنده خود را تغییر می‌دهد. شکل ۴-۲۶، مدل‌های رفتار آزاد و ضابطه‌های کنترلی پیشنهاد شده برای وظیفه همگام‌سازی را نمایش داده است. مدل رفتار آزاد $G_1$ به قابلیت ربات در روشن و خاموش کردن چراغ‌های ال‌ای‌دی اشاره دارد. رخدادهای قابل‌کنترل $led\_on$ و $led\_off$ به ترتیب چراغ‌های ربات را روشن و خاموش می‌کنند. مدل $G_2$ شکل عملکرد شمارنده ربات را توصیف می‌کند. ربات تا زمانی که به سقف

[۱] Phase Synchronization

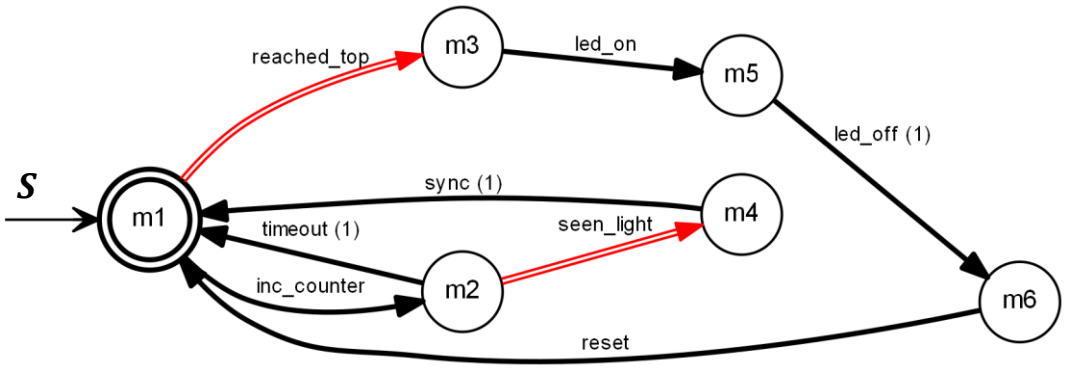
شمارش نرسیده است (رخداد غیرقابل‌کنترل $reached\_top$)، شمارنده‌اش را افزایش می‌دهد (رخداد $inc\_counter$)؛ درغیراینصورت، شمارنده را صفر خواهد کرد (رخداد $reset$). بعد از هر شمارش، رخداد $timeout$ اتفاق خواهد افتاد که به اندازه یک واحد زمانی از ساعت سیستم تاخیر اعمال می‌کند. این امر موجب می‌شود تا در هر گام از اجرای سیستم، شمارنده تنها یک واحد افزایش یابد. مدل $G_3$ عملکرد ربات در ارتباط با سایرین را توصیف می‌کند. ربات تا زمانی که روشنایی در اطرف خود مشاهده نکرده است، به شمارش ادامه می‌دهد. در صورت مشاهده روشنایی، شمارنده خود را توسط رخداد $sync$ همگام خواهد کرد.

ضابطه کنترلی $E_2$، عملکرد ربات در زمان رسیدن شمارنده به سقف شمارش را تعیین می‌کند. به این شکل که ابتدا چراغ ال‌ای‌دی ربات روشن شده، پس از مدتی خاموش می‌شود و آنگاه شمارنده صفر می‌گردد. ضابطه کنترلی $E_1$، زمان رخدادهای زمان‌دار را تعیین کرده است. اجرای این رخدادها باید با یک واحد زمانی تاخیر صورت گیرد.




**شکل ۴-۲۶: مدل‌های رفتار آزاد ($G_1$، $G_2$ و $G_3$) و ضابطه‌های کنترلی ($E_1$) مربوط به وظیفه همگام‌سازی با استفاده از ptSCT.**

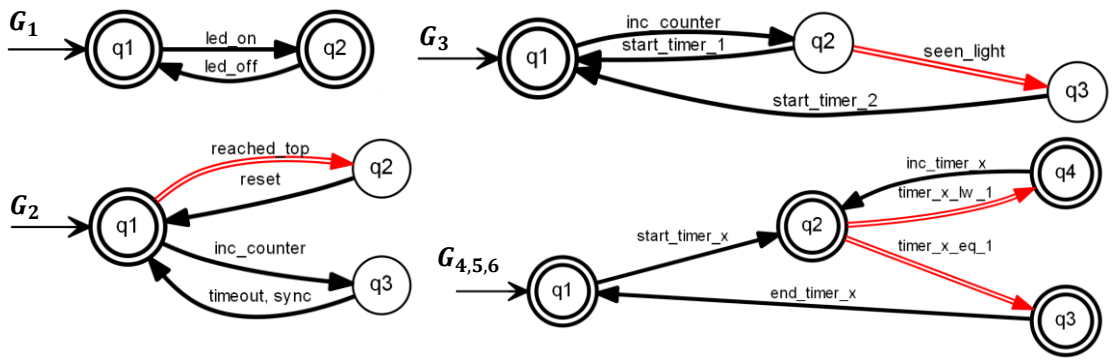
شکل ۴-۲۷، سوپروایزر بدست آمده از ترکیب مدل‌های رفتار آزاد و ضابطه‌های کنترلی را نشان می‌دهد. همان‌طور که در شکل مشخص است، منطق محاسبه شده کاملا صحیح است. علاوه‌براین، زمان رخدادهای زمان‌دار نیز به درستی اعمال شده است.




**شکل ۴-۲۷: سوپروایزر محاسبه شده برای وظیفه همگام‌سازی با استفاده از ptSCT.**

در ادامه، برای مقایسه عملکرد ptSCT و SCT، مدل‌های رفتار آزاد و ضابطه‌های کنترلی پیشنهاد شده با استفاده از SCT نیز ارائه شده‌اند. به این ترتیب می‌توان به روشنی به تاثیر مثبت ptSCT در فشردگی و سادگی هرچه بیشتر طراحی پی برد. در طراحی‌های جدید از مولفه‌های احتمالات و زمان استفاده نشده است.

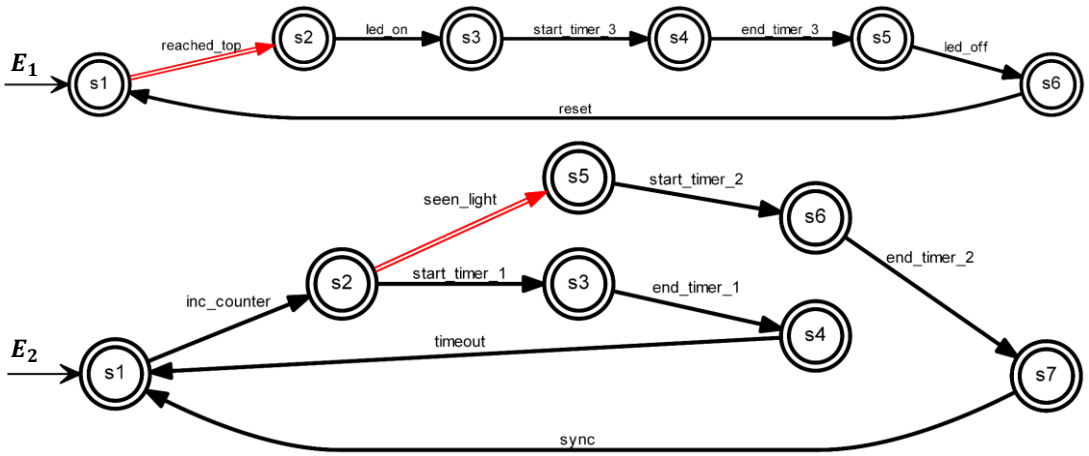
شکل ۴-۲۸ مدل‌های رفتار آزاد طراحی شده با استفاده از SCT را نمایش داده است. مدل‌های $G_4$ تا $G_6$، مدل‌های جدیدی هستند که به طراحی قبلی اضافه شده‌اند و وظیفه ایجاد تاخیر را بر عهده دارند. عدم وجود رخدادهای زمان‌دار، استفاده از چنین مدل‌هایی را اجبار کرده است. شمارنده $x$ ام با رویداد رخداد $start\_timer\_x$، شمارش را آغاز کرده و به کمک دو رویداد غیرقابل کنترل $timer\_x\_lw\_1$ و $timer\_x\_eq\_1$، یک گام از اجرای سیستم را می‌شمارد. در پایان شمارش، رخداد $end\_timer\_x$ فعال خواهد شد. رویداد$inc_timer_x$ یک واحد به شمارنده $x$ ام اضافه می‌کند. رخداد $timer\_x\_eq\_1$ زمانی که شمارنده $x$ ام برابر با یک باشد، فعال می‌شود و رخداد $timer\_x\_lw\_1$ زمانی که شمارنده $x$ ام کوچکتر از یک باشد، فعال می‌شود.



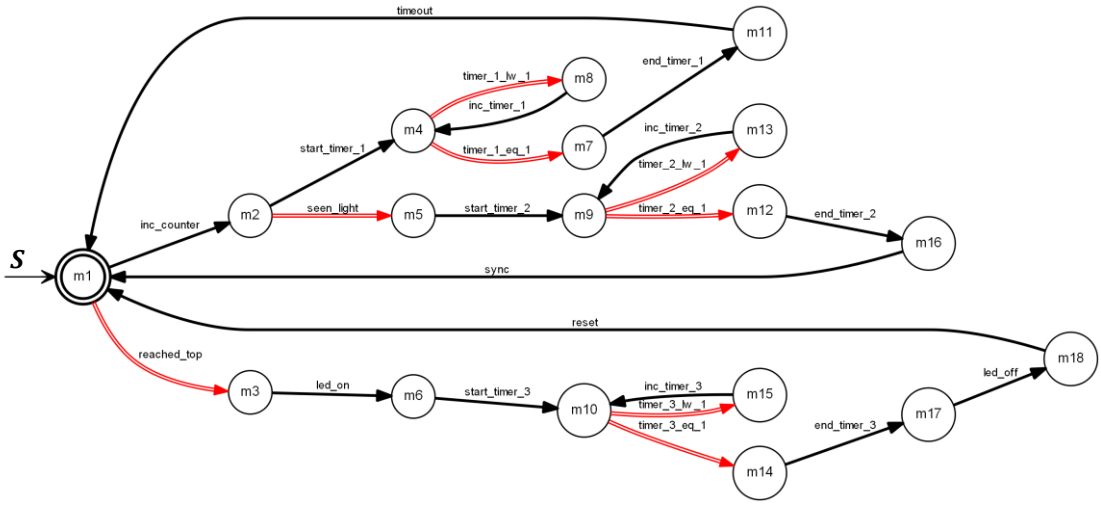

**شکل ۴-۲۸: مدل‌های رفتار آزاد ($G_1$ تا $G_6$) مربوط به وظیفه همگام‌سازی با استفاده از SCT: $x \in \{1, 2, 3\}$.**

شکل ۴-۲۹ ضابطه‌های کنترلی جدید را نمایش داده است. وظیفه هر دو ضابطه‌ی کنترلی $E_1$ و $E_2$، اضافه کردن تاخیرهای موردنیاز قبل از اجرای رویدادهایی است که نیاز به تاخیر دارند؛ یعنی سه رویداد $timeout$، $sync$ و $led\_off$.

سوپروایزر محاسبه شده در شکل ۴-۳۰ به دلیل وجود سه شمارنده در آن، بسیار بزرگتر از معادل محاسبه شده در شکل ۴-۲۷ است. این امر علاوه‌بر مشکلات برشمرده شده در بخش قبل، موجب افزایش دشواری در تحلیل و اشکال‌یابی از طراحی انجام شده نیز خواهد شد. جدول ۴-۱، تعداد حالت‌ها، گذارها و فضای اشغال شده (با توجه به فرمول (۲۱)) توسط سوپروایزر را با استفاده از SCT و ptSCT ارائه کرده است. نتایج ارائه شده در این جدول، نشان از اختلاف فاحش دو روش مورد بررسی دارد؛ با بزرگ‌تر شدن مسئله، این اختلاف افزایش پیدا می‌کند.




**شکل ۴-۲۹: ضابطه‌های کنترلی ($E_1$ و $E_2$) مربوط به وظیفه همگام‌سازی با استفاده از SCT.**




**شکل ۴-۳۰: سوپروایزر محاسبه شده برای وظیفه همگام‌سازی با استفاده از SCT**

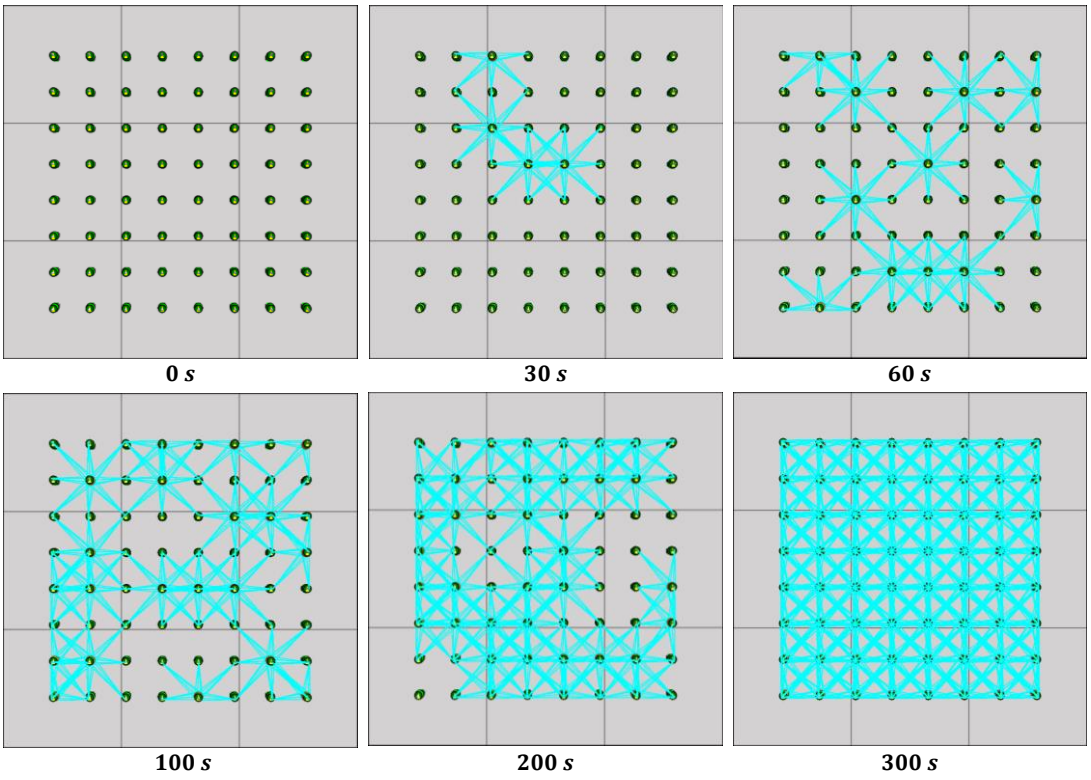
**جدول ۴-۱: مقایسه تعداد حالت‌ها، گذارها و فضای اشغالی توسط سوپروایزری محاسبه شده به روش SCT و ptSCT**

| | ptSCT | SCT |
|---|---|---|
| تعداد حالات (عدد) | ۶ | ۱۸ |
| تعداد گذارها (عدد) | ۸ | ۲۳ |
| فضای اشغالی (بایت) | ۸۴ | ۲۴۱ |

برای شبیه‌سازی وظیفه همگام‌سازی از ۶۴ ربات در محیطی همگن بهره برده شده است. ربات‌ها به صورت یک شبکه ۸ در ۸ چیده شده و فاصله‌ی آن‌ها به گونه‌ای تعیین شده که بتوانند چراغ‌های یکدیگر را ببینند. سقف شمارش نیز برابر با $T = 50$ قرار داده شده است.

شکل ۱۶ چیدمان اولیه ربات‌ها و وضعیت همگام‌سازی ربات‌ها در زمان‌های مختلف را نمایش داده است. وقتی ربات در یک جهت از اطراف خود، روشنایی ببیند، یک خط در همان جهت در محیط شبیه‌ساز ترسیم می‌شود تا وضعیت همگام‌سازی ربات‌ها توسط ببیننده قابل تشخیص باشد. بنابراین در ثانیه ۳۰۰ از شروع اجرای سیستم، همه ربات‌ها همگام شده‌اند. این امر از درستی عملکرد سوپروایزر طراحی شده حکایت دارد.

در دنباله‌ی تصاویر ارائه شده در شکل ۱۶، روند همگام‌سازی ربات‌ها نیز قابل مشاهده است. ابتدا ربات‌های همسایه به تدریج همگام شده و دسته‌های کوچکِ همگام را تشکیل داده‌اند؛ سپس دسته‌های کوچک در هم ادغام شده و دسته‌های بزرگ‌تر همگام را ساخته‌اند که در نهایت به همگام‌سازی کلیه ربات‌ها منجر شده است.




**شکل ۴-۳۱: دنباله تصاویر گرفته شده در زمان‌های صفر، ۳۰، ۶۰، ۱۰۰، ۲۰۰ و ۳۰۰ ثانیه از آزمایش همگام‌سازی ربات‌ها.**

## ۴-۷- جمع‌بندی

در این فصل به تشریح ابزار جامع پیشنهادی پرداخته شد. این ابزار قادر است تمامی مراحل از طراحی و مدل‌سازی گرفته تا تولید خودکار کد را مدیریت کند. هشدار اشتباهات انجام شده در طراحی به طراح، انجام خودکار عملیات سنتز، حذف حالت‌های غیرقابل دسترس و حالت‌های بد، رسم مولدها و تولید خودکار کد کنترل کننده از جمله قابلیت‌های ابزار پیشنهاد شده می‌باشد. کد تولید شده توسط این ابزار، در صورتی قابل اجرا بر روی سکوی ARGoS خواهد بود که مطابق با قواعد این سکو تولید شده باشد. بنابراین قبل از معرفی ابزار، به پیاده‌سازی یک لایه بر روی سکوی ARGoS پرداخته شد. این لایه خود از سه بخش تشکیل شده است که معرفی شدند. بخش‌های توضیح داده شده در این فصل به صورت مختصر از قرار زیر بودند:

۱. ابزار جامع طراحی، مدل‌سازی و تولید خودکار کد
۲. لایه نرم‌افزاری برای اجرای سوپروایزرهای تولید شده توسط ابزار
    ۱. لایه سوپروایزرها
    ۲. لایه اجرا کننده مولدها
    ۳. لایه رویه‌های عملیاتی

لایه سوپروایزرها و لایه اجرا کننده مولدها مستقل از ربات هستند و برای هر پروژه‌ای قابل استفاده می‌باشند. بنابراین با توجه به کوچک بودنشان و استفاده مداوم در پروژه‌های مختلف، تایید صحت عملکردشان با اطمینان بالایی ممکن است. تنها لایه سوم، وابسته به نوع ربات مورد استفاده در آزمایش است که باید توسط برنامه‌نویس نوشته شود. هرچند توابع فراخوان ارائه شده در این لایه نیز دارای عملکرد ساده‌ای هستند؛ چون هریک متناظر با یک رخداد در مدل مربوطه می‌باشند. درضمن، این توابع نیز قابلیت استفاده مجدد دارند. در نتیجه، سه لایه پیشنهادی، در پروژه‌های دنیای واقعی نیز به خوبی قابل استفاده هستند.

علاوه‌براین، با ذکر یک مثال، به مقایسه عملکرد ptSCT در برابر SCT پرداخته شد؛ طراحی انجام شده حکایت از اختلاف بارز فضای اشغالی و مولد تولید شده داشت. به دلیل حافظه و قدرت پردازشی محدود ربات‌ها، طراحی فشرده‌تر از اولویت بالاتری برخوردار است. درضمن، طراحی کوچک‌تر طبیعتا ساده‌تر قابل انجام است؛ در نتیجه قابلیت تحلیل و اشکال‌یابی بالاتری نیز خواهد داشت.

# فصل

# ۵- روش پیشنهادی برای پیاده‌سازی وظیفه کاوش هدف

## ۵-۱- مقدمه

در فصل قبل به معرفی ابزار جامع پیشنهادی پرداخته شد. در این فصل، با استفاده از ابزار معرفی شده، به طراحی و پیاده‌سازی موضوع اصلی این رساله، یعنی کاوش هدف خواهیم پرداخت. علت‌های انتخاب این وظیفه در ادامه فهرست شده‌اند:

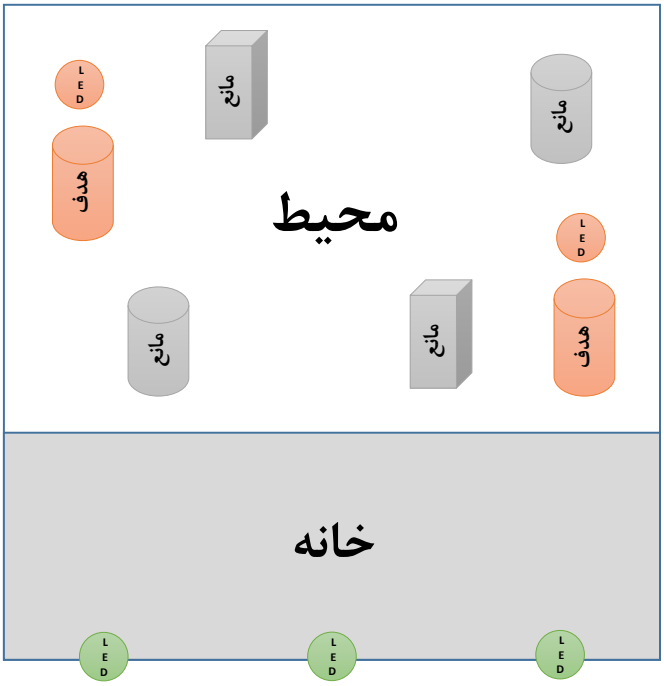
- کاوش هدف یک وظیفه‌ی شناخته شده است [۱۱۵, ۱۲۰–۱۲۴]؛ بنابراین روش‌های متعددی در این حوزه ارائه شده‌اند که می‌توان از آن‌ها برای بهبود عملکرد روش پیشنهادی ایده گرفت و یا به مقایسه پارامترهای مورد استفاده در روش‌های گوناگون پرداخت.
- کاوش هدف یک وظیفه نسبتا پیچیده است [۷۵]؛ که می‌توان از این ویژگی برای نشان دادن قدرت و انعطاف ptSCT در طراحی و مدل‌سازی بهره برد.
- این وظیفه خود از وظیفه‌های کوچکتری تشکیل می‌شود که هریک قابلیت استفاده مجدد در پروژه‌های دیگر را دارند. وظیفه‌هایی از قبیل تجمع ربات‌ها، پخش شدن ربات‌ها، تخصیص وظایف، جستجو در محیط و بازگشت به خانه چند نمونه از این موارد هستند [۷].
- با توجه به ساختار وظیفه کاوش هدف که خود از چند زیروظیفه تشکیل شده است، می‌توان به قابلیت طراحی ماژولار با استفاده از تئوری کنترل سوپروایزری نیز تاکید کرد.
- کد تولید شده توسط ابزار جامع پیشنهادی برای وظیفه کاوش هدف، قابل رقابت با پیاده‌سازی‌های خاص منظوره و سایر روش‌های موجود است. برای نشان دادن این موضوع، سناریویی نسبتا پیچیده انتخاب گشته و در مرحله‌ی شبیه‌سازی، جوانب متعددی از آن مورد بررسی قرار گرفته است؛ تا به این ترتیب، کیفیت کد تولید شده مورد سنجش قرار گیرد.

در ابتدای این فصل به تحلیل نیازهای مسئله پرداخته و توانایی‌های فیزیکی ربات‌ها مرور شده است. سپس براساس توانایی‌های مطرح شده، به پیشنهاد مدل‌های رفتار آزاد و ضابطه‌های کنترلی موردنیاز مسئله پرداخته شده است. در پایان این فصل، سوپروایزرهای نهایی محاسبه شده از مدل‌های پیشنهادی ارائه شده‌اند؛ و در نهایت جمع‌بندی صورت گرفته است.

## ۵-۲- تحلیل نیازهای مسئله

قابلیت‌های فیزیکی ربات‌ها برای پیاده‌سازی وظیفه کاوش هدف در ادامه مطرح شده است. رخدادهای مورد استفاده در طراحی مدل‌ها با توجه به این توانایی‌ها تعریف می‌شوند. جدول ۵-۱ در پایان این بخش، فهرستی از رخدادهای مورد استفاده در طراحی را به همراه محرک یا حسگر مربوط به هر یک ارائه کرده است.

محیط مورد استفاده برای پیاده‌سازی وظیفه کاوش هدف، یک محیط بسته است که با چهار دیوار محصور گشته است. رنگ زمین در محدوده خانه متفاوت از سایر نواحی است. بنابراین ربات‌ها نیاز به حسگر زمین دارند تا بتوانند رنگ‌ها را تمیز دهند (شکل ۵-۱). برای اینکه ربات‌ها بتوانند خانه را از نواحی دورتر هم ببینند، لامپ‌های رنگی در بالای محدوده خانه قرار داده می‌شوند؛ بنابراین ربات‌ها نیاز به حسگر نور نیز دارند. برای جلوگیری از برخورد ربات‌ها با یکدیگر، موانع، اهداف و دیوارها، نیاز به حسگر مجاورت‌سنج وجود دارد. برای تشخیص اهداف، از اجسام رنگی در محیط استفاده شده است؛ برای دیدن این اجسام رنگی نیاز به دوربین ۳۶۰ درجه یا دوربین VGA هست. ربات‌ها می‌توانند با استفاده از محرک ال‌ای‌دی، وضعیت فعلی خود را با رنگ‌های متفاوت نمایش دهند (در حال جستجو، در حال بازگشت به خانه، در حال استراحت و ...). محرک موتورها نیز برای حرکت و چرخش ربات‌ها ضروری است. ربات E-puck که ربات منتخب این رساله برای انجام وظیفه کاوش هدف است، قابلیتی برای برداشتن و حمل اهداف ندارد. بنابراین از ترفندهای شبیه‌سازی برای نمایش برداشت، حمل و رهاسازی اهداف بهره برده شده است؛ به صورتی که با رسیدن ربات به یک هدف مشخص، هدف از محیط حذف و در بالای سر ربات نمایش داده می‌شود. در برخی مقالات، مکان خانه در مرکز محیط قرار داده می‌شود؛ موقعیت خانه در طراحی ارائه شده تاثیری ندارد. از طرفی شکل پیاده‌سازی اهداف منعطف است و با توجه به امکانات موجود و قابلیت‌های ربات تعیین می‌شود.




**شکل ۵-۱: نمایی از محیط پیشنهادی برای پیاده‌سازی وظیفه کاوش هدف.**

با توجه به توضیحات داده شده، از ۴ حسگر و ۲ محرک در ربات E-puck برای پیاده‌سازی وظیفه کاوش هدف استفاده خواهد شد. همان‌طور که در فصل ۴ توضیح داده شد، از برد توسعه لینوکس برای گسترش توانایی‌های E-puck بهره برده شده است. در جدول ۵-۱، فهرست رخدادهای موردنیاز برای طراحی مدل‌های رفتار آزاد ارائه شده است. برای هر رخداد، قابل کنترل بودن آن، عملکرد آن و حسگر یا محرک مربوط به آن، بیان شده است.

**جدول ۵-۱: تعریف رخدادهای موردنیاز با توجه به توانایی فیزیکی ربات**

| نام رخداد | قابل کنترل | عملکرد | حسگر/محرک مرتبط |
|---|---|---|---|
| saw-home | خیر | دیدن خانه | حسگر نور |
| turn-to-home | بلی | چرخش به سمت خانه | محرک موتورها |
| turn-inv-home | بلی | چرخش در جهت عکس خانه | محرک موتورها |
| saw-obs | خیر | دیدن مانع | حسگر مجاورت‌سنج |
| turn-inv-obs | بلی | چرخش در جهت عکس مانع | محرک موتورها |
| turn-left | بلی | چرخش به چپ | محرک موتورها |
| turn-right | بلی | چرخش به راست | محرک موتورها |
| Go | بلی | حرکت به جلو | محرک موتورها |
| Stop | بلی | توقف | محرک موتورها |
| is-in-home | خیر | بررسی در خانه بودن | حسگر زمین |
| saw-food | خیر | دیدن غذا (هدف) | حسگر دوربین ۳۶۰ درجه |
| turn-to-food | بلی | چرخش به سمت غذا | محرک موتورها |
| have-food | خیر | داشتن غذا | محرک دستگیره (مجازی) |
| is-on-food | خیر | بررسی نزدیکی به غذا برای برداشتن آن | حسگر دوربین ۳۶۰ درجه |
| pick-food | بلی | برداشتن غذا | محرک دستگیره (مجازی) |

| | | | |
|---|---|---|---|
| pick-ended | خیر | اتمام برداشتن غذا | محرک دستگیره (مجازی) |
| drop-food | بلی | رها کردن غذا | محرک دستگیره (مجازی) |
| drop-ended | خیر | اتمام رها کردن غذا | محرک دستگیره (مجازی) |

## ۵-۳- طراحی مدل‌های رفتار آزاد

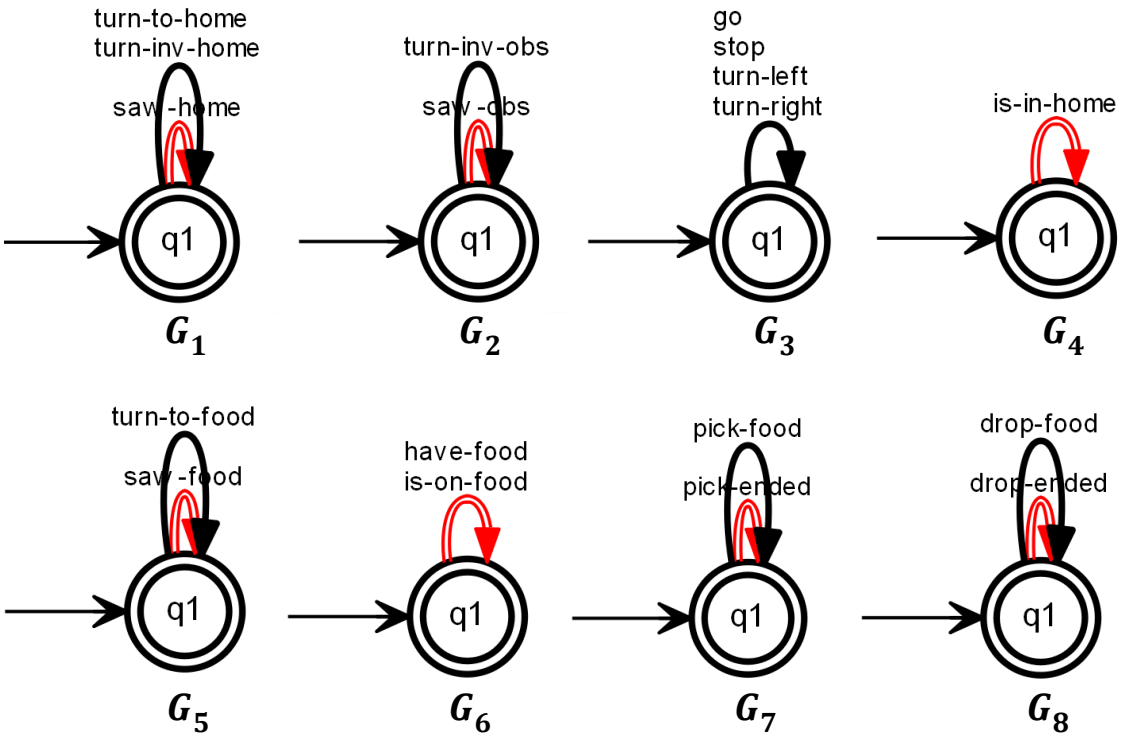
در این بخش، طراحی‌های پیشنهادی برای توصیف قابلیت‌های فیزیکی ربات‌ها در قالب مدل‌های رفتار آزاد ارائه گشته‌اند. مدل‌های ارائه شده در این بخش و بخش بعد، یکی از راه‌حل‌های ممکن این مسئله است و در هیچ جای رساله‌ی جاری، ادعایی مبنی بر تنها راه‌حل موجود بودن و یا بهترین راه‌حل موجود بودن، نشده است.

وظیفه کاوش هدف پیاده‌سازی شده در این رساله از مراحلی که در ادامه تشریح شده است، تشکیل می‌شود. وظیفه توسط تعداد ربات مشخصی آغاز می‌شود. ربات‌ها کار را از داخل خانه آغاز می‌کنند. سپس با رعایت عدم برخورد با موانع و سایر ربات‌ها از خانه خارج شده و به جستجوی اهداف می‌پردازنند. در صورتی که هدف موردنظر را پس از طی کردن تعداد گام مشخصی بیابند، هدف مربوطه را توسط محرک دستگیره گرفته و به خانه بازمی‌گردانند؛ در مسیر بازگشت نیز باید از موانع و سایر ربات‌ها دوری کنند. هیچ ارتباطی بین ربات‌های مختلف وجود نداشته و کاملا مستقل از هم عمل می‌کنند. در صورتی که یک ربات پس از طی کردن تعداد گام‌های مشخص، موفق به یافتن هیچ هدفی نگردد، به خانه بازگشته و مدتی را به استراحت می‌پردازد. تعیین سقف تعداد گام برای یافتن اهداف و مدت استراحت در خانه با توجه به نوع ربات و اندازه محیط و به صورت تجربی صورت می‌گیرد.

در مواردی جستجوی مستقل و انفرادی ربات‌ها نسبت به همکاری و تعاون در جستجو ارجحیت دارد. به عنوان مثال، در مواقعی که غذا زیاد است و به سادگی پیدا می‌شود یا هنگامی که اطمینان اطلاعات مخابره شده پایین باشد و یا در محیط‌هایی که یافتن سریع غذا دشوار است. ممکن است در برخی موارد عملکرد منفرد و گروهی نزدیک به هم باشد، در این صورت به دلیل هزینه‌ی کم‌تر حالت منفرد انتخاب می‌گردد.

همه‌ی مدل‌های رفتار آزاد پیشنهادی در شکل ۵-۲ نمایش داده شده است. برای سادگی و فشردگی بیشتر در نمایش مدل‌ها، گذارهای مختلف به شکل یک گذار با رخدادهای متعدد، نمایش داده شده‌اند. مدل رفتار آزاد $G_1$، به قابلیت‌های ربات برای دیدن خانه، چرخش به سمت خانه و چرخش در خلاف جهت خانه اشاره دارد. با استفاده از این مجموعه قابلیت‌ها، ربات می‌تواند از خانه خارج شده و به خانه بازگردد. مدل رفتار آزاد $G_2$ به قابلیت‌های ربات در دیدن موانع و چرخش در جهت عکس مسیری که

موانع در آن دیده شده‌اند، اشاره دارد. طبیعتا، بدون این قابلیت‌ها، ربات قادر به دوری جستن از موانع نخواهد بود. قابلیت‌های حرکتی ربات شامل حرکت به جلو، توقف، چرخش به راست و چپ توسط مدل رفتار آزاد $G_3$ توصیف شده‌اند. این قابلیت‌ها، اساس بسیاری از قابلیت‌های حرکتی دیگر هستند؛ برای مثال قابلیت چرخش به سمت خانه می‌توانست با استفاده از یکی از این قابلیت‌های پایه توصیف شود. مدل رفتار آزاد $G_4$، قابلیت تشخیص در خانه بودن/نبودن ربات را توصیف می‌کند. مدل رفتار آزاد $G_5$ به توانایی ربات در دیدن اهداف و چرخش به سمت آن‌ها اشاره دارد. رباتی که دارای محرک دستگیره است، باید دارای سه مجموعه قابلیت باشد: ۱) آیا در حال حاضر، هدفی توسط دستگیره گرفته شده است؟ - آیا هدفی در دسترس دستگیره برای گرفتن وجود دارد؟ ۲) هدف در دسترس را بگیر – آیا گرفتن هدف با موفقیت به پایان رسید؟ ۳) هدف گرفته شده را رها کن – آیا رهاسازی با موفقیت به پایان رسید؟ سه مدل رفتار آزاد $G_6$ تا $G_8$ به این سه مجموعه توانایی اشاره دارند.




**شکل ۵-۲: مدل‌های رفتار آزاد پیشنهادی برای وظیفه کاوش هدف.**

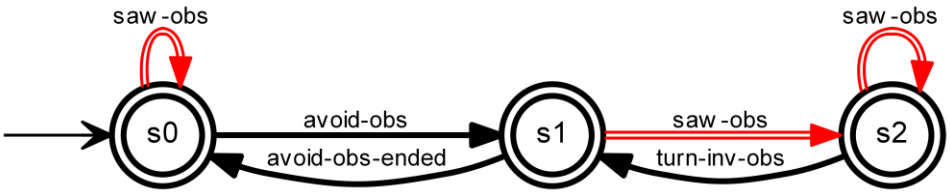
در یک طراحی درست، هر مدل به صورت نسبی، قابلیت‌های یک حسگر یا محرک را توصیف می‌کند. مدل‌ها باید تا اندازه‌ی ممکن پایه‌ای و ساده باشند [۴۹]. در طراحی پیشنهادی، مدل $G_1$ به حسگر نور، مدل $G_2$ به حسگر مجاورت‌سنج، مدل $G_3$ به محرک موتورها، مدل $G_4$ به حسگر زمین، مدل $G_5$ به حسگر دوربین ۳۶۰ درجه و مدل‌های $G_6$، $G_7$ و $G_8$ به سه مجموعه متفاوت از قابلیت‌های محرک دستگیره اشاره دارند.

## ۵-۴- طراحی ضابطه‌های کنترلی

همان‌طور که در بخش‌های قبل به این موضوع اشاره شد، یکی از مزایای طراحی توسط SCT قابلیت استفاده مجدد می‌باشد. در این بخش هر ضابطه‌ی کنترلی به صورت یک ماژول طراحی شده است تا به صورت مستقل عمل کرده و قابلیت استفاده مجدد در سایر پروژه‌ها را نیز داشته باشد. در پایان این بخش، یک مدل اصلی پیشنهاد شده است که از همه ماژول‌ها استفاده کرده و ضابطه‌ی کنترلی اصلی مسئله کاوش هدف را ارائه می‌کند. البته هر ماژول می‌تواند بسته به نیازش، از سایر ماژول‌ها استفاده کند؛ برای مثال، ماژول دوری از موانع تقریبا توسط همه ماژول‌های دیگر مورد استفاده قرار گرفته است. هر ماژول توسط یک رخداد قابل کنترل آغاز شده و توسط یک رخداد قابل کنترل به پایان می‌رسد. در واقع نقطه ورود و خروج هر ماژول مشخص است که امکان استفاده از آن ماژول را به سادگی فراهم می‌آورد. در ادامه به تشریح هر یک از ماژول‌ها پرداخته شده است. در ادامه این فصل، دو واژه ضابطه‌ی کنترلی و ماژول به جای یکدیگر مورد استفاده قرار گرفته‌اند.

### ۵-۴-۱- ماژول دوری از موانع

ماژول دوری از موانع در قالب ضابطه‌ی کنترلی $E_1$ ارائه شده که در شکل ۵-۳ به نمایش گذاشته شده است. این ماژول توسط رخداد $avoid-obs$ آغاز شده و توسط رخداد $avoid-obs-ended$ به پایان می‌رسد. هر جا که نیاز به استفاده از این ماژول وجود داشته باشد، رخداد قابل کنترل $avoid-obs$ فراخوانی می‌شود؛ سپس تا رویدادن رخداد $avoid-obs-ended$ انتظار صورت می‌گیرد که در واقع نشانه‌ی پایان اجرای ماژول است.



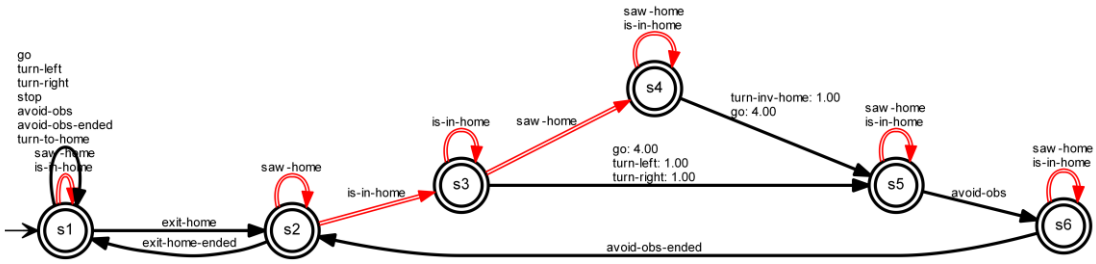

**شکل ۵-۳: ضابطه‌ی کنترلی پیشنهادی $E_1$ برای دوری از موانع در وظیفه کاوش هدف.**

این ضابطه‌ی کنترلی تا زمان فراخوانی رخداد $avoid-obs$، همه رخدادهای $saw-obs$ را نادیده می‌گیرد؛ یعنی با دیدن موانع هیچ عکس‌العملی نشان نمی‌دهد. پس از فراخوانی رخداد $avoid-obs$، تا زمانی که مانعی دیده (حس) شود، در جهت عکس مسیر دیده شدن مانع، چرخش صورت می‌گیرد. این چرخش تا زمانی ادامه پیدا می‌کند که مانعی احساس نشود. موانع توسط دو حسگر مجاورت‌سنجی که در جلوی ربات قرا دادند، حس می‌شوند.

### ۵-۴-۲- ماژول خروج از خانه

ماژول خروج از خانه که در شکل ۵-۴ نمایش داده شده است، توسط رخداد $exit - home$ آغاز و توسط رخداد $exit - home - ended$ به پایان می‌رسد. پس از شروع کار ماژول، در صورتی که ربات در خانه باشد $(is - in - home)$ و خانه را ببیند $(saw - home)$، به احتمال ۲۰ درصد در جهت عکس خانه چرخیده و به احتمال ۸۰ درصد به مسیر قبلی خود ادامه می‌دهد. در صورتی که خانه را نبیند، به احتمال ۱۰ درصد به سمت چپ و به احتمال ۱۰ درصد به سمت راست می‌چرخد؛ و یا به احتمال ۸۰ درصد به مسیر قبلی خود ادامه می‌دهد. مقادیر درنظر گرفته شده برای احتمالات در این ماژول و در سایر ماژول‌ها به صورت تجربی انتخاب شده‌اند. این چرخش‌های احتمالاتی به ربات اجازه می‌دهد تا به صورت تدریجی از خانه خارج شود؛ علاوه‌بر این همه ربات‌ها با زاویه یکسان نسبت به خانه از خانه خارج نمی‌شود؛ این امر منجر به جستجوی بهتر فضای خارج از خانه می‌شود.

پس از هر گام که منجر به چرخش یا حرکت مستقیم می‌شود، یک مرتبه ماژول دوری از موانع فراخوانی می‌شود تا در صورتی که پس از گام طی شده، مانع جدیدی در مسیر ربات ظاهر شود، ربات بتواند از برخورد با آن اجتناب کند.

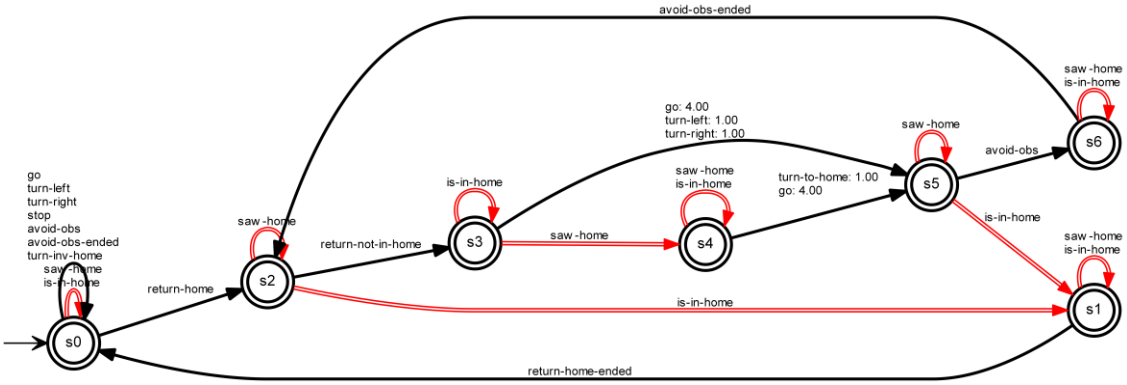



**شکل ۵-۴: ضابطه‌ی کنترلی پیشنهادی $E_2$ برای خروج از خانه در وظیفه کاوش هدف.**

در این ماژول و سایر ماژول‌ها دیده می‌شود که در برخی حالات، رخدادهایی به صورت self-loop مورد استفاده قرار گرفته‌اند. استفاده از رخدادهای غیرقابل کنترل به صورت self-loop برای جلوگیری از ایجاد حالت بد در سوپروایزر است؛ به بیانی دیگر، ضابطه‌ی کنترلی مربوطه، به این روش، رویدادن یک رخداد غیرقابل کنترل را مجاز اعلام می‌دارد. و اما در رابطه با رخدادهای قابل کنترل، این امر معمولا در حالت ابتدایی ماژول رخ می‌دهد؛ جایی که به دلیل عدم فراخوانی رخداد آغازین ماژول (برای مثال رخداد $exit - home$ در شکل ۵-۴)، ماژول آغاز به کار نکرده است؛ بنابراین به این وسیله و با قرار دادن مجموعه رخدادهای قابل کنترل مجاز به صورت self-loop در حالت ابتدایی خود، رویدادن آن‌ها در سایر ماژول‌ها را مجاز اعلام می‌کند. این کار تنها در روش سنتز ماژولار معنی دارد؛ به بیانی دیگر، رخداد مورد استفاده باید حداقل بین دو ماژول مشترک باشد.

### ۵-۴-۳- ماژول بازگشت به خانه

ماژول بازگشت به خانه در شکل ۵-۵ به نمایش گذاشته شده است. این ماژول به ترتیب با دو رخداد $return - home$ و $return - home - ended$ آغاز شده و به پایان رسیده است. در طراحی ضابطه‌های کنترلی، نوع جدیدی از رخدادها به نام رخدادهای سایه پیشنهاد شده‌اند. از آنجا که در لایه اجرا کننده مولد، رخدادهای غیرقابل کنترل اولویت بالاتری نسبت به سایر رخدادها دارند؛ در صورتی که از یک حالت مشخص، دو گذار خروجی وجود داشته باشد؛ که اولی دارای یک رخداد غیرقابل کنترل و دومی دارای یک رخداد قابل کنترل باشد؛ اولویت اجرا با رخداد غیرقابل کنترل است. بنابراین رخداد قابل کنترل تنها در صورتی اجرا خواهد شد که رخداد غیرقابل کنترل فعال نباشد. برای چنین رخدادهایی که در زیر سایه‌ی یک رخداد غیرقابل کنترل عمل می‌کنند، نام «رخداد سایه» را برگزیده‌ایم. یک کاربرد رخدادهای سایه، بررسی عدم فعال بودن یک رخداد غیرقابل کنترل است. برای مثال در حالت $s_2$ از ضابطه‌ی کنترلی $E_3$، قصد بررسی عدم وجود ربات در خانه وجود داشته است؛ از طرفی تنها یک رخداد غیرقابل کنترل برای بررسی وجود ربات در خانه تعریف شده است ($is - in - home$) و رخدادی برای بررسی عدم وجود ربات در خانه وجود ندارد. بنابراین از رخداد سایه ($return - not - in - home$) استفاده شده تا در صورت فعال نبودن رخداد $is - in - home$، بتوان مسیری جدید در مولد تعریف کرد.




**شکل ۵-۵: ضابطه‌ی کنترلی پیشنهادی $E_3$ برای بازگشت به خانه در وظیفه کاوش هدف.**

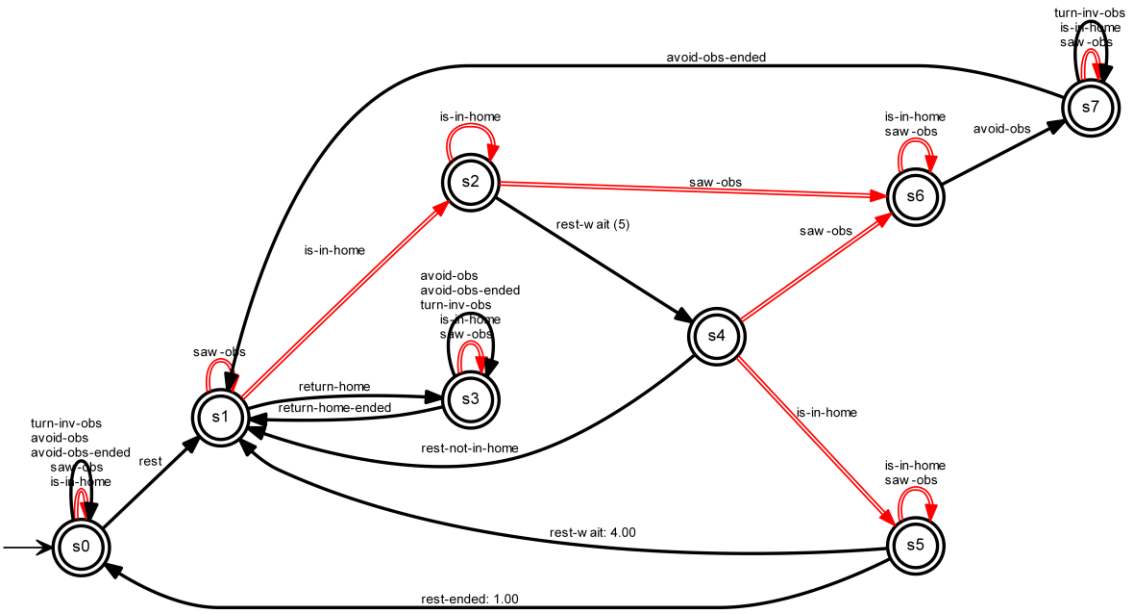
در این ضابطه‌ی کنترلی، ربات تنها در صورتی به سمت خانه می‌چرخد که اولا) در حال حاضر در خانه نباشد و دوما) خانه را ببیند؛ در غیراینصورت به صورت تصادفی یکی از سه حرکت ۱) به مسیر فعلی ادامه بده، ۲) به سمت راست بچرخ و ۳) به سمت چپ بچرخ را انتخاب می‌کند. در همه این حالات، احتمال انتخاب رخداد به مسیر فعلی ادامه بده، بیشتر از سایر رخدادها است. مانند ماژول خروج از خانه، در پایان هر گام، یک مرتبه ماژول دوری از موانع فراخوانی می‌شود.

### ۵-۴-۴- ماژول استراحت

ماژول استراحت به ترتیب با رخدادهای $rest$ و $rest - ended$ آغاز شده و به پایان می‌رسد (شکل ۵-۶). این ماژول پس از آغاز به کار، ابتدا وجود ربات در خانه ($is - in - home$) را بررسی می‌کند؛ در صورت وجود ربات در خانه، ۵ تیک ساعت در خانه حرکت کرده و سپس به صورت احتمالاتی رخداد $rest - ended$ را اجرا می‌کند. این روش عملکرد به این دلیل انتخاب شده است که همه ربات‌ها به محض رسیدن به مرز خانه، توقف نکرده و به استراحت نپردازند. به این ترتیب ربات‌ها در محدوده خانه حرکت کرده و احتمالا در مکانی خلوت‌تر به استراحت می‌پردازند. علاوه‌بر این، ترجیحا مرزهای خانه باید خلوت باشند تا رفت و آمد ربات‌هایی که غذا به همراه دارند با راحتی بیشتر صورت گیرد.

ماژولی که ماژول استراحت را فراخوانی کرده باید تصمیم بگیرد، ربات به استراحت بپردازد یا به جستجو ادامه دهد. این ماژول می‌تواند کاربردهای دیگری جز استراحت نیز داشته باشد؛ برای مثال، برای حمل هدف به خانه نیز می‌توان از همین ماژول استفاده کرد؛ ولی پس از وارد شدن ربات به خانه، بجای استراحت، به رهاسازی هدف در خانه پرداخته شود.

در صورتی که ربات در حال حاضر در خانه نباشد، ماژول بازگشت به خانه فراخوانی می‌شود؛ پس از پایان ماژول بازگشت به خانه، مراحل توضیح داده شده در بالا تکرار می‌شوند. در ماژول استراحت از دو ماژول «دوری از موانع» و «بازگشت به خانه» بهره برده شده است؛ عدم وجود ماژول‌ها قطعا منجر به بزرگ‌تر شدن ضابطه‌ی کنترلی جاری می‌شد.

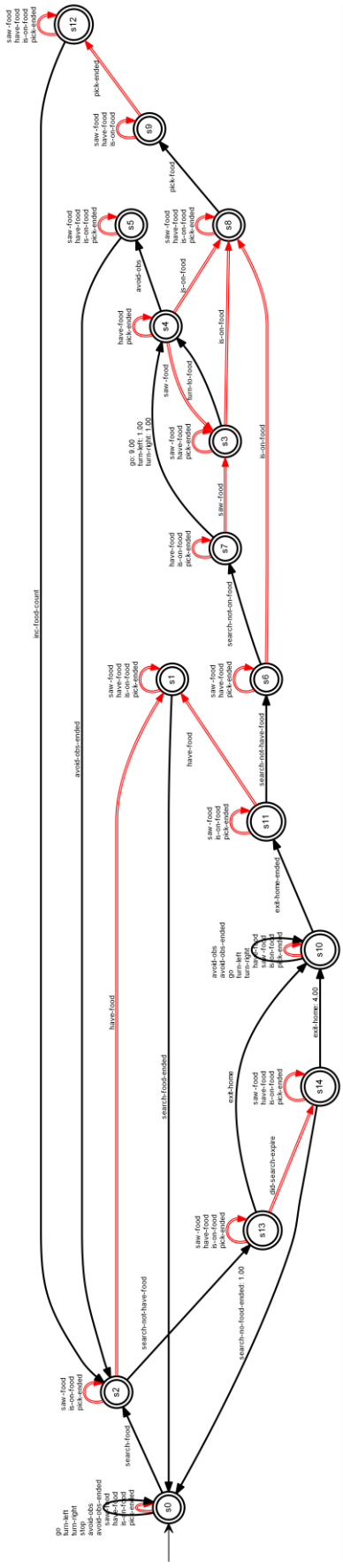



**شکل ۵-۶: ضابطه‌ی کنترلی پیشنهادی $E_4$ برای استراحت در وظیفه کاوش هدف.**

در ضابطه‌ی کنترلی جاری هم از رخدادهای احتمالاتی ($rest - ended$) استفاده شده است و هم از رخدادهای زمان‌دار ($rest - wait$). رخدادهای $rest - wait$ و $rest - not - in - home$ نیز دو نمونه‌ی دیگر از رخدادهای سایه هستند که در این ماژول مورد استفاده قرار گرفته‌اند.

### ۵-۴-۵- ماژول جستجو

ماژول جستجو پیچیده‌ترین ماژول پیشنهادی است که در شکل ۵-۷ به نمایش گذاشته شده است. این ماژول با فراخوانی رخداد $search - food$ آغاز به کار کرده و با رخداد $search - food - ended$ به کار خود پایان می‌دهد.




**شکل ۵-۷: ضابطه‌ی کنترلی پیشنهادی $E_5$ برای جستجوی هدف در وظیفه کاوش هدف.**

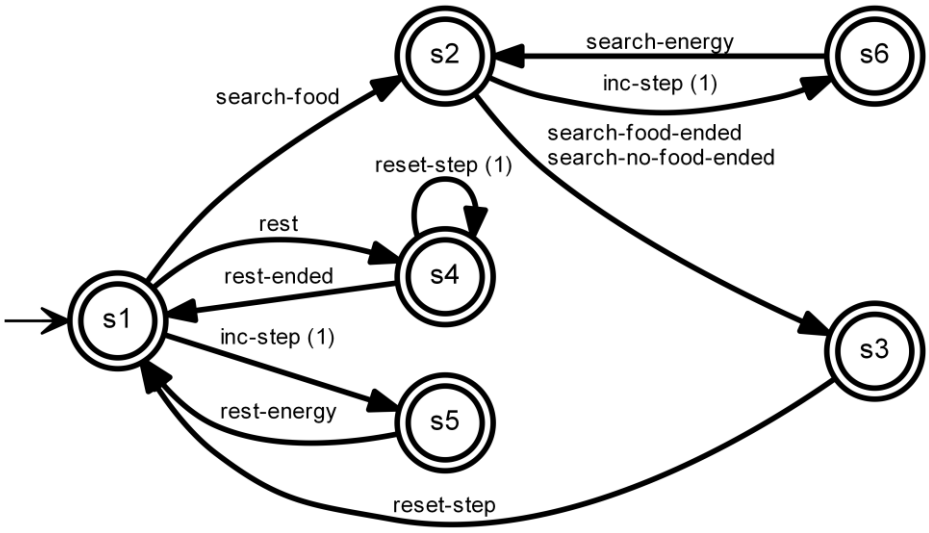
در ابتدای اجرای ماژول، بررسی می‌شود که آیا ربات در حال حمل غذا است؛ در صورتی که اینچنین باشد، ربات به کار جستجو پایان می‌دهد. در غیراینصورت، رخداد سایه $search - not - have - food$ اجرا می‌شود؛ سپس با فراخوانی ماژول خروج از خانه، از خارج از خانه بودن اطمینان حاصل می‌شود. آنگاه تا زمانی که یکی از دو شرط زیر برقرار باشد، ربات به صورت تصادفی به حرکت در محیط می‌پردازد. در حرکت تصادفی به احتمال نه به یازده، ربات به مسیر فعلی خود ادامه داده و به احتمال یک به یازده به سمت راست یا چپ می‌چرخد.

۱. یافتن هدف $(saw - food)$
۲. طی کردن تعداد مشخصی گام $(did - search - expire)$

در صورت یافتن هدف، ربات اقدام به برداشتن هدف مربوطه می‌کند و سپس یکی به تعداد اهداف حمل شده توسط خود اضافه می‌کند $(inc - food - count)$. در صورت طی شدن تعداد مشخصی از گام‌ها و نیافتن هدف، ربات به جستجو پایان می‌دهد ولی نه با رخداد پایانی معمول خود $(search - food - ended)$، بلکه با رخداد $search - no - food - ended$ که به معنی پایان جستجو به دلیل طی کردن حداکثر تعداد گام‌های مجاز است. با استفاده از این رخداد می‌توان از ربات خواست که به استراحت بپردازد و یا هر عمل دلخواه دیگری توسط طراح درنظر گرفته شود. ماژول جستجو از ماژول‌های «دوری از موانع» و «خروج از خانه» بهره می‌برد.

### ۵-۴-۶- ضابطه‌ی کنترلی محاسبه تعداد گام‌ها و انرژی

محاسبه تعداد گام‌ها و انرژی مصرف شده توسط ربات‌ها معمولا به صورت خاص‌منظوره در پروژه‌ها پیاده‌سازی می‌شود [۱۲۲, ۱۲۵]؛ در واقع طراح، مستقل از روش پیاده‌سازی سیستم پایه، برای محاسبه انرژی توسط ربات‌ها برنامه‌نویسی انجام می‌دهد. این امر هرچند قابل قبول است ولی با مزایایی که در طراحی فرمال سیستم نامبرده شد، در تعارض است. از این رو، در این بخش مدلی ارائه دادیم که به موازات سایر ضابطه‌های کنترلی اجرا شده و مقدار انرژی مصرفی و تعداد گام‌های طی شده توسط هر ربات را شمارش کند. مدل پیشنهادی در شکل ۵-۸ به نمایش گذاشته شده است. از آنجا که این مدل برای محاسبه یک مجموعه از پارامترهای نرم‌افزاری ارائه گشته است، طبیعتا تنها از رخدادهای قابل کنترل بهره برده که هیچ یک در مدل‌های رفتار آزاد تعریف نشده‌اند.




**شکل ۵-۸: ضابطه‌ی کنترلی پیشنهادی $E_6$ برای محاسبه تعداد گام‌ها و انرژی در وظیفه کاوش هدف.**

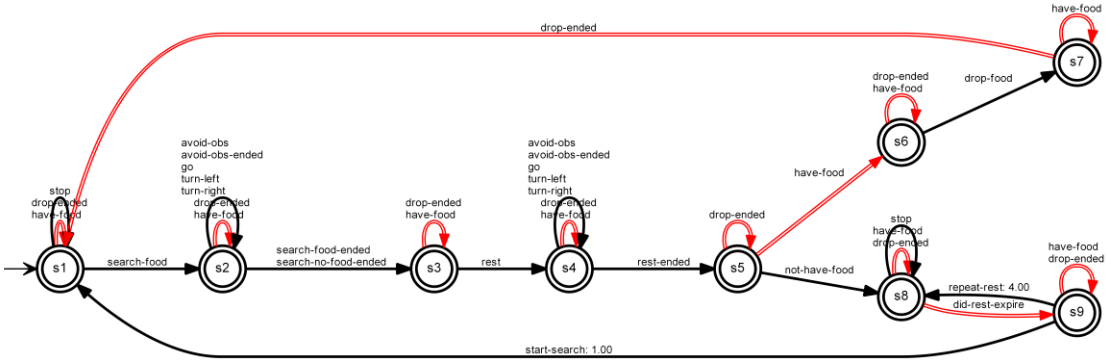
برای محاسبه انرژی مصرفی کل برای هر یک از اعمال زیر، مقدار انرژی مصرفی متفاوتی پیشنهاد شده است:

- به ازای هر هدف پیدا شده و به خانه آورده شده، مقدار انرژی مثبت $E_f$ به انرژی کل ربات افزوده می‌شود.
- طی کردن هر گام (تیک ساعت سراسری سیستم) در فاز جستجو، مقدار انرژی منفی $\mathrm{E_s}$ را از انرژی کل ربات می‌کاهد.
- هر واحد زمانی که ربات در فاز استراحت سپری می‌کند، مقدار انرژی منفی $E_r$ را از انرژی کل ربات کم می‌کند.

این مدل به سادگی قابل گسترش است و می‌توان پارامترهای دیگری به آن افزود. در طراحی ارائه شده در شکل ۵-۸ با اجرا شدن رخداد $search - food$، شمارش گام‌ها آغاز شده و با هر تیک ساعت سراسری سیستم، انرژی مصرفی کل به‌روزرسانی می‌شود. با اجرا شدن رخداد $rest$، ابتدا تعداد گام‌های قبلی به صفر تغییر داده شده و شمارش مجددا آغاز می‌شود و سپس انرژی مصرفی در فاز استراحت به انرژی مصرفی کل افزوده می‌شود.

### ۵-۴-۷- ضابطه کنترلی مدیریت چرخه‌ی اصلی کاوش هدف

در بخش‌های قبل به ارائه‌ی ماژول‌های مختلف برای پیاده‌سازی زیربخش‌های موردنیاز از وظیفه کاوش هدف پرداخته شد. در این بخش، یک ضابطه‌ی کنترلی پیشنهاد شده است که اتصال و ارتباط بین ماژول‌ها را برقرار می‌کند. ضابطه‌ی کنترلی پیشنهادی در شکل ۵-۹ نمایش داده شده است.




**شکل ۵-۹: ضابطه‌ی کنترلی پیشنهادی $E_7$ برای مدیریت چرخه‌ی اصلی در وظیفه کاوش هدف.**

ضابطه‌ی کنترلی پیشنهادی در ابتدا ماژول جستجو $(search - food)$ را فراخوانی می‌کند. سپس منتظر می‌ماند تا ماژول جستجو با یکی از شرایط زیر به کار خود پایان دهد:

۱. یافتن هدف و اتمام جستجو $(search - food - ended)$
۲. طی شدن سقف تعداد گام‌های مجاز و نیافتن هدف $(search - no - food - ended)$

پس از اتمام جستجو، ماژول استراحت $(rest)$ فراخوانی می‌شود تا ربات به خانه بازگردد. در این حالت چه ربات در حال حمل هدف باشد و چه بدون هدف، به خانه بازگردانده می‌شود. پس از رسیدن به خانه $(rest - ended)$ و پیدا کردن یک مکان مناسب در محوطه خانه، بر اساس به همراه داشتن یا نداشتن هدف $(have - food)$ تصمیم‌گیری انجام می‌شود. در صورتی که ربات، هدفی را به همراه داشته باشد، باید آن را در خانه رها کرده $(drop - food)$ و مجددا به جستجو بپردازد. ولی اگر بدون همراه داشتن هدف به خانه بازگشته باشد، هدفی جز استراحت نداشته است؛ بررسی به همراه نداشتن غذا با استفاده از رخداد سایه $not - have - food$ صورت می‌گیرد. برای استراحت ربات، رخداد $stop$ اجرا شده و به صورت تکراری بررسی می‌شود که تعداد تیک‌های مجاز ساعت سراری برای استراحت به پایان رسیده است یا خیر $(did - rest - expire)$. برای پایان دادن به استراحت (start-search) از تصمیم‌گیری احتمالاتی استفاده شده تا ربات‌هایی که به صورت همزمان، استراحت را آغاز کرده‌اند، به صورت همزمان تصمیم به آغاز به جستجو نگیرند. به این ترتیب از ترافیک احتمالی این اتفاق جلوگیری به عمل می‌آید. برای این امر، هر ربات به احتمال ۸۰٪ به استراحت خود ادامه می‌دهد و به احتمال ۲۰٪ جستجو را آغاز می‌کند. این تصمیم به صورت چرخشی تکرار می‌شود تا زمانی که جستجو آغاز شود.

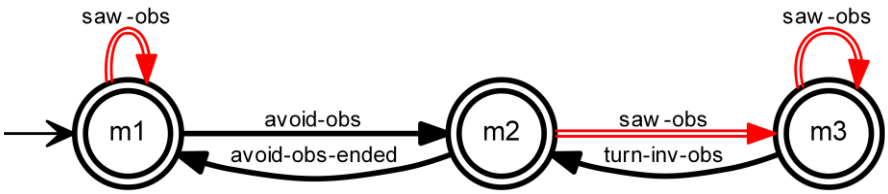
## ۵-۵- محاسبه سوپروایزرهای ماژولار محلی

ابزار جامع پیشنهادی برای محاسبه سوپروایزرهای ماژولار محلی، ابتدا عبارت‌های $G_i^{loc}\ \ i \in \{1, \dots, 7\}$ را به صورت ارائه شده در فرمول (۲۳) محاسبه می‌کند؛ سپس با استفاده از فرمول‌های (۱۵) و (۱۶)،

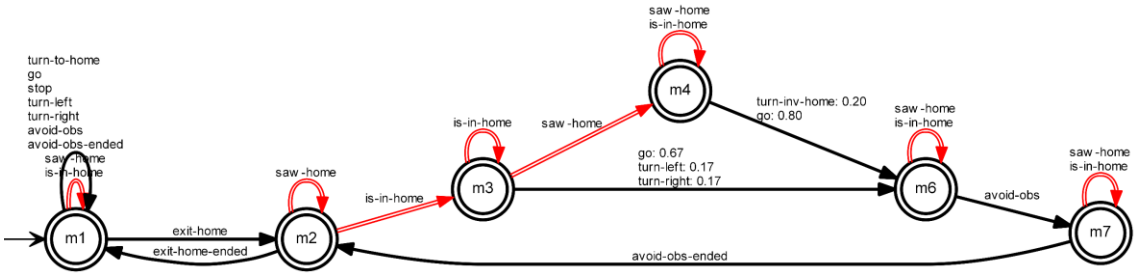
عبارات $K_i^{loc}, S_i^{loc}\ \ i \in \{1, \dots, 7\}$ را بدست می‌آورد. هفت سوپروایزر ماژولار محلی محاسبه شده در شکل ۵-۱۰ تا شکل ۵-۱۶ نمایش داده شده‌اند. سوپروایزرهای $S_1^{loc}$ و $S_6^{loc}$ با ضابطه‌های کنترلی متناظر خود برابر هستند و بدون هیچ تغییری مورد استفاده قرار می‌گیرند.

$$G_1^{loc} = G_2,\ G_2^{loc} = G_1 \parallel G_3 \parallel G_4,\quad G_3^{loc} = G_1 \parallel G_3 \parallel G_4,\quad G_4^{loc} = G_2 \parallel G_4, \qquad (۲۳)$$

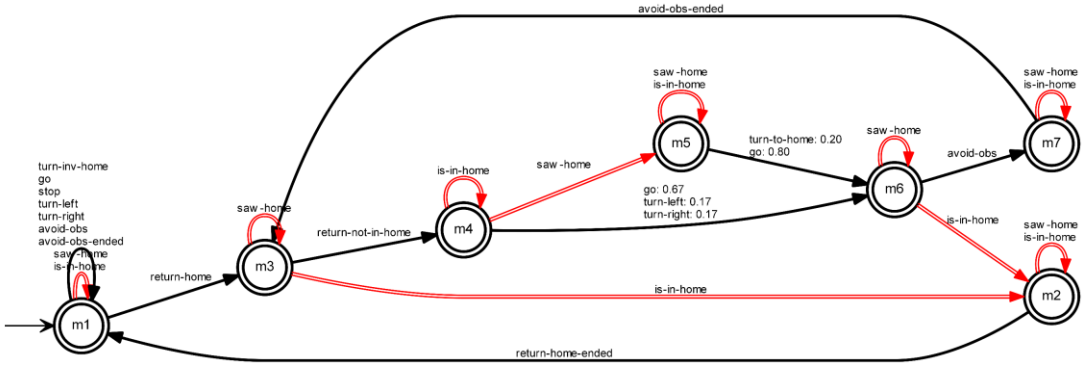
$$G_5^{loc} = G_3 \parallel G_5 \parallel G_6 \parallel G_7,\ G_6^{loc} = \emptyset,\ G_7^{loc} = G_3 \parallel G_6 \parallel G_8$$



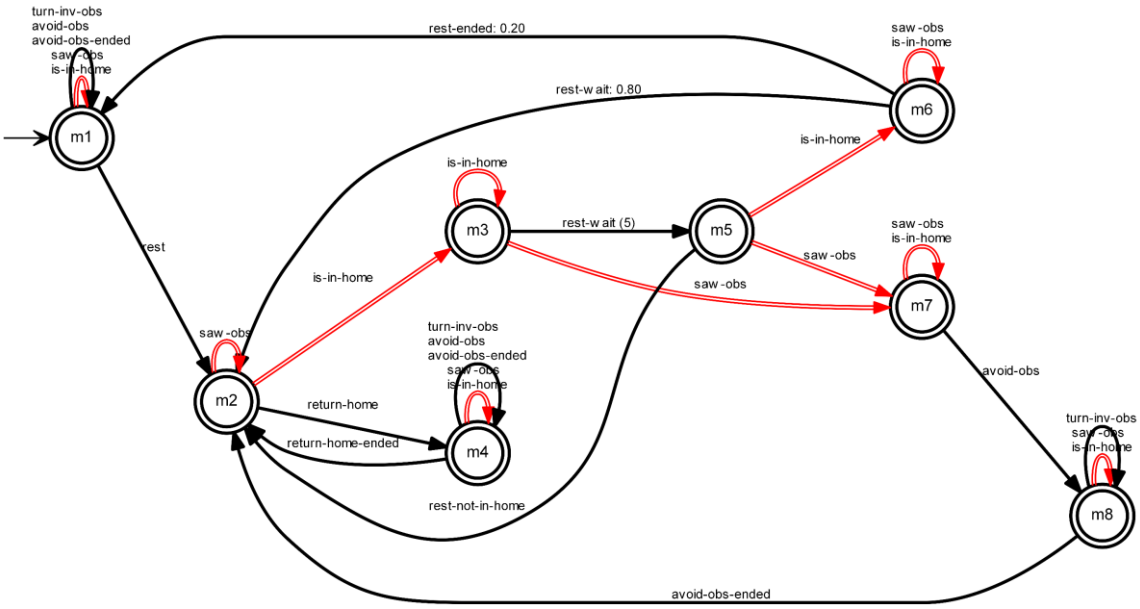

**شکل ۵-۱۰: سوپروایزر ماژولار محلی $S_1^{loc}$ برای دوری از موانع در وظیفه کاوش هدف.**



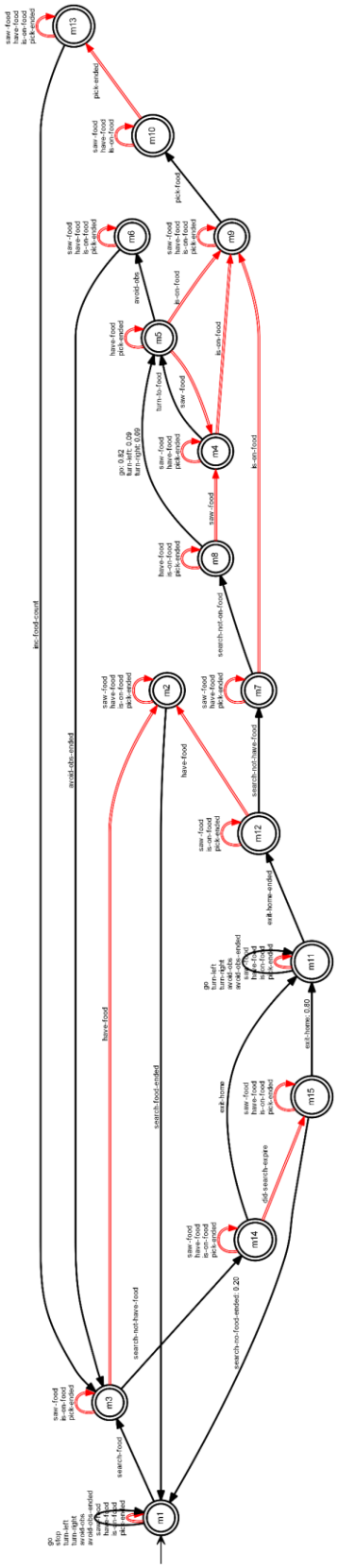

**شکل ۵-۱۱: سوپروایزر ماژولار محلی $S_2^{loc}$ برای خروج از خانه در وظیفه کاوش هدف.**



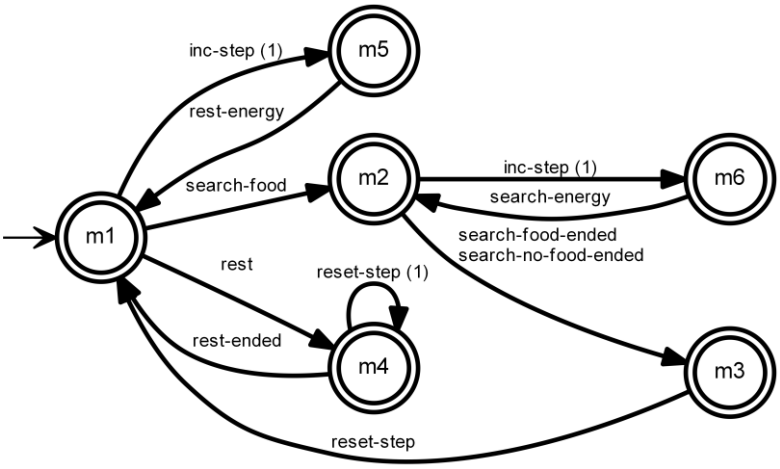

**شکل ۵-۱۲: سوپروایزر ماژولار محلی $S_3^{loc}$ برای بازگشت به خانه در وظیفه کاوش هدف.**



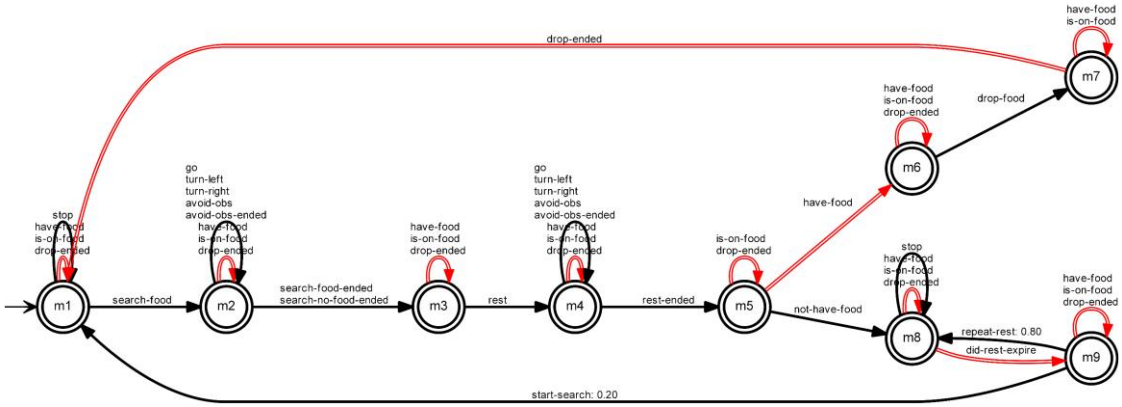

**شکل ۵-۱۳: سوپروایزر ماژولار محلی $S_4^{loc}$ برای استراحت در وظیفه کاوش هدف.**




**شکل ۵-۱۴: سوپروایزر ماژولار محلی $S_5^{loc}$ برای جستجوی غذا در وظیفه کاوش هدف.**




**شکل ۵-۱۵: سوپروایزر ماژولار محلی $S_6^{loc}$ برای محاسبه انرژی و شمارش گام در وظیفه کاوش هدف.**




**شکل ۵-۱۶: سوپروایزر ماژولار محلی $S_7^{loc}$ برای ترکیب همه ماژول‌ها در وظیفه کاوش هدف.**

با اعتبارسنجی صورت گرفته توسط ابزار پیشنهادی، حاصل ترکیب موازی سوپروایزرهای ماژولار محلی برابر است با سوپروایزر یکپارچه سراسری. بنابراین طراحی ماژولار پیشنهادی عملکرد مشابهی با طراحی یکپارچه خواهد داشت. هر چند از نظر تعداد حالت‌ها و گذارها تفاوت قابل توجهی بین این دو طراحی وجود دارد. جدول ۵-۲، تعداد حالت‌ها و گذارهای طراحی ماژولار محلی و یکپارچه را ارائه کرده است. در این جدول مشاهده می‌شود که مجموع تعداد گذارها در طراحی ماژولار محلی تقریبا یک سوم تعداد گذارها در طراحی یکپارچه می‌باشد.

**جدول ۵-۲: مقایسه بین تعداد حالت‌ها و گذارهای طراحی یکپارچه و ماژولار محلی.**

| روش سنتز | تعداد سوپروایزرها | تعداد حالت‌ها | تعداد گذارها |
|---|---|---|---|
| یکپارچه | ۱ | ۶۹ | ۷۲۶ |
| ماژولار محلی | ۷ | ۵۴ | ۲۴۶ |

## ۵-۶- جمع‌بندی

با توجه به معرفی ابزار جامع پیشنهادی در فصل پیشین، در این فصل به ارائه یک طراحی معتبر برای پیاده‌سازی وظیفه کاوش هدف با استفاده از تئوری کنترل سوپروایزری احتمالاتی و زمان‌دار پرداخته شد. تمامی بخش‌های طراحی شامل تحلیل نیازهای مسئله، ارائه مدل‌های رفتار آزاد و ضابطه‌های کنترلی پوشش داده شد. سوپروایزرهای ماژولار محلی محاسبه شده نیز در پایان فصل ارائه گشت و تعداد حالت‌ها و گذارهای دو روش سنتز یکپارچه و ماژولار محلی مورد مقایسه قرار گرفت. حال همه چیز برای اجرای طراحی صورت گرفته در محیط شبیه‌سازی فراهم است؛ زیرا ابزار جامع پیشنهادی، کد موردنیاز برای اجرای وظیفه کاوش هدف در سکوی ARGoS را به صورت خودکار تولید کرده است. تنها بخشی که باید به کد تولید شده اضافه شود، توابع فراخوان مربوط به هر یک از رخدادها می‌باشد.

# فصل

# ۶- ارزیابی و نتایج آزمایش‌ها

## ۶-۱- مقدمه

در دو فصل قبل، ابزار جامع پیشنهادی معرفی گردید؛ سپس توسط ابزار معرفی شده، به پیاده‌سازی وظیفه کاوش هدف با استفاده از تئوری کنترل سوپروایزری احتمالاتی و زمان‌دار پرداخته شد. در این بخش به آزمایش همه جوانب پیاده‌سازی صورت گرفته خواهیم پرداخت. طراحی پیشنهادی در مجموع ۴۸۰ مرتبه با تنظیمات متفاوت در محیط شبیه‌سازی ARGoS اجرا شده است. اهداف آزمایش‌های صورت گرفته در ادامه فهرست شده‌اند:

- بررسی عملکرد سیستم در برابر اعمال مقادیر متفاوت نویز
- بررسی تاثیر افزایش تعداد ربات‌ها بر عملکرد سیستم
- بررسی عملکرد سیستم در حالتی که ربات‌ها قادر به استراحت نیز هستند
- بررسی تاثیر تعداد اهداف و موانع بر عملکرد سیستم
- بررسی مقدار انرژی مصرفی در واحد زمان در حضور عامل استراحت و بدون حضور آن
- بررسی عملکرد سیستم در زمان ثابت

جدول ۶-۱ پارامترهای قابل تنظیم در هر آزمایش و مقادیر ممکن بر هریک را نمایش داده است. همه ترکیبات ممکن پارامترهای ارائه شده، آزمایش شده‌اند که برابر است با $4 \times 4 \times 5 \times 3 \times 2 = 480$. هر آزمایش پنج مرتبه تکرار شده و نمونه‌ای که دارای زمان مجموع اجرای میانه[۱] بوده، به عنوان نماینده انتخاب گشته است. مقدار نویز مورد اشاره به صورت نویز یکنواخت تجمعی[۲] به خروجی حسگرها اعمال می‌گردد.

**جدول ۶-۱: پارامترهای قابل تنظیم در آزمایش‌های انجام شده.**

| پارامتر | توضیح | مجموعه مقادیر |
|---|---|---|
| $N_o$ | تعداد موانع | $[0, 5, 10, 20]$ |
| $N_t$ | تعداد اهداف | $[10, 20, 40, 80]$ |
| $N_r$ | تعداد ربات‌ها | $[10, 20, 40, 70, 100]$ |
| $E$ | مقدار نویز | $[0, 0.02, 0.04]$ |
| $RE$ | امکان استراحت | [بلی، خیر] |

[۱] Median

[۲] Additive uniform noise

در برخی آزمایش‌ها، امکان استراحت ربات‌ها غیرفعال شده است؛ در این آزمایش‌ها، تابع فراخوان مربوط به رخداد $did - search - expire$ (ضابطه‌ی کنترلی $E_5$ در شکل ۵-۷) همیشه مقدار false بازمی‌گرداند؛ بنابراین ربات هیچگاه به فاز استراحت تغییر حالت نمی‌دهد. ربات‌ها بلافاصله پس از رهاسازی اهداف در خانه، به جستجو در محیط می‌پردازند. برای آزمایش‌هایی که امکان استراحت ربات وجود دارد، رخداد $did - search - expire$ پس از طی شدن ۴۰۰۰ گام، مقدار true بازمی‌گرداند. تعداد گام‌ها (تیک‌های ساعت سراسری) برای رخداد $did - rest - expire$ برابر با ۱۰۰۰ عدد است. ابعاد محیط مربع‌شکل مورد استفاده در آزمایش‌ها $10 \times 10m$ می‌باشد؛ یک ربات به صورت تقریبی با برداشتن ۲۰۰۰ گام، طول محیط را طی می‌کند. انتخاب مقدار ۴۰۰۰ گام برای استراحت ربات با این منطق صورت گرفته که ربات یک مرتبه طول محیط را طی کرده و بازگشته و چیزی پیدا نکرده است. هرچند ربات‌ها بلافاصله پس از این تعداد گام به استراحت نمی‌پردازند؛ زیرا همان‌طور که در بخش مربوطه اشاره شد، تغییر حالت از جستجو به استراحت و برعکس به صورت احتمالاتی صورت می‌گیرد. تعداد گام‌ها و انرژی رابطه خطی نسبت به هم دارند. بنابراین می‌توان از المان انرژی بجای تعداد گام‌ها بهره برد. طراحی پیشنهادی در بخش ۵-۴-۶- رابطه‌ی بین تعداد گام‌ها و انرژی مصرفی را تعریف کرده است و استفاده از هر یک از این دو المان، تغییری در رویکرد پیاده شده برای استراحت نخواهد داشت.

محیط طراحی شده برای آزمایش‌ها در چهار طرف با دیوار محصور شده است. خانه در ضلع جنوبی محیط قرار گرفته است؛ هرچند مکان خانه تاثیری بر پیاده‌سازی پیشنهادی نمی‌گذارد. عرض خانه سه متر در نظر گرفته شده و طول آن برابر با طول ضلع محیط است. تعدادی ال‌ای‌دی (چهار عدد) در بالای خانه قرار گرفته تا ربات‌ها بتوانند با استفاده از حسگر نور، جهت خانه را تشخیص دهند. با این حال، ابعاد خانه به شکلی انتخاب گشته که ربات‌ها در انتهای ضلع شمالی خانه قادر به دیدن (احساس) خانه نباشند؛ آن‌ها ناچارند به صورت تصادفی حرکت کرده تا جایی که بتوانند خانه را ببینند.

بر روی هر هدف، یک ال‌ای‌دی قرار گرفته تا ربات‌ها بتوانند با استفاده از دوربین ۳۶۰ درجه، اهداف را ببینند. برای شبیه‌سازی محرک دستگیره، وقتی فاصله ربات تا هدف به کمتر یا مساوی هفت میلی‌متر می‌رسد، هدف از محیط شبیه‌سازی حذف و در بالای سر ربات قرار می‌گیرد. همه آزمایش‌ها در حالتی آغاز می‌شوند که همه ربات‌ها در خانه قرار دارند؛ و در حالتی به پایان می‌رسند که همه اهداف جمع‌آوری شده باشد.

سه معیار ارائه شده در جدول ۶-۲ از هر آزمایش استخراج گشته است. معیار $T_O$، معیار شناخته شده‌ای است که معمولا برای وظیفه کاوش هدف ارائه می‌گردد. با کم و زیاد شدن زمان کل کاوش هدف، می‌توان به سادگی به تاثیر یک پارامتر بر روی عملکرد ربات‌ها پی برد. معیار $T_{88\%}$ اولین بار توسط [۱۲۶] به عنوان معیاری کاربردی‌تر در مقایسه با زمان کل کاوش هدف ارائه شد. آن‌ها ادعا کردند که

پس از جمع‌آوری ۸۸٪ اهداف، معمولا یک افزایش نمایی در زمان کل کاوش هدف اتفاق می‌افتد که از پراکندگی زیاد اهداف باقیمانده در محیط ناشی می‌شود. به همین دلیل، این معیار نیز از آزمایشات استخراج گشته تا کارایی آن مورد بررسی قرار گیرد. معیار $R2S$ میانگین نسبت تعداد ربات‌های در حال استراحت به تعداد کل ربات‌ها را ارائه می‌دهد. روشن است که در یک پنجره زمانی ثابت، هر چقدر مقدار این معیار به یک نزدیک‌تر شود، نشان از عملکرد بهتر سیستم دارد.

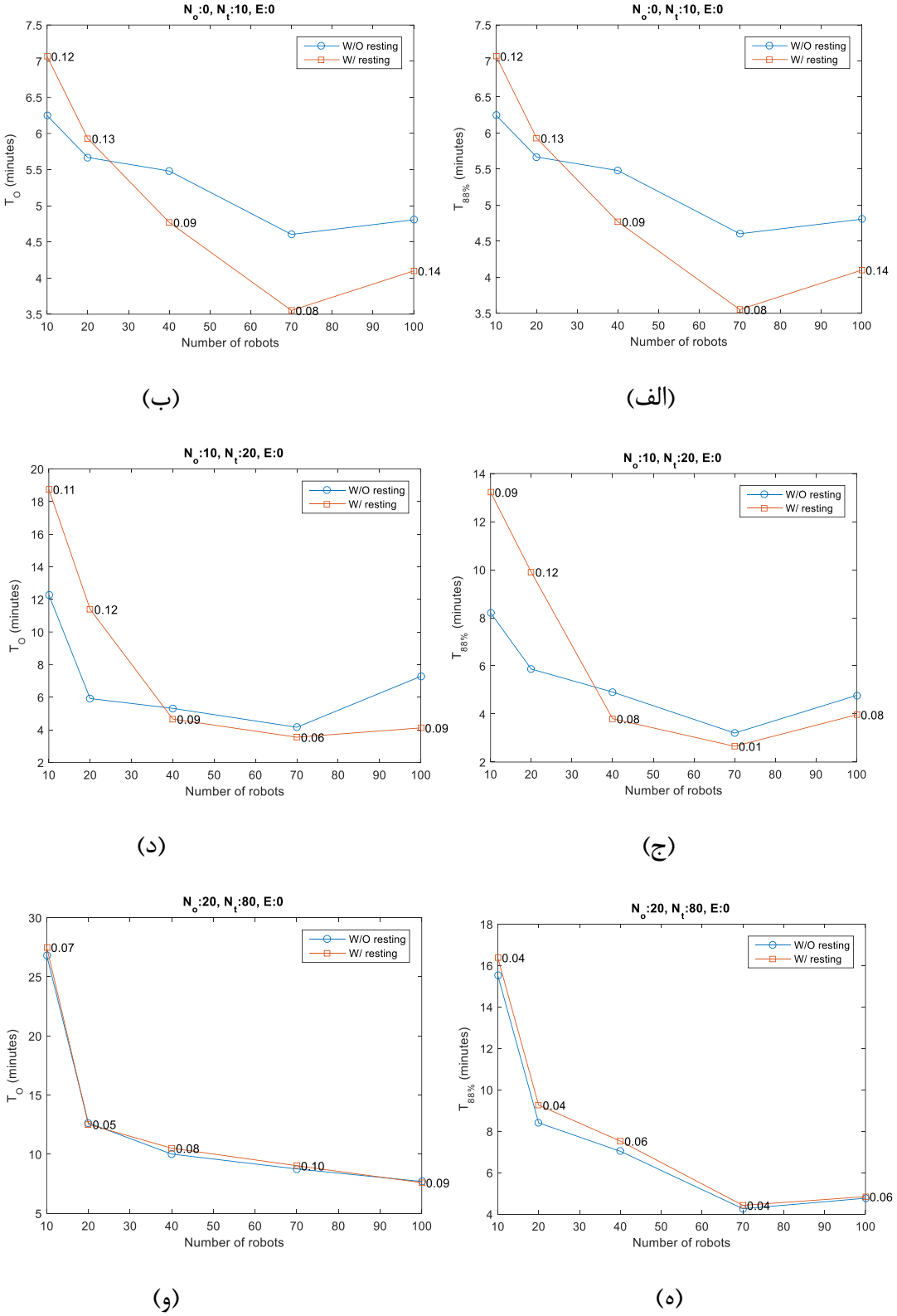
برای خوانایی و دسته‌بندی بهتر نتایج، آن‌ها را بر اساس معیارهای مورد مطالعه، دسته‌بندی کرده و هر یک را در بخشی متفاوت ارائه کرده‌ایم.

**جدول ۶-۲: معیارهای استخراج شده از همه آزمایش‌ها.**

| پارامتر | توضیح |
|---|---|
| $T_O$ | زمان کل کاوش هدف |
| $T_{88\%}$ | زمان جمع‌آوری ۸۸٪ اهداف |
| $R2S$ | نسبت تعداد ربات‌های در حال جستجو به در حال استراحت |

## ۶-۲- بررسی تاثیر استراحت بر عملکرد

در این بخش، نتایج تعدادی از آزمایش‌ها برای بررسی تاثیر استراحت بر عملکرد سیستم مورد مقایسه قرار گرفته‌اند. به ازای هر جفت آزمایش انتخاب شده، پارامترهای $N_o$، $N_t$ و $E$ ثابت درنظر گرفته شده‌اند. در هر آزمایش، یکی از دو معیار $T_O$ یا $T_{88\%}$ در محور عمودی و تعداد ربات‌ها در محور افقی قرار دارند. معیار $R2S$ طبیعتا تنها برای حالتی که امکان استراحت ربات‌ها وجود دارد، ارائه گشته است. این معیار بر روی تک‌تک نقاط خطوط مربوطه نمایش داده شده است. شکل ۶-۱ سه جفت آزمایش را به نمایش گذاشته که هر جفت، دو معیار $T_O$ یا $T_{88\%}$ را در حالت با استراحت و بدون استراحت مورد مقایسه قرار داده است.




**شکل ۶-۱: مقایسه زمان کل کاوش هدف با وجود استراحت و بدون آن (ب، د، و) و مقایسه زمان جمع‌آوری ۸۸٪ اهداف با وجود استراحت و بدون آن (الف، ج، ه) برای تنظیمات مختلف.**

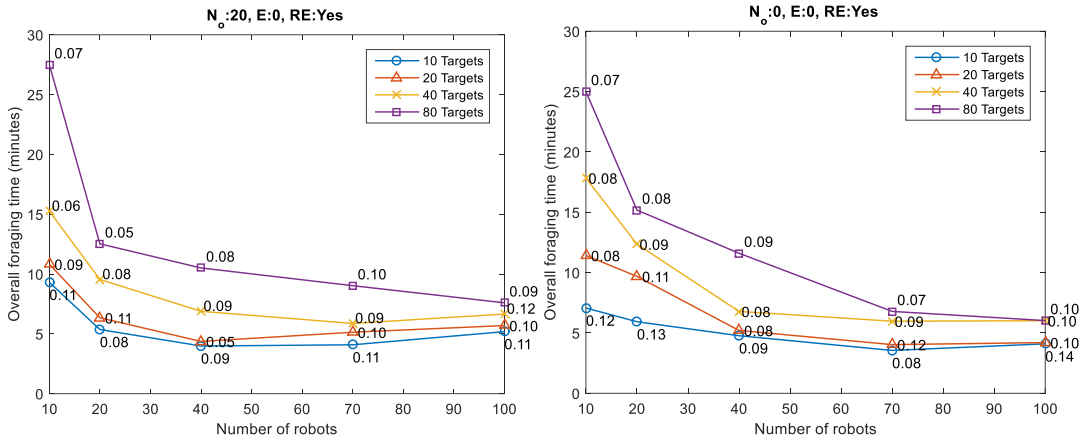
در سطر اول (الف، ب) از شکل ۶-۱، آزمایش‌ها با صفر مانع، ده هدف و بدون نویز اجرا شده‌اند. عملکرد شکل الف و ب کاملا مشابه با یکدیگر است؛ علاوه‌بر عملکرد مشابه $T_O$ و $T_{88\%}$ در این جفت تصاویر، در سایر جفت تصاویر نیز، عملکرد مشابهی دیده می‌شود. وقتی جمعیت ربات‌ها کم است، زمان کل کاوش هدف در حالت بدون استراحت کمتر از حالت با استراحت است. این امر منطقی بنظر می‌رسد، زیرا به بهای مصرف انرژی بیشتر، کاوش هدف در زمان کمتری اتفاق می‌افتد. هرچند با افزایش جمعیت ربات‌ها، تاثیر مثبت استفاده از استراحت خود را نشان می‌دهد. به دلیل ترافیک ناشی از جمعیت بالا، ربات‌ها برای جستجو در محیط و حمل اهداف به خانه باید دائما در حال دوری از موانع باشند؛ این امر موجب کاهش سرعت و در نتیجه، افزایش زمان کل کاوش هدف خواهد شد. وقتی امکان استراحت برای ربات‌ها فراهم است، کمی از بار ترافیکی جمعیت کاسته می‌شود و بنابراین زمان کل کاوش هدف کاهش می‌یابد. ترافیک به صورت نسبی با توجه به نسبت تعداد ربات به تعداد اهداف و موانع تعریف می‌شود. در سطر اول از شکل ۶-۱، تنها ده هدف وجود دارد؛ بنابراین با افزایش جمعیت ربات‌ها از ۲۰ به ۴۰ عدد، تاثیر منفی ترافیک به تاثیر مثبت افزایش جمعیت ربات‌ها غلبه می‌کند. در این حالت است که وجود استراحت موثر خواهد بود. به همین دلیل است که در سطر سوم از شکل ۶-۱، تاثیر استراحت از همان ابتدا نمایان می‌گردد.

نکته‌ی دیگری که در شکل ۶-۱ قابل مشاهده است، نزدیک شدن عملکرد سیستم با وجود استراحت و بدون وجود آن در جمعیت‌های بالا است. هرچقدر به انتهای محور افقی نزدیک می‌شویم، اختلاف بین زمان‌های کاوش هدف در حالت بدون استراحت و با استراحت کاهش می‌یابد. در پاراگراف قبل، به تاثیر استراحت بر کاهش ترافیک اشاره شد؛ هرچند وقتی ترافیک از حد مشخصی بیشتر شود؛ (برای مثال، ۱۰۰ ربات، ۸۰ هدف و ۲۰ مانع در سطر سوم از شکل ۶-۱)، تاثیر استراحت بر کاهش بار ترافیکی محیط نیز کاهش می‌یابد. معیار R2S بر روی تک تک نقاط مربوط به نمودارهای دارای استراحت نمایش داده شده است؛ مشاهده می‌شود که نرخ جمعیت در حال استراحت در هر نمودار تقریبا یکسان است که بستگی به تعداد اهداف و موانع موجود در محیط دارد. مورد دیگری که می‌توان به آن اشاره کرد، افزایش زمان کاوش هدف پس از افزایش جمعیت ربات‌ها به بیش از ۷۰ است؛ این نکته را می‌توان با توجه به ابعاد محیط توجیه کرد.

## ۶-۳- بررسی تاثیر تعداد اهداف بر عملکرد

در این بخش، آزمایش‌هایی با تعداد اهداف متغیر ترتیب داده شده‌اند. سایر پارامترها ثابت درنظر گرفته شده تا تاثیر تعداد اهداف بر عملکرد سیستم مورد بررسی قرار گیرد. شکل ۶-۲، دو مجموعه آزمایش را به تصویر کشیده است. در مجموعه اول (الف)، تعداد موانع و نویز اعمال شده به حسگرها برابر با صفر

قرار گرفته است؛ همچنین امکان استراحت برای ربات‌ها وجود دارد. در مجموعه دوم (ب)، تنها تعداد ۲۰ موانع به محیط اضافه شده است و سایر پارامترها مشابه هستند.



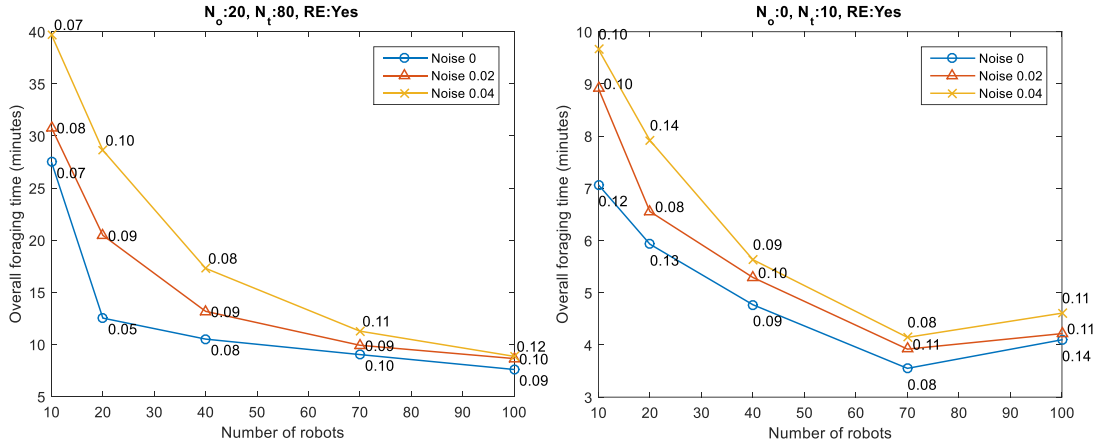

(ب) (الف)

**شکل ۶-۲: مقایسه زمان کل کاوش هدف با تعداد اهداف متفاوت. سایر پارامترها در هر آزمایش ثابت در نظر گرفته شده است. (الف) تعداد صفر مانع، بدون نویز و امکان استراحت، (ب) ۲۰ مانع، بدون نویز و امکان استراحت.**

در هر دو مجموعه آزمایش، مستقل از تعداد اهداف موجود در محیط، زمان کل کاوش هدف با افزایش جمعیت ربات‌ها، کاهش یافته است. با افزایش تعداد ربات‌ها بدون درنظر گرفتن تعداد اهداف، نمی‌توان همیشه به بهترین نتیجه دست یافت. نتایج بدست آمده نشان می‌دهند که افزایش بیش از دو برابری جمعیت ربات‌ها نسبت به تعداد اهداف منجر به کاهش زمان کل کاوش هدف نخواهد شد. افزایش بی‌رویه ربات‌ها منجر به افزایش ترافیک محیط با اندازه ثابت شده و کارایی سیستم را کاهش می‌دهد. هرچند ربات‌ها قادر به استراحت هستند ولی این عامل در ترافیک‌های بالا، تاثیرگذار نخواهد بود.

## ۶-۴- بررسی تاثیر مقدار نویز بر عملکرد

برای بررسی تاثیر مقدار نویز بر عملکرد سیستم، نتایج دو مجموعه آزمایش استخراج شده و در شکل ۶-۳ به نمایش گذاشته شده است. در مجموعه اول (الف)، برای کاوش ده هدف در محیطی بدون وجود مانع، مقادیر مختلف نویز به حسگرها اعمال شد. در مجموعه دوم (ب)، حداکثر تعداد مانع و هدف در محیط وجود دارد؛ یعنی ۲۰ مانع و ۸۰ هدف. در هر دو مجموعه آزمایش، امکان استراحت برای ربات‌ها وجود داشته است.




(ب) (الف)

**شکل ۶-۳: مقایسه زمان کل کاوش هدف با مقدار نویز متفاوت. سایر پارامترها در هر آزمایش ثابت در نظر گرفته شده است. (الف) تعداد صفر مانع، ده هدف و امکان استراحت، (ب) ۲۰ مانع، ۸۰ هدف و امکان استراحت.**

در هر دو بخش از شکل ۶-۳ مشاهده می‌شود که با افزایش جمعیت ربات‌ها، تاثیر نویز بر عملکرد سیستم کاهش می‌یابد. به شکلی که برای جمعیت‌هایی با بیشتر از ۶۰ ربات، عملکرد سیستم به ازای مقادیر مختلف نویز تقریبا برابر است. تنها تفاوت بخش (الف) و (ب) در شکل ۶-۳ ناشی از تعداد اهداف موجود در محیط است؛ با توجه به کم بودن تعداد اهداف در بخش (الف)، با افزایش تعداد ربات‌ها به ۱۰۰، زمان کل کاوش هدف کمی افزایش می‌یابد که با توجه به توضیحات داده شده در بخش‌های قبل، منطقی است.

## ۶-۵- بررسی مقدار انرژی مصرفی

مدل پیشنهادی با استفاده از ماژول ارائه شده در بخش ۵-۴-۶- قادر به محاسبه مقداری انرژی مصرفی ربات‌ها می‌باشد. در این بخش معیاری به نام مقدار انرژی مصرفی در واحد زمان مورد بررسی قرار گرفته است. واحد زمانی مورد استفاده، یک تیک ساعت سراسری سیستم است؛ طبیعتا دلخواه ما، مقدار انرژی مصرفی کمتر در واحد زمان خواهد بود. مقدار انرژی مصرفی و اکتسابی برای جستجو و یافتن اهداف به صورت زیر درنظر گرفته شده است:

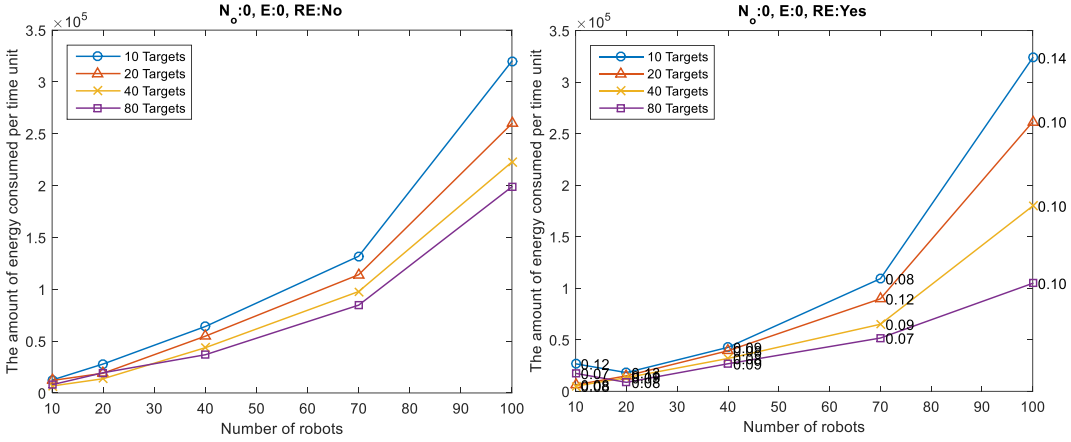
- به ازای هر هدف پیدا شده و به خانه آورده شده، مقدار انرژی $E_f = 2000$ به انرژی اکتسابی ربات افزوده می‌شود. در واقع این مقدار انرژی، از انرژی مصرفی کل، کسر می‌گردد.
- طی کردن هر گام در فاز جستجو، مقدار انرژی $\mathrm{E_s} = 1$ را از انرژی کل ربات می‌کاهد.
- هر واحد زمانی که ربات در فاز استراحت سپری می‌کند، مقدار انرژی $E_r = 0$ را از انرژی کل ربات کم می‌کند.

- حمل غذا به خانه نیز، دو برابر جستجو برای غذا ($E_t = 2$)، انرژی مصرف می‌کند.

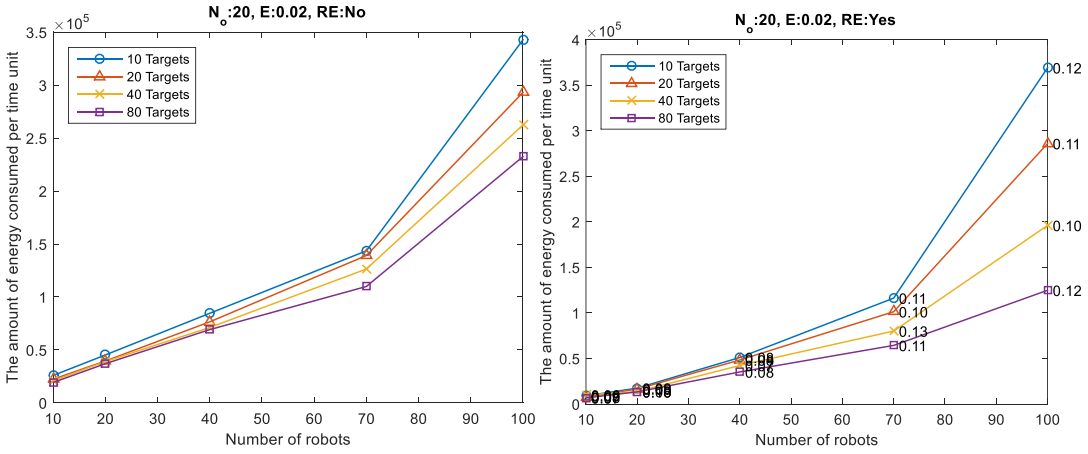
مقادیر ارائه شده به صورت تجربی و با توجه به حدود آستانه درنظر گرفته شده برای آغاز و پایان استراحت، انتخاب گشته‌اند. آزمایشات انجام شده مورد بررسی قرار گرفته و نتایج مرتبط با بخش جاری، در دو گروه زیر استخراج شده‌اند:

- بررسی مقدار انرژی مصرفی در واحد زمان در حالتی که امکان استراحت وجود دارد و وجود ندارد برای حداقل تعداد موانع (صفر مانع) و نویز (بدون نویز).
- بررسی مقدار انرژی مصرفی در واحد زمان در حالتی که امکان استراحت وجود دارد و وجود ندارد برای حداکثر تعداد موانع (۲۰ مانع) و نویز (۰٫۰۲).

نتایج مربوط به دو گروه مورد اشاره در شکل ۶-۴ به نمایش گذاشته شده است. نکته‌ی اولی که در این تصویر قابل مشاهده می‌باشد، مصرف انرژی بیشتر در آزمایش‌هایی است که تعداد اهداف کمتری وجود دارد. علت این امر، دریافت انرژی مثبت به ازای اهداف به خانه آورده شده است. طبیعتا با جمعیت بیشتر و تعداد اهداف بیشتر، می‌توان با سرعت بیشتری اهداف را جمع‌آوری کرد و بنابراین، انرژی مصرفی در واحد زمان را کاهش داد.




(ب) (الف)




(د) (ج)

**شکل ۶-۴: مقایسه مقدار انرژی مصرفی در واحد زمان با تعداد ربات، اهداف و نویز متفاوت. (الف) تعداد صفر مانع، بدون نویز و امکان استراحت، (ب) تعداد صفر مانع، بدون نویز و عدم امکان استراحت، (ج) ۲۰ مانع، نویز ۰،۰۲ و امکان استراحت، (د) ۲۰ مانع، نویز ۰،۰۲ و امکان عدم استراحت.**

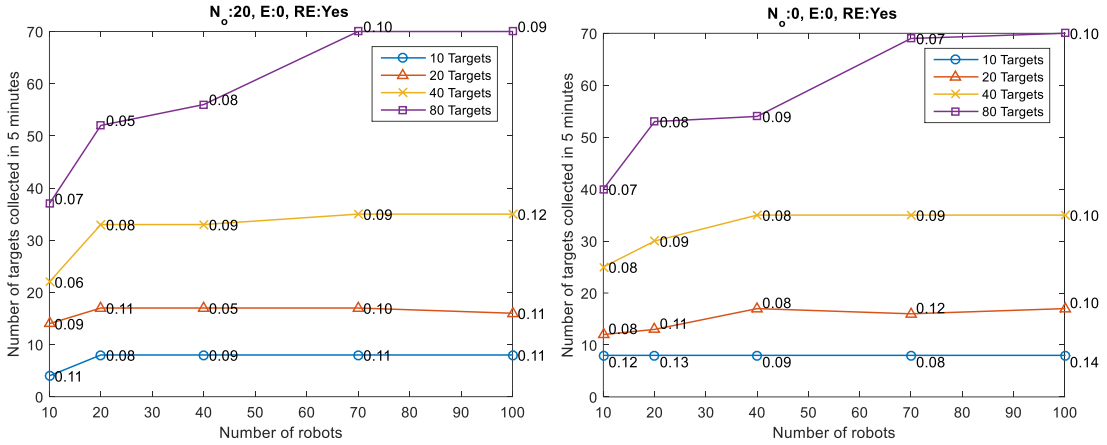
حال به بررسی سطر اول از شکل ۶-۴ می‌پردازیم. در این سطر، تعداد موانع موجود در محیط و نویز اعمال شده به حسگرها، صفر می‌باشد. تفاوت شکل (الف) و (ب)، در وجود و عدم وجود استراحت است. در حالتی که ربات‌ها امکان استراحت کردن دارند، شیب نمودار کمتر است؛ از طرفی، با افزایش جمعیت ربات‌ها، شیب، ملایم‌تر نیز خواهد شد. اختلاف مقدار انرژی مصرف شده در واحد زمان بین دو تصویر، با افزایش جمعیت ربات‌ها، بیشتر می‌شود. به شکلی که برای ۱۰۰ ربات، به بیشترین حد خود می‌رسد. در اینجا بهتر می‌توان به تاثیر مثبت استراحت بر انرژی مصرفی پی برد.

در سطر دوم از شکل ۶-۴، تعداد موانع برابر با حداکثر میزان خود تنظیم شده‌اند که برابر است با تعداد ۲۰ مانع و نویز نیز برابر با ۰،۰۲ قرار گرفته است. توضیحات داده شده برای سطر اول، در مورد این سطر نیز صادق است. علاوه‌براین، به دلیل وجود موانع و نویز، مقدار مصرف انرژی در واحد زمان نسبت به سطر اول افزایش پیدا کرده است. همچنین، شیب موجود در تصاویر (ج) و (د) بیشتر از تصاویر معادل در سطر اول می‌باشد. با این حال، مشاهده می‌شود که تاثیر استراحت حتی با وجود تعداد موانع حداکثری و نویز، مثبت است که در جمعیت‌های بالاتر، بیشتر اثر خود را نشان می‌دهد.

## ۶-۶- کاوش هدف در زمان ثابت

یکی از پارامترهای کاربردی که معمولا در آزمایش‌ها سنجیده می‌شود، کاوش هدف در زمان ثابت است؛ در این آزمایش، تعداد اهداف جمع‌آوری شده در یک بازه زمانی ثابت اندازه گرفته می‌شود. با تغییر تعداد اهداف و اندازه جمعیت ربات‌ها می‌توان میزان انعطاف سیستم و عملکرد آن را مورد بررسی قرار

داد. نتایج ارائه شده در شکل ۶-۵، عملکرد سیستم را با تعداد اهداف و موانع متفاوت به نمایش گذاشته است.




(ب) (الف)

**شکل ۶-۵: مقایسه تعداد اهداف جمع‌آوری شده در زمان ثابت پنج دقیقه با تعداد اهداف متفاوت. سایر پارامترها در هر آزمایش ثابت در نظر گرفته شده است. (الف) تعداد صفر مانع، بدون نویز و امکان استراحت، (ب) ۲۰ مانع، بدون نویز و امکان استراحت.**

زمان ثابت در نظر گرفته شده برای این آزمایش‌ها، پنج دقیقه است. این زمان با توجه به سریع‌ترین زمان کاوش هدف انتخاب شده است. بازه زمانی مورد بررسی باید از کمترین زمان کل کاوش هدف کوچکتر باشد. در نتایج ارائه شده مشاهده می‌شود که با افزایش تعداد اهداف در محیط، تعداد اهداف جمع‌آوری شده در زمان ثابت نیز افزایش می‌یابد؛ که امری منطقی است. مشاهده جالب توجه، افزایش تعداد اهداف جمع‌آوری شده با وجود افزایش جمعیت ربات‌ها است؛ البته افزایش جمعیت تا حد مشخصی منجر به نتیجه مورد اشاره می‌گردد. برای نمونه، با وجود ۸۰ هدف در محیط، افزایش جمعیت از ده به صد، افزایش پیوسته تعداد اهداف جمع‌آوری شده را به دنبال دارد. در صورتی که با وجود بیست هدف، این افزایش تنها تا جمعیت ۴۰ تایی ربات‌ها اتفاق می‌افتد. اشاره به این نکته نیز حائز اهمیت است که با افزایش جمعیت نسبت به تعداد اهداف، تعداد ربات‌های بیشتری به استراحت پرداخته و همچنین زمان بیشتری صرف دوری از موانع می‌گردد؛ در نتیجه باید انتظار کاهش تعداد اهداف جمع‌آوری شده در زمان ثابت را داشت. علاوه‌بر موارد مورد اشاره، مشاهده می‌شود که افزایش تعداد موانع در بخش (ب) از شکل ۶-۵، تنها منجر به کاهش جزئی در عملکرد سیستم گشته است.

## ۶-۷- جمع‌بندی

پس از طراحی فرمال وظیفه کاوش هدف در فصل‌های پیشین، در این فصل، به شبیه‌سازیِ پیاده‌سازی انجام شده در محیط ARGoS پرداخته شد. برای بررسی صحت عملکرد سیستم، ۴۸۰ آزمایش مختلف

با پارامترهای متفاوت ترتیب داده شد. هر آزمایش نیز پنج مرتبه تکرار گشت. معیارهای متفاوتی از آزمایش‌های انجام شده استخراج شد؛ و پارامترهای متفاوتی نیز در هر آزمایش مورد استفاده قرار گرفت. به این وسیله، چند جنبه از عملکرد سیستم تحلیل و اعتبارسنجی شد و نتایج مربوطه ارائه گشت؛ جوانبی همچون، تاثیر استراحت بر عملکرد سیستم، تاثیر تعداد اهداف و مقدار نویز اعمال شده به حسگرها در عملکرد سیستم و کاوش هدف در زمان ثابت. به واسطه نتایج ارائه شده، می‌توان یک دید مناسب نسبت به عملکرد سیستم در شرایط مختلف بدست آورد؛ شرایطی مانند تعداد موانع و جمعیت ربات‌ها، تعداد اهداف و مقدار نویز. ویژگی‌های رباتیک جمعی مانند انعطاف، مقیاس‌پذیری، مقاومت و ارتباطات محلی و محدود در پیاده‌سازی صورت گرفته وجود دارند.

نتیجه‌ی مهمی که می‌توان از این بخش بدست آورد، عملکرد قابل قبول سیستم در انجام وظیفه اصلی آن یعنی کاوش هدف می‌باشد. پس از انجام بیش از ۵۰۰ آزمایش، می‌توان به صحت کد تولید شده برای وظیفه کاوش هدف توسط ابزار جامع پیشنهادی اطمینان کرد. هرچند پیاده‌سازی توابع فراخوان، جزئی از کد تولید شده نیست و باید به آن توجه ویژه مبذول داشت.

# فصل

# ۷- نتیجه‌گیری

رباتیک جمعی از موضوعات مورد توجه محققان در حوزه رباتیک و هوش مصنوعی است. محققان تلاش می‌کنند تا از هوش جمعی حیوانات در طبیعت الهام گرفته و تا حد امکان، واقعی‌ترین شکل آن را پیاده کنند. برای پیاده‌سازی سیستم‌های رباتیک جمعی از روش‌های متعددی استفاده می‌شود؛ یکی از این روش‌ها، روش‌های فرمال هستند که رفتارهای سیستم و نیازمندی‌های کنترلی سیستم را در قالب مدل‌های فرمال توصیف می‌کنند. غالبا، این روش‌ها در صنعت به راحتی پذیرفته نمی‌شوند؛ زیرا پس از طراحی فرمال، روال پیاده‌سازی سیستم‌ها باید از نو آغاز شده و به صورت مستقل پیگیری شود. بنابراین طراحی فرمال یک گام مجزا، زمان‌بر و پرهزینه قلمداد می‌شود.

در این رساله، تئوری کنترل سوپروایزری برای پیاده‌سازی وظیفه کاوش هدف پیشنهاد شده است. هدف از این پیشنهاد، کم کردن فاصله بین طراحی فرمال یک سیستم و پیاده‌سازی آن است. بنابراین می‌توان هم از مزایای طراحی فرمال بهره برد و هم از این گام پرهزینه در جهت پیاده‌سازی کم‌نقص‌تر استفاده کرد. برای این منظور، ابزار جامعی پیشنهاد شده است که طراحی فرمال را دریافت کرده و کد کنترلی را به صورت خودکار تولید می‌کند. کد تولید شده معادل با طراحی صورت گرفته می‌باشد؛ بنابراین نیازی به اعتبارسنجی و آزمون مجدد کد تولید شده نیست؛ علاوه‌براین، عملیات مرتبط با ماشین‌های حالت و مولدها نیز توسط ابزار جامع پیشنهادی انجام می‌شود؛ عملیات سنتز و اعتبارسنجی سوپروایزرها نیز به صورت خودکار صورت می‌گیرد. بنابراین نیازی به کمینه‌سازی مولدها، حذف حالات غیرقابل دسترس و حالات بد و سایر محاسبات زمان‌بر توسط طراح نیست. در روند طراحی ممکن است اشتباهاتی توسط طراح صورت گیرد؛ اشتباهات قابل تشخیص ساختاری توسط ابزار پیشنهادی به طراح تذکر داده می‌شوند. وجود حالت بد در سوپروایزرها، تعریف اشتباه حالت‌ها و رخدادها، عدم وجود حالت اولیه و حالات علامت‌دار و وجود چند گذار خروجی با رخداد یکسان از یک حالت مشخص چند مورد از موارد نامبرده هستند.

تئوری کنترل سوپروایزری دارای محدودیت‌هایی در طراحی نیز هست؛ برای مثال، تعریف رخدادهای احتمالی در ماشین‌های حالت استاندارد امکان‌پذیر نیست؛ و یا استفاده از رخدادهای زمان‌دار به سادگی صورت نمی‌پذیرد. برای حل تعدادی از این محدودیت‌ها، استفاده از تئوری کنترل سوپروایزری احتمالاتی و زمان‌دار پیشنهاد شد. با استفاده از این دو مولفه، می‌توان مسائل پیچیده‌تر را با مدل‌های کوچک‌تر حل کرد. چنان‌که در بخش‌های مربوطه از رساله نیز این امر نشان داده شد. ابزار جامع پیشنهادی نیز برای استفاده از دو مولفه احتمالات و زمان توسعه داده شد. کد تولید شده توسط این ابزار، قابل اجرا بر روی سکوی ARGoS خواهد بود. ابزار پیشنهادی از دو لایه اصلی تشکیل شده است؛ لایه‌ی اول عملیات مربوط به محاسبه سوپروایزر و تولید کد کنترلی را انجام می‌دهد؛ لایه‌ی دوم بر روی سکوی ARGoS پیاده‌سازی گشته و واسط موردنیاز برای اجرای کد کنترلی تولید شده را فراهم می‌آورد. جزئیات

پیاده‌سازی این دو لایه در بخش‌های مربوطه توضیح داده شده‌اند. برای بررسی صحت عملکرد ابزار پیشنهادی و آشنایی با روش عملکرد آن در طراحی، به پیاده‌سازی دو وظیفه کوچک پرداخته شد. روند طراحی این دو وظیفه به همراه نتایج حاصل شده از شبیه‌سازی آن‌ها نیز ارائه گشت.

پس از معرفی روش پیشنهادی و تشریح ابزار جامع پیشنهادی، به موضوع اصلی رساله یعنی پیاده‌سازی وظیفه کاوش هدف پرداخته شد. تمامی مراحل طراحی وظیفه کاوش هدف ارائه و توضیح داده شد. پس از اتمام طراحی و تولید کد موردنیاز برای انجام شبیه‌سازی، توابع فراخوان نیز به کد تولید شده اضافه شدند؛ آنگاه تعداد دورهای متعددی از شبیه‌سازی با پارامترهای مختلف در محیط ARGoS به اجرا درآمدند. پارامترهای مورد مطالعه از نتایج آزمایش‌ها استخراج گشته و در قالب نمودارهایی ارائه شدند. طبیعتا ارائه نتایج همه‌ی آزمایش‌ها به صورت خام، ارزش افزوده‌ای به همراه نخواهد داشت؛ بنابراین موضوعات مورد علاقه این رساله که در رابطه با وظیفه کاوش هدف دارای ارزش بودند، معرفی شده و تنها نتایج مرتبط در بخش‌های مختلف ارائه گشتند. آنگاه به تحلیل و بررسی هر یک از موضوعات مورد مطالعه پرداخته شد. ویژگی‌های خاص رباتیک جمعی مانند همگن بودن، عدم وجود رهبر مرکزی، انعطاف، مقیاس‌پذیری، مقاومت و ارتباطات محلی و محدود در پیاده‌سازی صورت گرفته وجود دارند. همه‌ی ربات‌ها یکسان بوده و رهبر کنترل کننده‌ای ندارند. تغییر وظایف در طول زمان نشان دهنده‌ی انعطاف سیستم است. با افزایش یا کاهش تعداد ربات‌ها در حین عملیات مشکلی در عملکرد سایرین ایجاد نخواهد شد. بنابرای سیستم مقیاس‌پذیر و مقاوم می‌باشد. سنسورها برد زیادی نداشته و تعامل و ارتباطی نیز بین ربات‌ها وجود ندارد.

# فصل

# ۸- کارهای آینده

مانند هر راه‌حل دیگری، راه‌حل پیاده شده در این رساله نیز قابل بهبود است؛ هریک از ضابطه‌های کنترلی پیشنهاد شده می‌توانند به اشکال دیگر نیز طراحی شوند. یکی از مهمترین کارهای آینده پیشنهادی، پیاده‌سازی وظیفه کاوش هدف به صورت تعاملی است؛ به شکلی که ربات‌ها با یکدیگر در ارتباط باشند؛ این تعامل می‌تواند در جنبه‌های مختلفی اتفاق بیفتد؛ برای مثال، هر ربات می‌تواند با استفاده از فرومون مصنوعی[1]، مکان اهدافی که پیدا کرده را به سایرین اطلاع دهد. فرومون مصنوعی می‌تواند برای نشانه‌گذاری مسیر خانه نیز به کار گرفته شود. حتی می‌توان از چند نوع فرومون در محیط بهره برد؛ برای نمونه، یک نوع فرومون برای نشانه‌گذاری مسیر اهداف در محیط و یک نوع دیگر برای تعیین مسیر خانه. برای پیاده‌سازی فرمون مصنوعی راه‌حل‌های متعددی پیشنهاد شده است [۱۲۶–۱۲۸].

ربات‌ها می‌توانند برای انتقال از حالت جستجو به استراحت و برعکس، از سایر ربات‌ها نظرخواهی کنند. برای پیدا کردن مسیر خانه نیز می‌توان از ربات‌های راهنما استفاده کرد؛ طوری که ربات‌های مشخصی، مسئولیت نشان دادن مسیر خانه به سایرین را بر عهده داشته باشند [۹۴, ۱۲۱]. هرچند نظرخواهی به صورت سراسری با قید ارتباطات محدود و محلی در رباتیک جمعی تناقض دارد.

یکی دیگر از رویکردهای حل مسئله، تشکیل مسیر است. در این روش، یک یا چند مسیر ترجیحی در محیط ساخته شده تا ربات‌های جستجو کنندهیا ربات‌های حامل آذوقه بتوانند به صورت بهینه حرکت نمایند. این مسیر به تدریج توسط ربات‌هایی که عمل کاوش را انجام داده‌اند ساخته شده و در صورت از بین رفتن دلیل ایجاد آن‌ها، حذف می‌شوند. به عنوان مثال هنگامی که یکی از منابع غذایی تخلیه شود.

در مواقعی که ترافیک سیستم از حدی بیشتر می‌شود، تاثیر استراحت بر عملکرد سیستم کاهش می‌یابد. در این حالت سیستم می‌تواند استراتژی خود را تغییر داده و به جای استفاده از استراحت، از روش‌های تعاملی در مسیریابی یا انتقال غذا استفاده کند. ایده‌ی دیگری که می‌تواند به بهبود عملکرد سیستم منجر شود، استفاده از چند خانه یا مکان میانی است؛ می‌توان چند مکان موقت برای جمع‌آوری اهداف درنظر گرفت و ربات‌های خاصی وظیفه انتقال اهداف از مکان‌های موقت به خانه را داشته باشند؛ مکان‌های موقت می‌توانند جای خود را به خانه‌ها بدهند و هر ربات، هدف موررنظر خود را به نزدیک‌ترین خانه انتقال دهد. یک ایده دیگر، در نظر گرفتن چند گروه ربات است؛ یک گروه وظیفه جمع‌آوری اهداف در فواصل دورتر از خانه را بر عهده گرفته و اهداف را به نزدیکی خانه انتقال دهند؛ گروهی دیگر، اهداف را از نزدیکی خانه به خود خانه حمل کنند.

[1] Artificial Pheromone

حمل شی در وظیفه‌ی کاوش هدف معمولا به دو شکل صورت می‌گیرد. در حالت اول ربات بدون هماهنگی و کمک گرفتن از سایرین، شی را به لانه منتقل می‌کند. در حالت دوم ربات‌ها برای جابجایی اشیا، هماهنگ عمل می‌کنند. هماهنگی برای حمل یک شی، الهام گرفته از حمل گروهی غذا توسط مورچه‌ها می‌باشد. در صورتی که ربات‌ها برای بردن غذا به لانه همکاری کنند، از ایجاد تداخل و برخورد آن‌ها کاسته می‌شود. به عنوان مثال ممکن است ربات‌ها بخواهند جعبه‌ای را هل دهند که یک ربات به تنهایی قادر به هل دادن آن نیست. یکی از نکات قابل توجه در این وظیفه، جهت هل دادن ربات‌ها است که اگر هماهنگ نباشد ممکن است نیروی یکدیگر را خنثی نمایند. ممکن است یک جهت از پیش تعیین شده به ربات‌ها داده شود یا اینکه از نیروهای جاذبه‌ی تعریف شده استفاده گردد. می‌توان لامپی در مقصد روشن کرد که ربات‌ها بتوانند این لامپ روشن را دیده و به سمت آن حرکت کنند.

مثالی دیگر از حمل‌ونقل تجمعی، برداشتن تکه‌های چوب از یک مکان و بردن آن به مکان دیگر است [۷۳]. اندازه‌ی چوب‌ها به گونه‌ای است که دو ربات باید آن را با همکاری یکدیگر حمل نمایند. رباتی که چوب را می‌یابد، یک سر آن را گرفته و به اندازه‌ی مشخصی منتظر می‌ماند. در صورتی که ربات دیگری به سمت او آمد با همکاری یکدیگر چوب را حمل می‌کنند؛ در غیر این صورت، چوب را رها کرده و به دنبال چوب دیگری می‌گردد. از آن‌جا که ربات‌ها محل قرار گرفتن چوب‌ها را نمی‌دانند، کارایی سیستم به تعاملات تصادفی بین ربات‌ها و بین ربات‌ها و محیط بستگی دارد. معیار سنجش عملکرد این رفتار را می‌توان زمان لازم برای حمل یک شی به مقصد در نظر گرفت [۱۲۹]. روش دیگر، استفاده از نسبت طول مسیر بهینه بین مبدا و مقصد و مسیر استفاده شده در سیستم است [۹۷].

در آینده می‌توان اهداف موجود در محیط را هوشمند در نظر گرفت. اهدافی که توانایی حرکت دارند و یا فرار می‌کنند. مطمئنا یافتن هدفی که بدون الگوی مشخصی در حرکت است، ساده نیست. در صورتی که هدف، با در نظر گرفتن ربات متهاجم، سعی در فرار داشته باشد، کار به مراتب سخت‌تر نیز خواهد شد.

ارائه‌ی معیار ارزیابی مناسب به منظور بررسی مواردی چون عملکرد تئوری کنترل سوپروایزری در طراحی رباتیک جمعی، میزان مشارکت اعضای اجتماع در انجام وظیفه‌ی جمعی، وجود یا عدم وجود خصیصه‌های اصلی رباتیک جمعی مانند مقاومت و مقیاس‌پذیری از جمله مواردی است که در ادامه می‌توان به آن پرداخت.

با الهام از روش AutoMoDe-vanilla، که در فصل دو معرفی شد، می‌توان با استفاده از الگوریتم‌های بهینه‌سازی پارامترهای ماشین حالت متناهی را تعیین کرد. در این صورت طراحی به صورت خودکار صورت خواهد گرفت. در این رساله وظایفی چون تجمع ربات‌ها، همگام‌سازی، جستجوی تجمعی، حمل و نقل تجمعی (بدون همکاری) و کاوش هدف که ترکیبی از چند وظیفه‌ی کوچک‌تر است، طراحی و

پیاده‌سازی شدند. طراحی وظایف متفاوتی چون آرایش زنجیره‌ای و خوداتصالی نیز از جمله مواردی هستند که می‌توان در آینده به آن پرداخت.

انتقال رباتیک جمعی به مقیاس میکرو یا نانو یکی از موضوعات جالب برای تحقیقات آینده است. از کاربردهای نانو رباتیک می‌توان به کاوش فضا، کاوش‌های زیرزمینی و مخازن نفت، انتقال دارو به نقاط مختلف بدن و مبارزه با تومورها اشاره نمود.

# مراجع


[1] Prasad, S., Rawool, S. “Swarm Robotics : Nature Inspired Systems.”, *International Journal of Engineering Research and General Science*., vol. 4, no. 5, pp. 168–174, 2016.

[2] Http://blogs.cornell.edu “NEIL LEACH: SWARM URBANISM,” [Online] 2015, http://blogs.cornell.edu/arch5302sp15/2015/05/21/neil-leach-swarm-urbanism/ (Accessed: 14 October 2017).

[3] Wilde, S. “Thoughts Forming in a Fish,” [Online] 2013, http://www.stuartwilde.com/2013/02/thoughts-forming-in-a-fish/ (Accessed: 14 October 2017).

[4] Anderson, C., Theraulaz, G., Deneubourg, J.-L. “Self-Assemblage in Insect Societies.”, *Insect Sociaux*., vol. 49, no. 2, pp. 99–110, 2002.

[5] Şahin, E. “Swarm Robotics: From Sources of Inspiration to Domains of Application.”, *International workshop on swarm robotics*, pp. 10–20, Springer, Berlin, Heidelberg, 2004.

[6] Kaminka, G.A., Schechter-Glick, R., Sadov, V. “Using Sensor Morphology for Multirobot Formations.”, *IEEE Transactions on Robotics*., vol. 24, no. 2, pp. 271–282, 2008.

[7] Brambilla, M., Ferrante, E., Birattari, M., Dorigo, M. “Swarm Robotics : A Review from the Swarm Engineering Perspective.”, *Swarm Intelligence*., vol. 7, no. 1, pp. 1–41, 2013.

[8] Bonabeau, E., Recherches, D. de, Marco, D.F., Dorigo, M., Théraulaz, G., Theraulaz, G. Swarm intelligence: from natural to artificial systems., Oxford university press, 1999.

[9] Khaldi, B., Cherif, F. “An Overview of Swarm Robotics : Swarm Intelligence Applied to Multi-Robotics.”, *International Journal of Computer Applications*., vol. 126, no. 2, pp. 31–37, 2015.

[10] Martínez, F., Jacinto, E., Martínez, F. “Study of Collective Robotic Tasks Based on the Behavioral Model of the Agent.”, *International Conference on Intelligent Data Engineering and Automated Learning*, pp. 224–231, 2015.

[11] Sahoo, S., Sahoo, D. “DISTRIBUTED ROBOTICS AND AD HOC NETWORKING.”, *INTERNATIONAL JOURNAL OF RESEARCH IN COMPUTER APPLICATIONS AND ROBOTICS*., vol. 4, no. 10, pp. 13–17, 2016.

[12] Arkin, R.C., Bekey, G.. “Special Issue on Robot Colonies.”, *Autonomous Robots*., vol. 4, no. 1, pp. 1–153, 1997.

[13] Abidin, Z.Z., Arshad, M.R., Ngah, U.K. “An Introduction to Swarming Robotics: Application Development Trends.”, *Artificial Intelligence Review*., vol. 43, no. 4, pp. 501–514, 2015.

[14] Iocchi, L., Nardi, D., Salerno, M. “Reactivity and Deliberation: A Survey on Multi-Robot Systems.”, *Balancing Reactivity and Social Deliberation in Multi-Agent Systems, From RoboCup to Real-World Applications*., pp. 9–34, 2001.

[15] Dudek, G., Jenkin, M.M., Milios, E., Wilkes, D. “A Taxonomy for Multi-Agent Robotics.”, *Autonomous Robots*., vol. 3, no. 4, 1996.

[16] Dorigo, M., Tuci, E., Mondada, F., Nolfi, S., Deneubourg, J.-L., Floreano, D., Gambardella, L.M. “The SWARM-BOTS Project.”, *International Workshop on Swarm Robotics*, pp. 31–44, 2005.

[17] Dorigo, M., Trianni, V., Şahin, E., Groß, R., Labella, T.H., Baldassarre, G., Nolfi, S., Deneubourg, J.-L., Mondada, F., Floreano, D. “Evolving Self-Organizing Behaviors for a Swarm-Bot.”, *Autonomous Robots*., vol. 17, no. 2–3, pp. 223–245, 2004.

[18] Jevtic, A., Andina, D. “Swarm Intelligence and Its Applications in Swarm Robotics.”, *Conference on Computational Intelligence, Man-Machine Systems and Cybernetics*, pp. 41–46, 2007.

[19] Dorigo, M., Sahin, E. “Swarm Robotics.”, *Autonomous Robots*., vol. 17, no. 2–3, pp. 111–113, 2004.

[20] Tan, Y., Zheng, Z.-Y. “Research Advance in Swarm Robotics.”, *Defence Technology*., vol. 9, no. 1, pp. 18–39, 2013.

[21] Khosravi, S., Jahangir, M., Afkhami, H. “Adaptive Fuzzy SMC-Based Formation Design for Swarm of Unknown Time-Delayed Robots.”, *Nonlinear Dynamics*., vol. 69, no. 4, pp. 1825–1835, 2012.

[22] Ranjbarsahraei, B., Roopaei, M., Khosravi, S. “Adaptive Fuzzy Formation Control for a Swarm of Nonholonomic Differentially Driven Vehicles.”, *Nonlinear Dynamics*., vol. 67, no. 4, pp. 2747–2757, 2012.

[23] نعمتی اصطهباناتی, ع. “بررسی تجربی حرکت توده ای رباتها.”, دانشگاه صنعتی شریف, ۱۳۸۹.

[24] Sahraei, B.R., Nemati, A., Farshchi, M., Meghdari, A. “ADAPTIVE FUZZY SLIDING MODE CONTROL APPROACH FOR SWARM FORMATION CONTROL OF MULTI-AGENT SYSTEMS.”, *10th Biennial Conference on Engineering Systems Design and Analysis*, pp. 1–6, 2010.

[25] Vaughan, R. “Massively Multi-Robot Simulation in Stage.”, *Swarm Intelligence*., vol. 2, no. 2–4, pp. 189–208, 2008.

[26] Collett, T.H.J., MacDonald, B.A., Gerkey, B.P. “Player 2.0: Toward a Practical Robot Programming Framework.”, *Proceedings of the Australasian Conference on Robotics and Automation*, p. 145, 2005.

[27] Koenig, N., Howard, A. “Design and Use Paradigms for Gazebo, an Open-Source Multi-Robot Simulator.”, *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems.*, pp. 2149–2154, 2004.

[28] “Homepage of Enki,” .

[29] Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Zufferey,

J., Floreano, D., Martinoli, A. “The E-Puck, a Robot Designed for Education in Engineering.”, *Proceedings of the 9th conference on autonomous robot systems and competitions*, pp. 59–65, 2009.

[30] Tricks, M.S.R. “A Behavior-Based Distributed Algorithm Library for Programming Swarms of Robots.”, *Paper submitted for requirements of BSEE at MIT.*, 2004.

[31] Michel, O. “WebotsTM: Professional Mobile Robot Simulation.”, *arXiv preprint cs/0412052.*, 2004.

[32] Klein, J., Spector, L. “3d Multi-Agent Simulations in the Breve Simulation Environment.”, *Artificial life models in software*, pp. 79–106, 2009.

[33] “Homepage of V-Rep Project,” .

[34] Pinciroli, C., Trianni, V., O’Grady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Di Caro, G., Ducatelle, F., Birattari, M., Gambardella, L.M., Dorigo, M. “ARGoS: A Modular, Parallel, Multi-Engine Simulator for Multi-Robot Systems.”, *Swarm Intelligence.*, vol. 6, no. 4, pp. 271–295, 2012.

[35] Pugh, J., Raemy, X., Favre, C., Falconi, R., Martinoli, A. “A Fast Onboard Relative Positioning Module for Multirobot Systems.”, *IEEE/ASME Transactions on Mechatronics.*, vol. 14, no. 2, pp. 151–162, 2009.

[36] Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.-C., Floreano, D., Martinoli, A. “The E-Puck, a Robot Designed for Education in Engineering.”, *Proceedings of the 9th conference on autonomous robot systems and competitions*, pp. 59–65, 2009.

[37] Caprari, G., Siegwart, R. “Mobile Micro-Robots Ready to Use: Alice.”, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3295–3300, 2005.

[38] Kornienko, S., Kornienko, O., Levi, P. “Minimalistic Approach towards Communication and Perception in Microrobotic Swarms.”, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2228–2234, 2005.

[39] Valdastri, P., Corradi, P., Menciassi, A., Schmickl, T., Crailsheim, K., Seyfried, J., Dario, P. “Micromanipulation, Communication and Swarm Intelligence Issues in a Swarm Microrobotic Platform.”, *Robotics and Autonomous Systems.*, vol. 54, no. 10, pp. 789–804, 2006.

[40] Mondada, F., Gambardella, L.M., Floreano, D., Nolfi, S., Deneuborg, J.-L., Dorigo, M. “The Cooperation of Swarm-Bots: Physical Interactions in Collective Robotics.”, *IEEE Robotics & Automation Magazine.*, vol. 12, no. 2, pp. 21–28, 2005.

[41] Turgut, A.E., Gokce, F., Celikkanat, H., Bayindir, L., Sahin, E. “Kobot: A Mobile Robot Designed Specifically for Swarm Robotics Research.”, *Middle East Technical University, Ankara, Turkey, METUCENG-TR Tech. Rep.*, vol. 5, 2007.

[42] Rubenstein, M., Cornejo, A., Nagpal, R. “Programmable Self-Assembly in a Thousand-Robot Swarm.”, *Science.*, vol. 345, no. 6198, pp. 795–799, 2014.

[43] Deyle, T. “I-Swarm Micro Robots Realized -- Impressive Full-System Integration,” [Online] 2009, http://www.hizook.com/blog/2009/08/29/i-swarm-micro-robots-realized-impressive-full-system-integration (Accessed: 14 October 2017).

[44] “S-Bot Mobile Robot,” [Online] 2017, https://en.wikipedia.org/wiki/S-bot_mobile_robot (Accessed: 14 October 2017).

[45] Bayındır, L. “A Review of Swarm Robotics Tasks.”, vol. 172, pp. 292–321, 2016.

[46] Russell, K., Schader, M., Andrea, K., Luke, S. “Swarm Robot Foraging with Wireless Sensor Motes.”, *International Conference on Autonomous Agents and Multiagent Systems*, pp. 287–295, 2015.

[47] Ducatelle, F., Di Caro, G.A., Pinciroli, C., Gambardella, L.M. “Self-Organized Cooperation between Robotic Swarms.”, *Swarm Intelligence*., vol. 5, no. 2, pp. 73–96, 2011.

[48] Stirling, T., Floreano, D. “Energy Efficient Swarm Deployment for Search in Unknown Environments.”, *International Conference on Swarm Intelligence*, pp. 562–563, 2010.

[49] Lopes, Y.K., Trenkwalder, S.M., Leal, A.B., Dodd, T.J., Groß, R. “Supervisory Control Theory Applied to Swarm Robotics.”, *Swarm Intelligence*., vol. 10, no. 1, pp. 65–97, 2016.

[50] Navarro, I., Matía, F. “An Introduction to Swarm Robotics.”, *ISRN robotics*., vol. 2013, pp. 1–10, 2013.

[51] Brambilla, M., Dorigo, M., Birattari, M. “Property-Driven Design for Robot Swarms : A Design Method Based on Prescriptive Modeling and Model Checking.”, *ACM Transactions on Autonomous and Adaptive Systems*., vol. 9, no. 4, pp. 17, 2015.

[52] Arvin, F., Samsudin, K., Ramli, A.R., Bekravi, M. “Imitation of Honeybee Aggregation with Collective Behavior of Swarm Robots.”, *International Journal of Computational Intelligence Systems*., vol. 4, no. 4, pp. 739–748, 2011.

[53] Kernbach, S., Häbe, D., Kernbach, O., Thenius, R., Radspieler, G., Kimura, T., Schmickl, T. “Adaptive Collective Decision-Making in Limited Robot Swarms without Communication.”, *International Journal of Robotics Research*., vol. 32, no. 1, pp. 35–55, 2013.

[54] Fatès, N. “Solving the Decentralised Gathering Problem with a Reaction–Diffusion–Chemotaxis Scheme.”, *Swarm Intelligence*., vol. 4, no. 2, pp. 91–115, 2010.

[55] Francesca, G., Brambilla, M., Brutschy, A., Trianni, V., Birattari, M. “AutoMoDe: A Novel Approach to the Automatic Design of Control Software for Robot Swarms.”, *Swarm Intelligence*., vol. 8, no. 2, pp. 89–112, 2014.

[56] Nouyan, S., Campo, A., Dorigo, M. “Path Formation in a Robot Swarm.”, *Swarm Intelligence*., vol. 2, no. 1, pp. 1–23, 2008.

[57] Soysal, O., Sahin, E. “Probabilistic Aggregation Strategies in Swarm Robotic

Systems.", *IEEE Proceedings In Swarm Intelligence Symposium*, pp. 325–332, 2005.

[58] Liu, W., Winfield, A.F.T., Sa, J. "Modelling Swarm Robotic Systems: A Case Study in Collective Foraging.", *Towards Autonomous Robotic Systems*, pp. 25–32, 2007.

[59] Attilio Priolo "Swarm Aggregation Algorithms for Multi-Robot Systems.", University of the West of England, 2013.

[60] Bachrach, J., Beal, J., McLurkin, J. "Composable Continuous-Space Programs for Robotic Swarms.", *Neural Computing and Applications*., vol. 19, no. 6, pp. 825–847, 2010.

[61] Kazadi, S. "Model Independence in Swarm Robotics.", *International Journal of Intelligent Computing and Cybernetics*., vol. 2, no. 4, pp. 672–694, 2009.

[62] Nolfi, S., Floreano, D., Floreano, D.D. Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines., MIT press, 2000.

[63] Nolfi, S., Floreano, D. Evolutionary robotics., MIT Press, 2001.

[64] Nolfi, S., Bongard, J., Husbands, P., Floreano, D. Evolutionary robotics., Springer Handbook of Robotics, Cham, 2016.

[65] Mahfoud, S. "Niching Methods for Genetic Algorithms.", 1995.

[66] Gould, J.L. "Animal Navigation: The Evolution of Magnetic Orientation.", *Current Biology*., vol. 18, no. 11, pp. R482–R484, 2008.

[67] Kaelbling, L., Littman, M., Cassandra, A. "Artificial Intelligence Planning and Acting in Partially Observable Stochastic Domains.", *Artificial Intelligence*., vol. 101, no. 101, pp. 99–134, 1998.

[68] Konur, S., Dixon, C., Fisher, M. "Analysing Robot Swarm Behaviour via Probabilistic Model Checking.", *Robotics and Autonomous Systems*., vol. 60, no. 2, pp. 199–213, 2012.

[69] Bayindir, L., Şahin, E. "A Review of Studies in Swarm Robotics.", *Journal of Real Estate Portfolio Management*., vol. 15, no. 2, pp. 115–147, 2007.

[70] Howard, A., Matarić, M.J., Sukhatme, G.S. "An Incremental Self-Deployment Algorithm for Mobile Sensor Networks.", *Autonomous Robots*., vol. 13, no. 2, pp. 113–126, 2002.

[71] Bahgeçi, E., Sahin, E. "Evolving Aggregation Behaviors for Swarm Robotic Systems: A Systematic Case Study.", *EEE Proceedings In Swarm Intelligence Symposium*, pp. 333–340, 2005.

[72] Min, H., Wang, Z. "Group Escape Behavior of Multiple Mobile Robot System by Mimicking Fish Schools.", *IEEE International Conference on Robotics and Biomimetics*, pp. 320–326, 2010.

[73] Ijspeert, A.J., Martinoli, A., Billard, A., Gambardella, L.M. "Collaboration through the Exploitation of Local Interactions in Autonomous Collective Robotics: The Stick Pulling Experiment.", *Autonomous Robots*., vol. 11, no. 2, pp.

149–171, 2001.

[74] Lerman, K., Martinoli, A., Galstyan, A. “A Review of Probabilistic Macroscopic Models for Swarm Robotic Systems.”, *International Workshop on Swarm Robotics*, pp. 143–152, Springer, Berlin, 2005.

[75] Liu, W., Winfield, A.F.T. “Modeling and Optimization of Adaptive Foraging in Swarm Robotic Systems.”, *The International Journal of Robotics Research*., vol. 29, no. 14, pp. 1743–1760, 2010.

[76] Pinciroli, C., O’Grady, R., Christensen, A.L., Birattari, M., Dorigo, M. “Parallel Formation of Differently Sized Groups in a Robotic Swarm.”, *SICE Journal of Control, Measurement, and System Integration*., vol. 52, no. 3, pp. 213–226, 2013.

[77] Hamann, H., Wörn, H. “A Framework of Space-Time Continuous Models for Algorithm Design in Swarm Robotics.”, *Swarm Intelligence*., vol. 2, no. 2–4, pp. 209–239, 2008.

[78] Martinoli, A., Easton, K., Agassounon, W. “Modeling Swarm Robotic Systems: A Case Study in Collaborative Distributed Manipulation.”, *The International Journal of Robotics Research*., vol. 23, no. 4–5, pp. 415–436, 2004.

[79] Massink, M., Brambilla, M., Latella, D., Dorigo, M., Birattari, M. “On the Use of Bio-PEPA for Modelling and Analysing Collective Behaviours in Swarm Robotics.”, *Swarm Intelligence*., vol. 7, no. 2–3, pp. 201–228, 2013.

[80] Xue, S., Zeng, J. “Control over Swarm Robots Search with Swarm Intelligence Principles.”, *Journal of System Simulation*., vol. 20, no. 13, pp. 3449–3454, 2008.

[81] Dorigo, M., Birattari, M. “Ant Colony Optimization.”, *Encyclopedia of machine learning*, pp. 36–39, Springer, Boston, 2011.

[82] Soysal, O., Şahi, E. “A Macroscopic Model for Self-Organized Aggregation in Swarm Robotic Systems.”, *International Workshop on Swarm Robotics*, pp. 27–42, Springer, Berlin, Heidelberg, 2006.

[83] Soysal, O., Sahin, E. “Probabilistic Aggregation Strategies in Swarm Robotic Systems.”, *Proceedings IEEE Swarm Intelligence Symposium.*, pp. 325–332, 2005.

[84] Labella, T.H., Dorigo, M., Deneubourg, J.-L. “Efficiency and Task Allocation in Prey Retrieval.”, *International Workshop on Biologically Inspired Approaches to Advanced Information Technology*, pp. 274–289, Springer, Berlin, Heidelberg, 2004.

[85] King, J., Pretty, R.K., Gosine, R.G. “Coordinated Execution of Tasks in a Multiagent Environment.”, *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*., vol. 33, no. 5, pp. 615–619, 2003.

[86] Rongier, P., Liégeois, A. “Analysis and Prediction of the Behavior of One Class of Multiple Foraging Robots with the Help of Stochastic Petri Nets.”, *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, pp. 143–148, 1999.

[87] Winfield, A.F., Sa, J., Fernández-Gago, M.-C., Dixon, C., Fisher, M. “On Formal

Specification of Emergent Behaviours in Swarm Robotic Systems.", *International journal of advanced robotic systems.*, vol. 2, no. 4, pp. 39, 2005.

[88] Dixon, C., Winfield, A., Fisher, M. "Towards Temporal Verification of Emergent Behaviours in Swarm Robotic Systems.", *Towards autonomous robotic systems*, pp. 336–347, 2011.

[89] de Queiroz, M.H., Cury, J.E.R. "Synthesis and Implementation of Local Modular Supervisory Control for a Manufacturing Cell.", *Sixth International Workshop on Discrete Event Systems*, pp. 377–382, 2002.

[90] Lopes, Y.K., Leal, A.B., Rosso, R.S.U., Harbs, E. "Local Modular Supervisory Implementation in Microcontroller.", *Proceedings of the 9th International Conference of Modeling, Optimization and Simulation*, 2012.

[91] Tsalatsanis, A., Yalcin, A., Valavanis, K.P. "Dynamic Task Allocation in Cooperative Robot Teams.", *Robotica.*, vol. 5, no. 2012, pp. 721–730, 2012.

[92] Tsalatsanis, A., Yalcin, A., Valavanis, K.P. "Optimized Task Allocation in Cooperative Robot Teams.", *17th Mediterranean Conference on Control and Automation*, pp. 270–275, 2009.

[93] Silva, D.B., Santos, E.A., Vieira, A.D., Paula, M.A. de "Application of the Supervisory Control Theory in the Project of a Robot-Centered, Variable Routed System Controller.", *IEEE International Conference on Emerging Technologies and Factory Automation*, pp. 751–758, 2008.

[94] Hoff, N., Wood, R., Nagpal, R. "Distributed Colony-Level Algorithm Switching for Robot Swarm Foraging.", *Distributed Autonomous Robotic Systems*, pp. 417–430, 2013.

[95] Groß, R., Dorigo, M. "Evolution of Solitary and Group Transport Behaviors for Autonomous Robots Capable of Self-Assembling.", *Adaptive Behavior.*, vol. 16, no. 5, pp. 285–305, 2008.

[96] Kube, C.R., Bonabeau, E. "Cooperative Transport by Ants and Robots.", *Robotics and autonomous systems.*, vol. 30, no. 1, pp. 85–101, 2000.

[97] Chen, J., Gauci, M., Groß, R. "A Strategy for Transporting Tall Objects with a Swarm of Miniature Mobile Robots.", *IEEE International Conference on Robotics and Automation*, pp. 863–869, 2013.

[98] Steels, L. "Cooperation between Distributed Agents through Self-Organisation.", *International Workshop on Intelligent Robots and Systems: Towards a New Frontier of Applications*, pp. 8–14, 1987.

[99] Østergaard, E.H., Sukhatme, G.S., Mataric, M.J. "Emergent Bucket Brigading-a Simple Mechanism for Improving Performance in Multi-Robot Constrained-Space Foraging Tasks.", *International Conference on Autonomous Agents*, pp. 4–10, 2001.

[100] Goldberg, D., Mataric, M.J. "Robust Behavior-Based Control for Distributed Multi-Robot Collection Tasks.", University of Southern California Los Angeles United States, 2000.

[101] Fontán, M.S., Matarić, M.J. “A Study of Territoriality : The Role of Critical Mass in Adaptive Task Division.”, *From Animals to Animats IV*, pp. 553–561, 1994.

[102] Vaughan, R.T., Støy, K., Sukhatme, G.S., Matarić, M.J. “Whistling in the Dark: Cooperative Trail Following in Uncertain Localization Space.”, *Proceedings of the fourth international conference on Autonomous agents*, pp. 187–194, 2000.

[103] Sadat, S.A., Vaughan, R.T. “SO-LOST-An Ant-Trail Algorithm for Multi-Robot Navigation with Active Interference Reduction.”, *ALIFE*, pp. 687–693, 2010.

[104] Hoff, N.R., Sagoff, A., Wood, R.J., Nagpal, R. “Two Foraging Algorithms for Robot Swarms Using Only Local Communication.”, *IEEE International Conference on Robotics and Biomimetics*, pp. 123–130, 2010.

[105] Sumpter, D., Pratt, S. “A Modelling Framework for Understanding Social Insect Foraging.”, *Behavioral Ecology and Sociobiology*., vol. 53, no. 3, pp. 131–144, 2003.

[106] Sugawara, K., Sano, M. “Cooperative Acceleration of Task Performance: Foraging Behavior of Interacting Multi-Robots System.”, *Physica D: Nonlinear Phenomena*., vol. 100, no. 3–4, pp. 343–354, 1997.

[107] Sugawara, K., Sano, M., Yoshihara, I., Abe, K., Watanabe, T. “Foraging Behaviour of Multi-Robot System and Emergence of Swarm Intelligence.”, *IEEE International Conference on Systems, Man, and Cybernetics*, pp. 257–262, 1999.

[108] Amorim, P. “Modeling Ant Foraging: A Chemotaxis Approach with Pheromones and Trail Formation.”, *Journal of Theoretical Biology*., vol. 385, pp. 160–173, 2015.

[109] Cassandras, C.G., Lafortune, S. Introduction to discrete event systems., Springer Science & Business Media, 2009.

[110] Ramadge, P.J., Wonham, W.M. “Supervisory Control of a Class of Discrete Event Processes.”, *SIAM Journal on Control and Optimization*., vol. 25, no. 1, pp. 206–230, 1987.

[111] Lopes, Y.K., Trenkwalder, S.M., Leal, A.B., Dodd, T.J., Groß, R. “Probabilistic Supervisory Control Theory (PSCT) Applied to Swarm Robotics.”, *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pp. 1395–1403, 2016.

[112] Brandin, B. a, Wonham, W.M. “The Supervisory Control of Timed DES.”, *IEEE Transactions on Automatic Control*., vol. 39, no. 2, pp. 329–342, 1994.

[113] Wonham, W., Cai, K. Supervisory Control of Discrete-Event Systems, v.20170901., 2017.

[114] Limited, R.T. “Rahal Tec,” [Online] 2018, http://rahalco.co.uk/ (Accessed: 1 December 2018).

[115] Castello, E., Yamamoto, T., Libera, F.D., Liu, W., Winfield, A.F.T., Nakamura, Y., Ishiguro, H. Adaptive foraging for simulated and real robotic swarms: the dynamical response threshold approach., 2016.

[116] Liu, W., Winfield, A.F.T. “Microprocessors and Microsystems Open-Hardware e-Puck Linux Extension Board for Experimental Swarm Robotics Research.”, *Microprocessors and Microsystems*., vol. 35, no. 1, pp. 60–67, 2011.

[117] Gauci, M., Chen, J., Li, W., Dodd, T.J., Groß, R. “Self-Organized Aggregation without Computation.”, *The International Journal of Robotics Research*., vol. 33, no. 8, pp. 1145–1161, 2014.

[120] Sakthivelmurugan, E., Senthilkumar, G., Prithiviraj, K.G., R, T.D.K. “Foraging Behavior Analysis of Swarm Robotics System.”, *MATEC Web of Conferences*, p. 01013, EDP Sciences, 2018.

[121] Lu, Q., Hecker, J.P., Moses, M.E. “Multiple-Place Swarm Foraging with Dynamic Depots.”, *Autonomous Robots*., pp. 1–18, 2018.

[122] B, J.W., Timmis, J., Tyrrell, A. Towards Autonomous Robotic Systems., Springer International Publishing, 2018.

[123] Lima, D.A., Oliveira, G.M.B. “A Cellular Automata Ant Memory Model of Foraging in a Swarm of Robots.”, *Applied Mathematical Modelling*., vol. 47, pp. 551–572, 2017.

[124] Lu, Q., Hecker, J.P., Moses, M.E. “The MPFA: A Multiple-Place Foraging Algorithm for Biologically-Inspired Robot Swarms.”, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3815–3821, 2016.

[125] Wenguo Liu, Winfield, A.F.T., Jin Sa, Jie Chen, Lihua Dou “Towards Energy Optimization: Emergent Task Allocation in a Swarm of Foraging Robots.”, *Adaptive Behavior*., vol. 15, no. 3, pp. 289–305, 2007.

[126] Hecker, J.P., Moses, M.E. “Beyond Pheromones: Evolving Error-Tolerant, Flexible, and Scalable Ant-Inspired Robot Swarms.”, *Swarm Intelligence*., vol. 9, no. 1, pp. 43–70, 2015.

[127] Campo, A., Gutiérrez, Á., Nouyan, S., Pinciroli, C., Longchamp, V., Garnier, S., Dorigo, M. “Artificial Pheromone for Path Selection by a Foraging Swarm of Robots.”, *Biological Cybernetics*., vol. 103, no. 5, pp. 339–352, 2010.

[128] Niimi, A., Kawakami, C., Fukuda, K., Sugino, R., Itami, S., Miyake, S. “Development of the Compact Swarm Robot System Based on Pheromone Communication -An Improvement Approach for Position Measurement System-.”, *IEEE 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, pp. 1439–1442, 2017.

[129] Fujisawa, R., Imamura, H., Matsuno, F. Cooperative transportation by swarm robots using pheromone communication., , Berlin, 2013.

## Abstract

Swarm robotic (SR) draws inspiration from nature and is emerged as a synthesis of swarm intelligence and robotic. Due to special properties like flexibility, robustness, and scalability, swarm robotic has several advantages in comparison to other multi robot systems. On the other hand, SR properties and characteristics expand swarm robotic applications domain to dangerous, industrial and military environments. Hence, scientists had a special interest in swarm robotic in the last years. Lack of common control software makes it difficult to analyze, maintain or verify swarm robotic systems. Using formal methods for SR modeling and design almost contributes to overcoming these problems. However, there is no guarantee that the implementation matches the system requirements and specifications in formal methods too. The proposed approach is using supervisory control theory (SCT) as system modeling and design tool. This theory addresses the problem of the formal methods with automatic generation of control codes.

One of the well-known tasks of swarm robotic is foraging inspired by ant's seed searching. Foraging as a combination of multiple subtasks like collective searching, collective transport, and task allocation is considered as one of the most complex tasks of swarm robotic. In this dissertation, a foraging task is designed and implemented. For this purpose, a comprehensive tool is proposed which gets system behavior and requirements as generators and produces desired control code automatically. The proposed tool uses a more powerful version of SCT named ptSCT that adds probability and time elements to SCT. Demonstrating the power of ptSCT, two tasks namely obstacle avoidance and synchronization of robots are designed, and implemented using both SCT and proposed ptSCT. For evaluation of implemented foraging, more than 2400 experiments are heavily tested on ARGoS simulation environment with E-puck robot. Effect of the number of targets, obstacles, and robots are evaluated. Moreover, the effect of resting of robots on the system performance is also considered. The experimental results declare the advantages of the ptSCT, in terms of simplicity, reusability, and automatic code generation. Therefore, ptSCT can tackle other SR tasks modeling.



**Keywords:** Swarm Robotic, Swarm Intelligence, Supervisory Control Theory, Foraging, Formal Modelling Approaches

**Shahrood University of Technology**

**Faculty of Computer Engineering**

**PhD Dissertation in Artificial Intelligence**

# Foraging In Swarm Robotic Using Supervisory Control Theory (SCT)

By: **Faezeh Mirzaei**

Supervisor:

**Dr. Ali Akbar Pouyan**

Advisor:

**Dr. Saideh Ferdowsi**

**September 2019**