Memetic algorithms for optimal task allocation in multi-robot systems for inspection problems with cooperative tasks

A Summary

Amrit Kochar 2014B4A7821P

The Problem: MRTA Problem

- Stands for multi robot task allocation problem. We try to efficiently allocate all tasks at hand to various robots with an aim to optimize our requirements.
- In this paper, tasks may require one or two robots to work together.
- The authors have used a memetic algorithm which combines GA for global optimization and two local search heuristics.

Target Area

- ➢ The authors have used a tank farm of a petroleum refinery as a target for this paper.
- ➢ The algorithm has been tested on two different layouts of tanks, ‘tank rows’ and ‘tank islands’.
- ➢ A bold dashed line defines the two inspection positions of a two-robot task that should be simultaneously carried out by two robots.

Fig. 1 PCK refinery (© PCK Raffinerie GmbH)

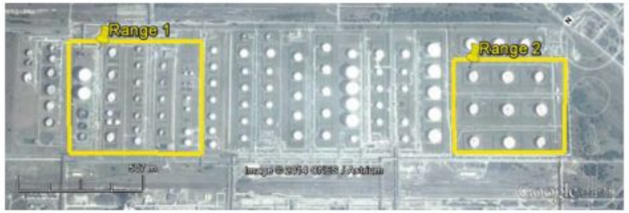


Fig. 2 Tank farm of PCK refinery (Source: Google Earth)

Two Layouts

- ➢ Tank Rows

  80 single-robot tasks
  5 two-robot tasks
  3 mobile robots

- ➢ Tank Islands

  90 single-robot tasks
  5 two-robot tasks
  3 mobile robots

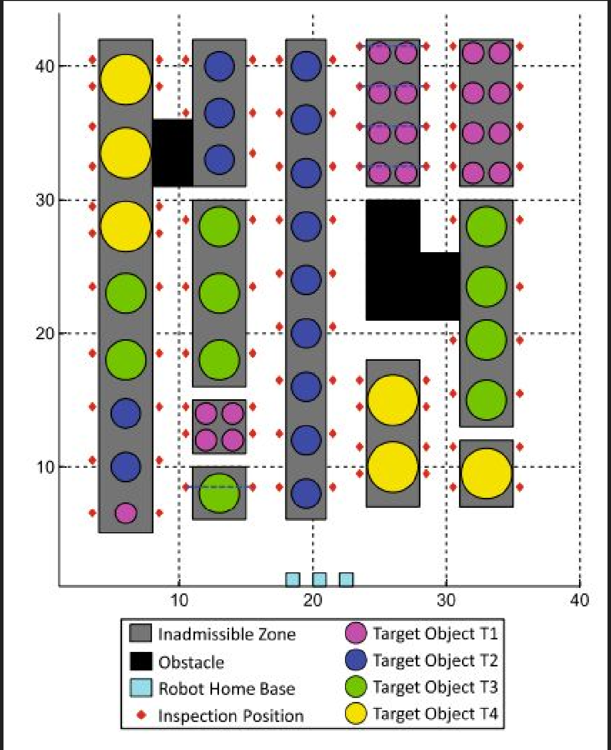


Fig. 3 Inspection area and tasks of "tank rows" scenario

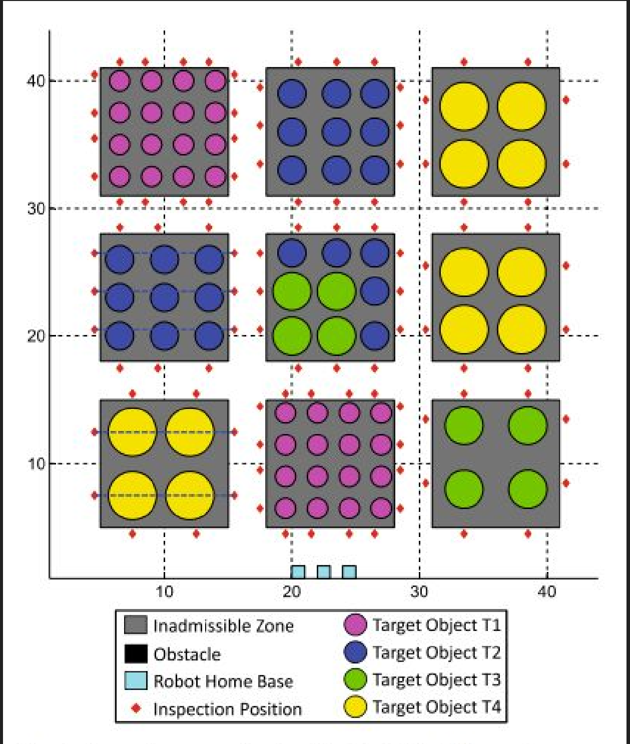


Fig. 4 Inspection area and tasks of the "tank islands" scenario

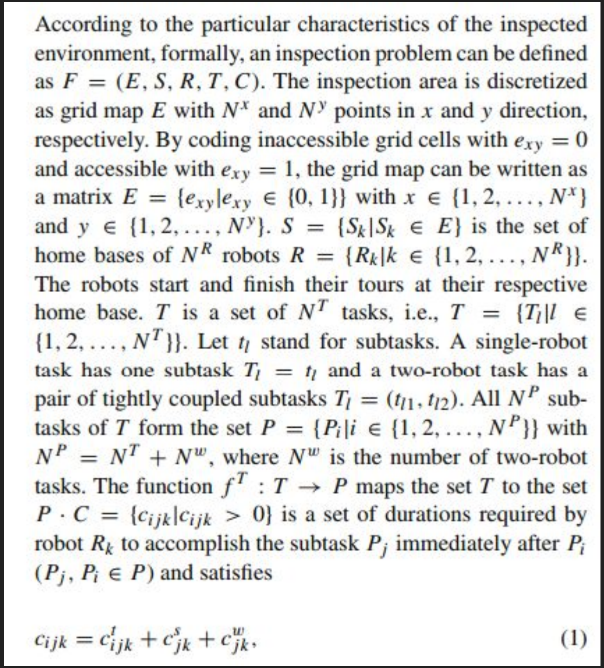
Assumptions

1. The robots work in a known static environment
2. The problem includes many more single-robot tasks than two-robot tasks
3. The number of robots is much smaller than the number of tasks
4. No changing or charging battery is required after every robot has left its home bases.

Violation of (1) will practically be answered by tactical rescheduling. (2)–(3) is natural for the intended problem class. Changing of batteries (4) could easily be considered as a task if lengthy missions are to be planned.

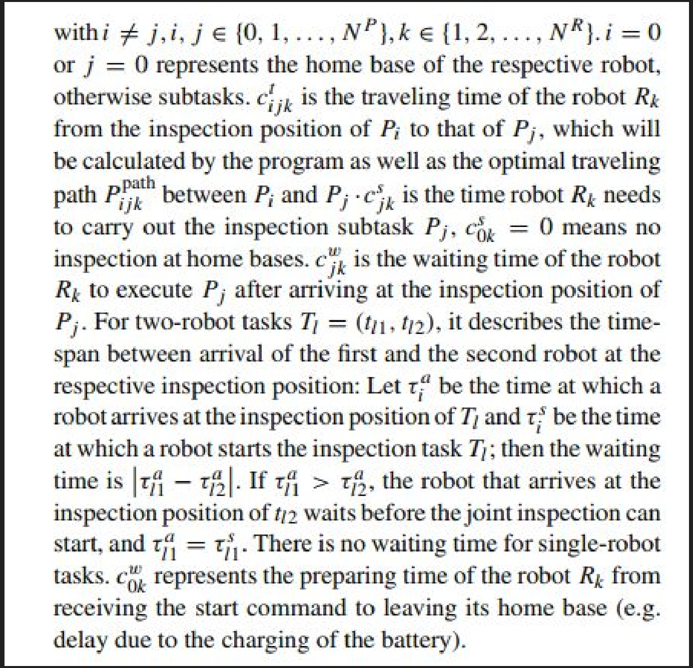
Mathematical Model

➢ Tasks and Resources



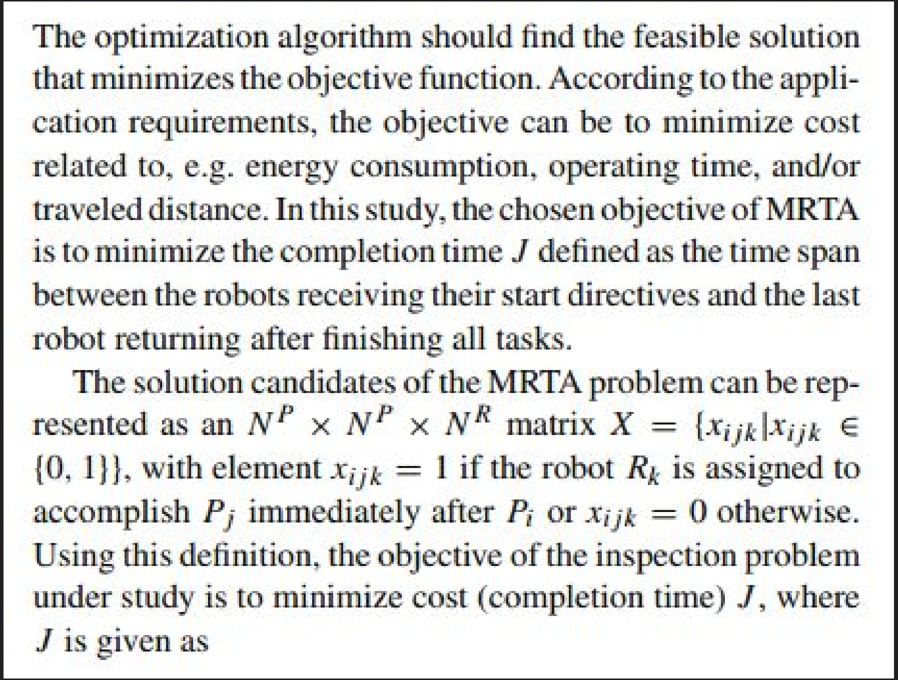
According to the particular characteristics of the inspected environment, formally, an inspection problem can be defined as $F = (E, S, R, T, C)$. The inspection area is discretized as grid map E with N^x and N^y points in x and y direction, respectively. By coding inaccessible grid cells with $e_{xy} = 0$ and accessible with $e_{xy} = 1$, the grid map can be written as a matrix $E = \{e_{xy}|e_{xy} \in \{0, 1\}\}$ with $x \in \{1, 2, \dots, N^x\}$ and $y \in \{1, 2, \dots, N^y\}$. $S = \{S_k|S_k \in E\}$ is the set of home bases of N^R robots $R = \{R_k|k \in \{1, 2, \dots, N^R\}\}$. The robots start and finish their tours at their respective home base. T is a set of N^T tasks, i.e., $T = \{T_l|l \in \{1, 2, \dots, N^T\}\}$. Let t_l stand for subtasks. A single-robot task has one subtask $T_l = t_l$ and a two-robot task has a pair of tightly coupled subtasks $T_l = (t_{l1}, t_{l2})$. All N^P sub-tasks of T form the set $P = \{P_i|i \in \{1, 2, \dots, N^P\}\}$ with $N^P = N^T + N^w$, where N^w is the number of two-robot tasks. The function $f^T : T \to P$ maps the set T to the set $P \cdot C = \{c_{ijk}|c_{ijk} > 0\}$ is a set of durations required by robot R_k to accomplish the subtask P_j immediately after P_i ($P_j, P_i \in P$) and satisfies

$$c_{ijk} = c^t_{ijk} + c^s_{jk} + c^w_{jk}, \qquad (1)$$



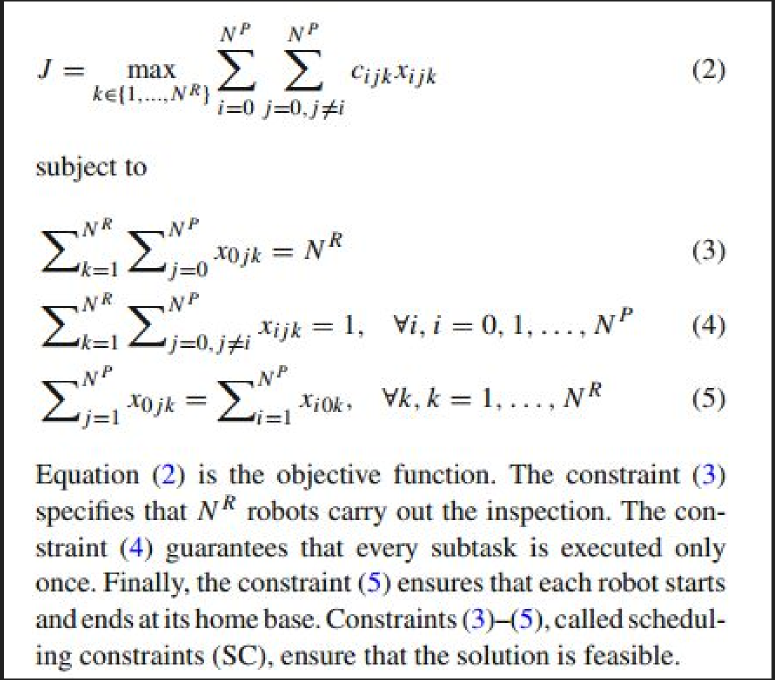
with $i \neq j, i, j \in \{0, 1, \dots, N^P\}, k \in \{1, 2, \dots, N^R\}$. $i = 0$ or $j = 0$ represents the home base of the respective robot, otherwise subtasks. c^t_{ijk} is the traveling time of the robot R_k from the inspection position of P_i to that of P_j, which will be calculated by the program as well as the optimal traveling path P^{path}_{ijk} between P_i and $P_j \cdot c^s_{jk}$ is the time robot R_k needs to carry out the inspection subtask P_j, $c^s_{0k} = 0$ means no inspection at home bases. c^w_{jk} is the waiting time of the robot R_k to execute P_j after arriving at the inspection position of P_j. For two-robot tasks $T_l = (t_{l1}, t_{l2})$, it describes the time-span between arrival of the first and the second robot at the respective inspection position: Let τ^a_i be the time at which a robot arrives at the inspection position of T_l and τ^s_i be the time at which a robot starts the inspection task T_l; then the waiting time is $\left|\tau^a_{l1} - \tau^a_{l2}\right|$. If $\tau^a_{l1} > \tau^a_{l2}$, the robot that arrives at the inspection position of t_{l2} waits before the joint inspection can start, and $\tau^a_{l1} = \tau^s_{l1}$. There is no waiting time for single-robot tasks. c^w_{0k} represents the preparing time of the robot R_k from receiving the start command to leaving its home base (e.g. delay due to the charging of the battery).

Objective function and constraints



The optimization algorithm should find the feasible solution that minimizes the objective function. According to the application requirements, the objective can be to minimize cost related to, e.g. energy consumption, operating time, and/or traveled distance. In this study, the chosen objective of MRTA is to minimize the completion time J defined as the time span between the robots receiving their start directives and the last robot returning after finishing all tasks.

The solution candidates of the MRTA problem can be represented as an $N^P \times N^P \times N^R$ matrix $X = \{x_{ijk} | x_{ijk} \in \{0, 1\}\}$, with element $x_{ijk} = 1$ if the robot R_k is assigned to accomplish P_j immediately after P_i or $x_{ijk} = 0$ otherwise. Using this definition, the objective of the inspection problem under study is to minimize cost (completion time) J, where J is given as



$$J = \max_{k \in \{1,\ldots,N^R\}} \sum_{i=0}^{N^P} \sum_{j=0, j \neq i}^{N^P} c_{ijk} x_{ijk} \tag{2}$$

subject to

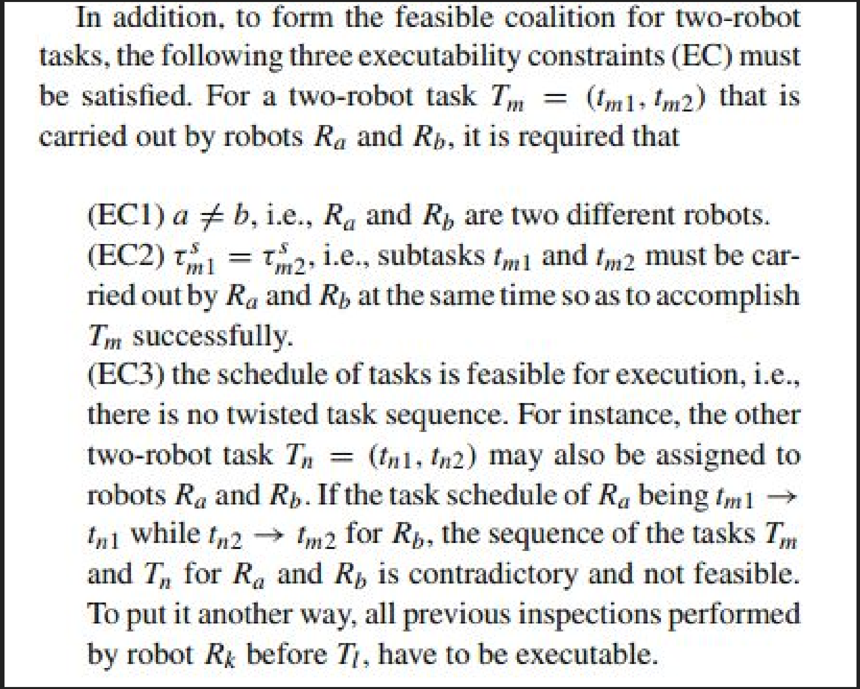
$$\sum_{k=1}^{N^R} \sum_{j=0}^{N^P} x_{0jk} = N^R \tag{3}$$

$$\sum_{k=1}^{N^R} \sum_{j=0, j \neq i}^{N^P} x_{ijk} = 1, \quad \forall i, i = 0, 1, \ldots, N^P \tag{4}$$

$$\sum_{j=1}^{N^P} x_{0jk} = \sum_{i=1}^{N^P} x_{i0k}, \quad \forall k, k = 1, \ldots, N^R \tag{5}$$

Equation (2) is the objective function. The constraint (3) specifies that N^R robots carry out the inspection. The constraint (4) guarantees that every subtask is executed only once. Finally, the constraint (5) ensures that each robot starts and ends at its home base. Constraints (3)–(5), called scheduling constraints (SC), ensure that the solution is feasible.

Continued



In addition, to form the feasible coalition for two-robot tasks, the following three executability constraints (EC) must be satisfied. For a two-robot task $T_m = (t_{m1}, t_{m2})$ that is carried out by robots R_a and R_b, it is required that

(EC1) $a \neq b$, i.e., R_a and R_b are two different robots.
(EC2) $\tau^s_{m1} = \tau^s_{m2}$, i.e., subtasks t_{m1} and t_{m2} must be carried out by R_a and R_b at the same time so as to accomplish T_m successfully.
(EC3) the schedule of tasks is feasible for execution, i.e., there is no twisted task sequence. For instance, the other two-robot task $T_n = (t_{n1}, t_{n2})$ may also be assigned to robots R_a and R_b. If the task schedule of R_a being $t_{m1} \to t_{n1}$ while $t_{n2} \to t_{m2}$ for R_b, the sequence of the tasks T_m and T_n for R_a and R_b is contradictory and not feasible. To put it another way, all previous inspections performed by robot R_k before T_l, have to be executable.

The Memetic Algorithm

- ➢ Memetic algorithms (MA) basically represents a synergy of evolutionary or any population-based approach with separate individual learning or local improvement procedures for problem search.
- ➢ The inspection time (c^s_{jk}) required by R_k to carry out P_j must be predefined according to the inspection method *(Bonow and Kroll 2013)*. In the step "path planning", the optimal path (P^{path}_{ijk}) and traveling time (c^t_{ijk}) between two inspection positions of any two subtasks is calculated by the A* algorithm *(Liu and Kroll 2012a)*.

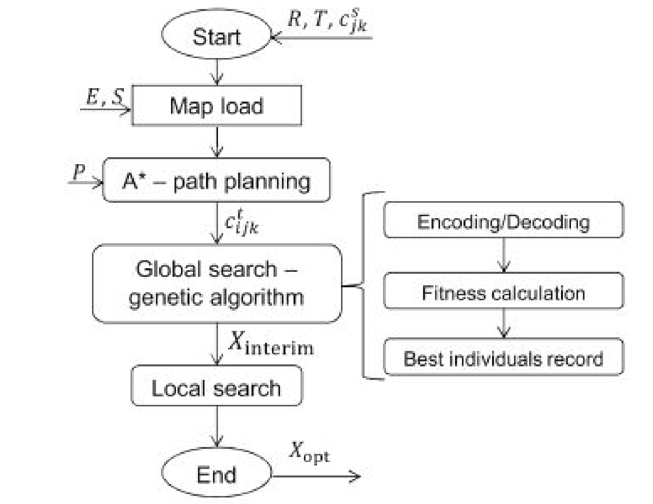
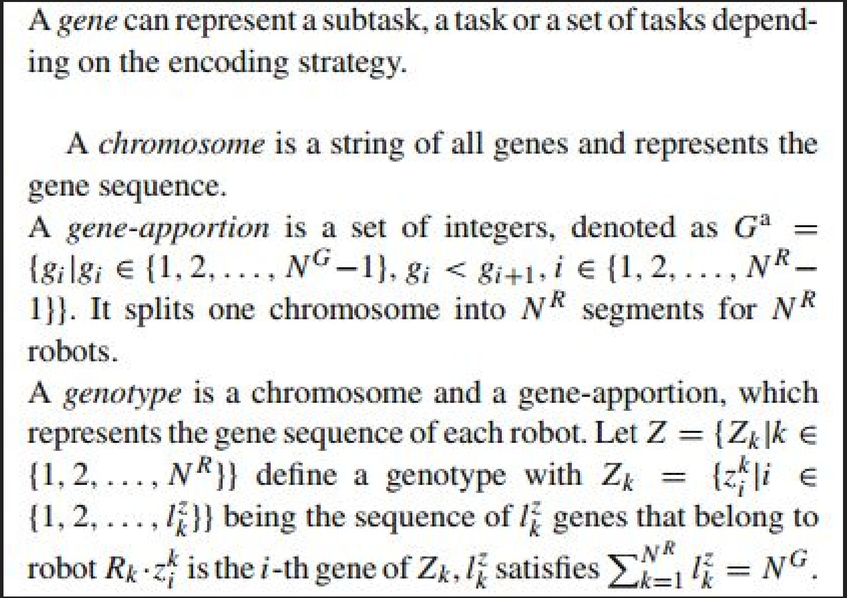


Fig. 5 Flow chart of the proposed memetic algorithm

Solution Representation

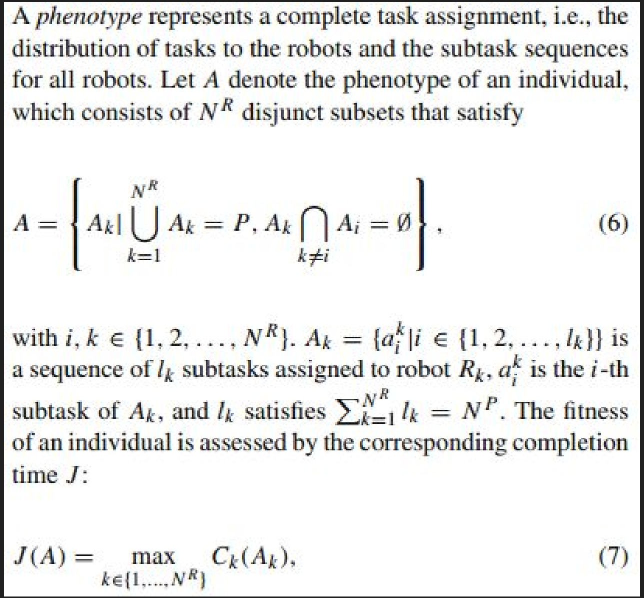


A *gene* can represent a subtask, a task or a set of tasks depending on the encoding strategy.

A *chromosome* is a string of all genes and represents the gene sequence.

A *gene-apportion* is a set of integers, denoted as $G^a = \{g_i | g_i \in \{1, 2, \ldots, N^G - 1\}, g_i < g_{i+1}, i \in \{1, 2, \ldots, N^R - 1\}\}$. It splits one chromosome into N^R segments for N^R robots.

A *genotype* is a chromosome and a gene-apportion, which represents the gene sequence of each robot. Let $Z = \{Z_k | k \in \{1, 2, \ldots, N^R\}\}$ define a genotype with $Z_k = \{z_i^k | i \in \{1, 2, \ldots, l_k^z\}\}$ being the sequence of l_k^z genes that belong to robot $R_k \cdot z_i^k$ is the i-th gene of Z_k, l_k^z satisfies $\sum_{k=1}^{N^R} l_k^z = N^G$.

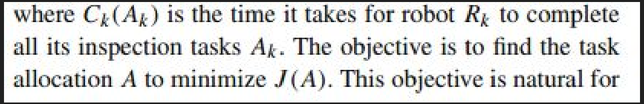


A *phenotype* represents a complete task assignment, i.e., the distribution of tasks to the robots and the subtask sequences for all robots. Let A denote the phenotype of an individual, which consists of N^R disjunct subsets that satisfy

$$A = \left\{ A_k | \bigcup_{k=1}^{N^R} A_k = P, A_k \bigcap_{k \neq i} A_i = \emptyset \right\}, \tag{6}$$

with $i, k \in \{1, 2, \ldots, N^R\}$. $A_k = \{a_i^k | i \in \{1, 2, \ldots, l_k\}\}$ is a sequence of l_k subtasks assigned to robot R_k, a_i^k is the i-th subtask of A_k, and l_k satisfies $\sum_{k=1}^{N^R} l_k = N^P$. The fitness of an individual is assessed by the corresponding completion time J:

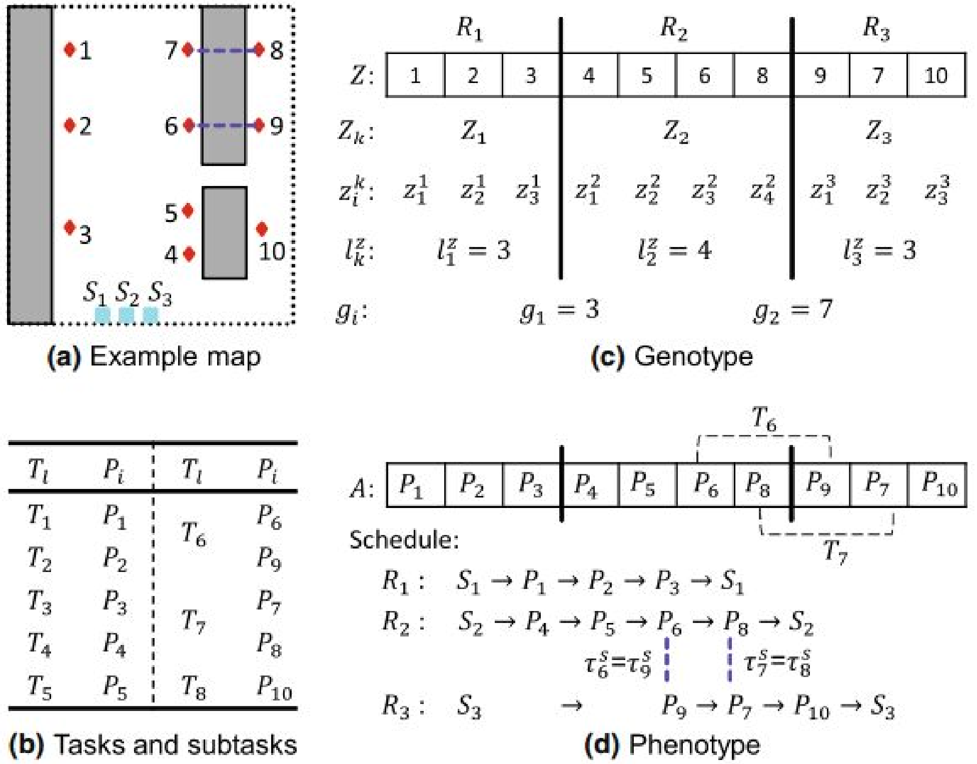
$$J(A) = \max_{k \in \{1, \ldots, N^R\}} C_k(A_k), \tag{7}$$



where $C_k(A_k)$ is the time it takes for robot R_k to complete all its inspection tasks A_k. The objective is to find the task allocation A to minimize $J(A)$. This objective is natural for

An Example

Fig. 6 Coding example with single-robot tasks ($T_1 - T_5$, T_8) and two-robot tasks (T_6, T_7)



1
2
3
4
5
6
7
8
9
10
S_1 S_2 S_3
(a) Example map
R_1 R_2 R_3
Z: 1 2 3 4 5 6 8 9 7 10
Z_k: Z_1 Z_2 Z_3
z_i^k: z_1^1 z_2^1 z_3^1 z_1^2 z_2^2 z_3^2 z_4^2 z_1^3 z_2^3 z_3^3
l_k^z: $l_1^z = 3$ $l_2^z = 4$ $l_3^z = 3$
g_i: $g_1 = 3$ $g_2 = 7$
(c) Genotype
T_6
A: P_1 P_2 P_3 P_4 P_5 P_6 P_8 P_9 P_7 P_{10}
T_7
Schedule:
R_1 : $S_1 \rightarrow P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow S_1$
R_2 : $S_2 \rightarrow P_4 \rightarrow P_5 \rightarrow P_6 \rightarrow P_8 \rightarrow S_2$
$\tau_6^s = \tau_9^s$
$\tau_7^s = \tau_8^s$
R_3 : $S_3 \rightarrow P_9 \rightarrow P_7 \rightarrow P_{10} \rightarrow S_3$
(d) Phenotype

T_l	P_i	T_l	P_i
T_1	P_1	T_6	P_6
T_2	P_2		P_9
T_3	P_3	T_7	P_7
T_4	P_4		P_8
T_5	P_5	T_8	P_{10}

(b) Tasks and subtasks

The Genetic Algorithm

- **Design parameter and coding decision:** population size, termination criterion and coding strategy is chosen.
- **Initial population:** It is randomly generated (each genotype).
- We then **decode** the genotypes for **fitness calculation** while **repairing infeasible solutions** *(discussed ahead)* in between the two processes.
- We **store N^b best individuals** of the current population in B.
- **New population generation**. *(discussed ahead)*
- Repeat until termination criterion is met.

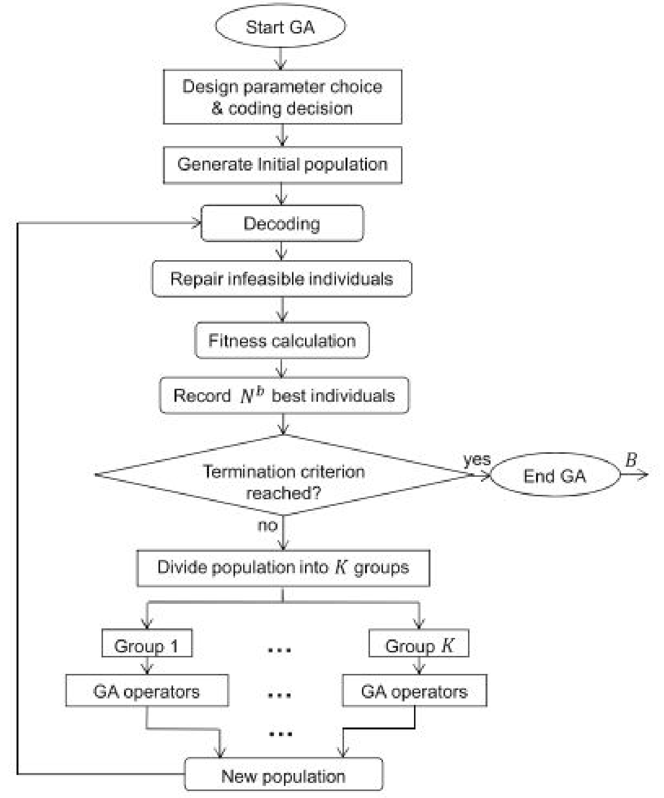
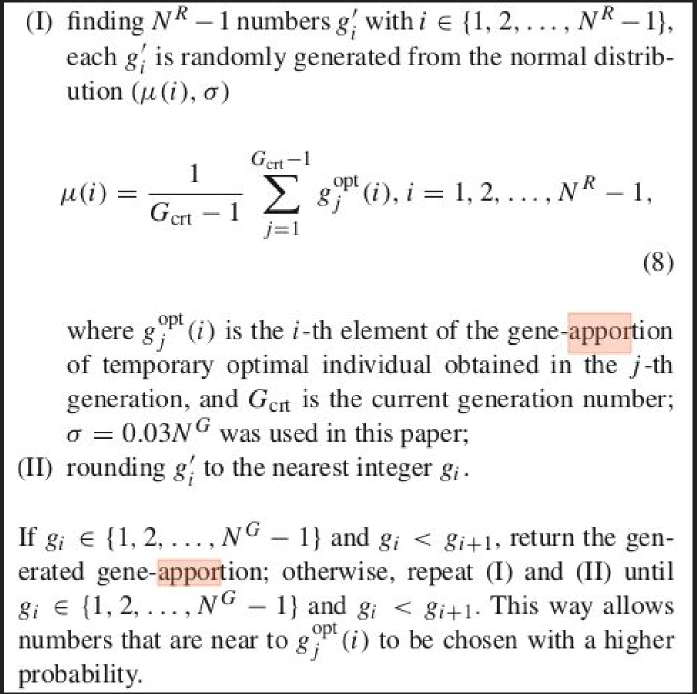

Start GA
Design parameter choice & coding decision
Generate Initial population
Decoding
Repair infeasible individuals
Fitness calculation
Record N^b best individuals
Termination criterion reached?
yes
End GA
B
no
Divide population into K groups
Group 1
...
Group K
GA operators
...
GA operators
...
New population

Fig. 7 Flow chart of the proposed genetic algorithm

New Population Generation

- Population is divided into K non overlapping groups with equal size and operators are applied on these individually.
- Swap, insertion, inversion, and displacement mutation applied to N$_{pop}$/5K times to the chromosome of the best individual in each group with probablity=1.
- The N$_{pop}$/5K best individuals are reproduced unchanged.
- The new gene apportion is determined by ---->



(I) finding $N^R - 1$ numbers g'_i with $i \in \{1, 2, \ldots, N^R - 1\}$, each g'_i is randomly generated from the normal distribution $(\mu(i), \sigma)$

$$\mu(i) = \frac{1}{G_{\text{crt}} - 1} \sum_{j=1}^{G_{\text{crt}}-1} g_j^{\text{opt}}(i), \; i = 1, 2, \ldots, N^R - 1, \tag{8}$$

where $g_j^{\text{opt}}(i)$ is the i-th element of the gene-apportion of temporary optimal individual obtained in the j-th generation, and G_{crt} is the current generation number; $\sigma = 0.03N^G$ was used in this paper;

(II) rounding g'_i to the nearest integer g_i.

If $g_i \in \{1, 2, \ldots, N^G - 1\}$ and $g_i < g_{i+1}$, return the generated gene-apportion; otherwise, repeat (I) and (II) until $g_i \in \{1, 2, \ldots, N^G - 1\}$ and $g_i < g_{i+1}$. This way allows numbers that are near to $g_j^{\text{opt}}(i)$ to be chosen with a higher probability.

Termination Criterion

- The GA terminates when the number of generations reaches a particular value.
- At the end, the set B (consisting of the best solutions from each generation) will be transferred to the local search.

Repair Schemes

- **Coalition infeasible solutions:** For a 2 robot task assigned to a single robot, we randomly exchange one of the subtasks of the task in hand with one of the subtasks of $A' = A/A_k$.

- **Schedule infeasible solutions:** For two 2 robot tasks that are assigned to two unique robots, one of the subtasks of R_k is exchanged with the other one of the same robot.

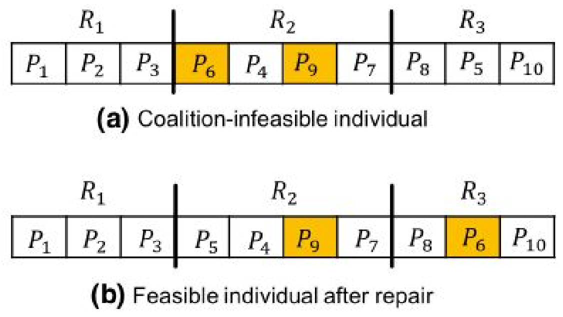

R_1
R_2
R_3
P_1 P_2 P_3 P_6 P_4 P_9 P_7 P_8 P_5 P_{10}
(a) Coalition-infeasible individual
R_1
R_2
R_3
P_1 P_2 P_3 P_5 P_4 P_9 P_7 P_8 P_6 P_{10}
(b) Feasible individual after repair

Fig. 8 Example of coalition-infeasible individual (**a**) and repair result (**b**)

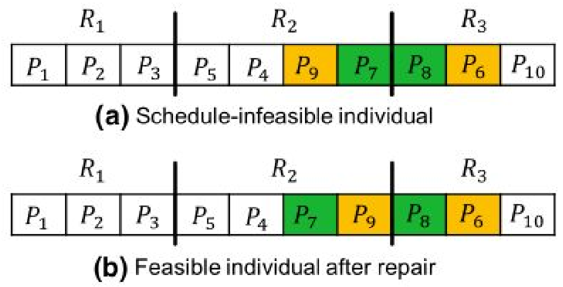
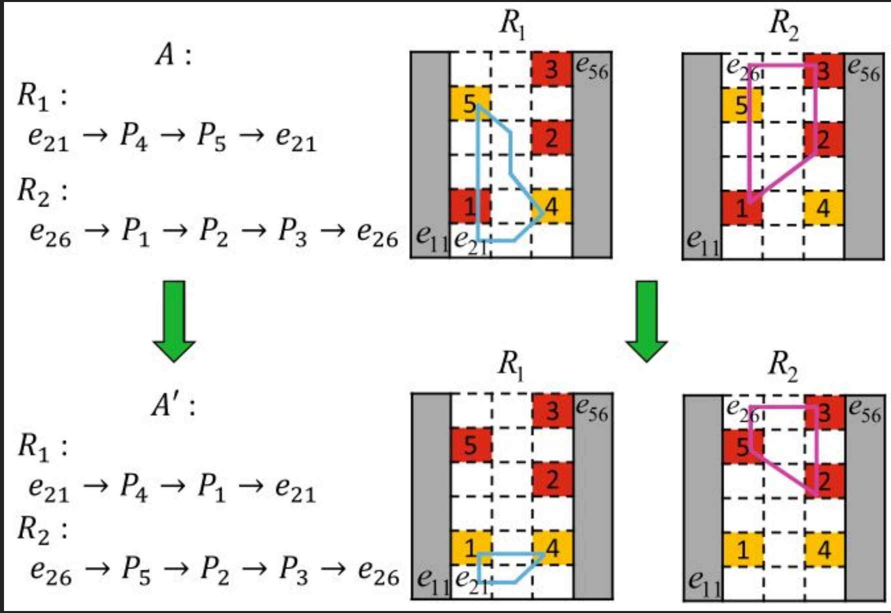

R_1
R_2
R_3
P_1 P_2 P_3 P_5 P_4 P_9 P_7 P_8 P_6 P_{10}
(a) Schedule-infeasible individual
R_1
R_2
R_3
P_1 P_2 P_3 P_5 P_4 P_7 P_9 P_8 P_6 P_{10}
(b) Feasible individual after repair

Fig. 9 Example of schedule-infeasible individual (**a**) and repair result (**b**)

Local Search Method

- Applied to the individuals of set B.
- We remove duplicates in B before starting off.
- **Pass-by Insertion:** We will insert a subtask P_m which the robot R_k passes by without inspecting the object in the schedule of R_k. This change is accepted only if it improves the performance.

- **2-nearest-neighbour swapping:** Two neighbouring subtasks a_j^k and a_{j+1}^k ($j = 1,2,....,l_k-1$). The new schedule is accepted if it improves performance. The local search runs for ($j = 1,2,....,l_k-1$)



A :
R_1 :
$e_{21} \to P_4 \to P_5 \to e_{21}$
R_2 :
$e_{26} \to P_1 \to P_2 \to P_3 \to e_{26}$
A' :
R_1 :
$e_{21} \to P_4 \to P_1 \to e_{21}$
R_2 :
$e_{26} \to P_5 \to P_2 \to P_3 \to e_{26}$
R_1
R_2
e_{11}
e_{21}
e_{26}
e_{56}

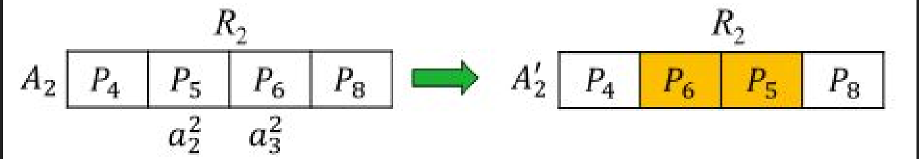


Fig. 11 Example of 2-nearest-neighbor swapping

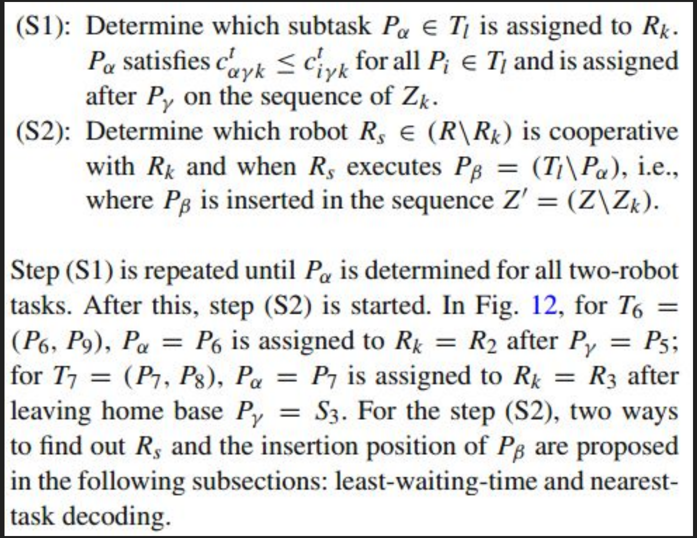
Coding Strategies

5.1: Subtask-based (SB) coding:

- ➢ Each subtask, P_i is encoded as one gene. ($N^G=N^P$).
- ➢ No solution would be missed given infinite time.

5.2: Task-based (TB) coding:

- ➢ Each task, T_i is encoded as one gene. ($N^G=N^T$).
- ➢ Decoded by separation of single and two robot tasks.
- ➢ Single robot tasks are mapped by the function f^T.
- ➢ Each 2 robot task carried out by R_k is decoded in two steps --->



(S1): Determine which subtask $P_\alpha \in T_l$ is assigned to R_k. P_α satisfies $c^t_{\alpha\gamma k} \le c^t_{i\gamma k}$ for all $P_i \in T_l$ and is assigned after P_γ on the sequence of Z_k.

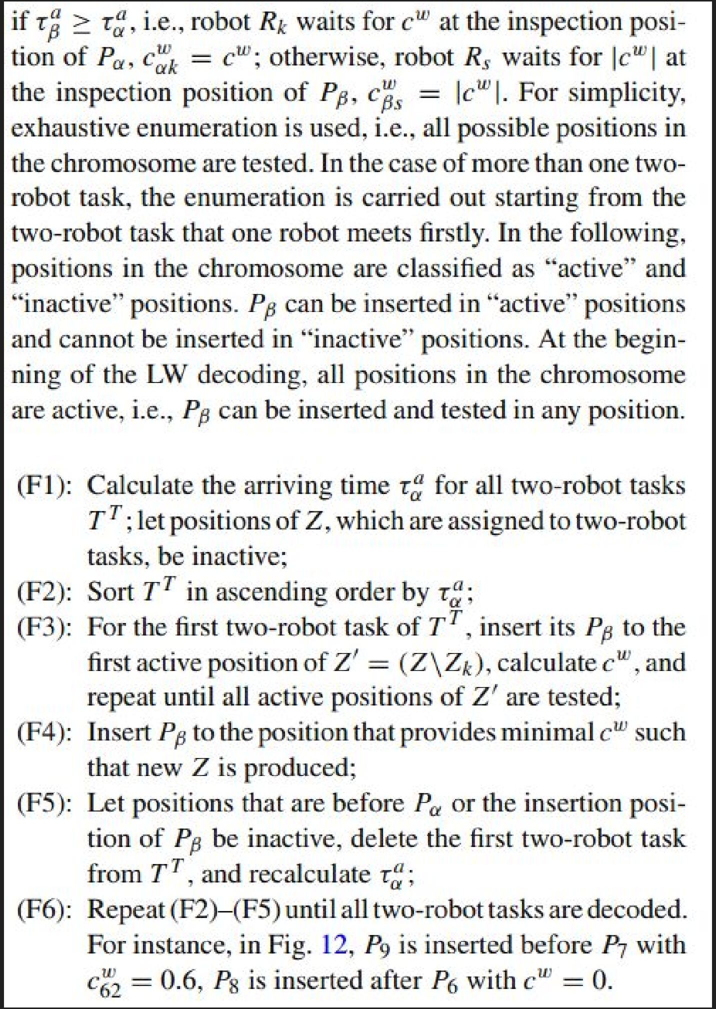
(S2): Determine which robot $R_s \in (R \backslash R_k)$ is cooperative with R_k and when R_s executes $P_\beta = (T_l \backslash P_\alpha)$, i.e., where P_β is inserted in the sequence $Z' = (Z \backslash Z_k)$.

Step (S1) is repeated until P_α is determined for all two-robot tasks. After this, step (S2) is started. In Fig. 12, for $T_6 = (P_6, P_9)$, $P_\alpha = P_6$ is assigned to $R_k = R_2$ after $P_\gamma = P_5$; for $T_7 = (P_7, P_8)$, $P_\alpha = P_7$ is assigned to $R_k = R_3$ after leaving home base $P_\gamma = S_3$. For the step (S2), two ways to find out R_s and the insertion position of P_β are proposed in the following subsections: least-waiting-time and nearest-task decoding.

Coding Strategies

5.2.1 Least Waiting Time(LW) decoding:

- The insertion position for P_B is determined such that the waiting time c^w is minimal.



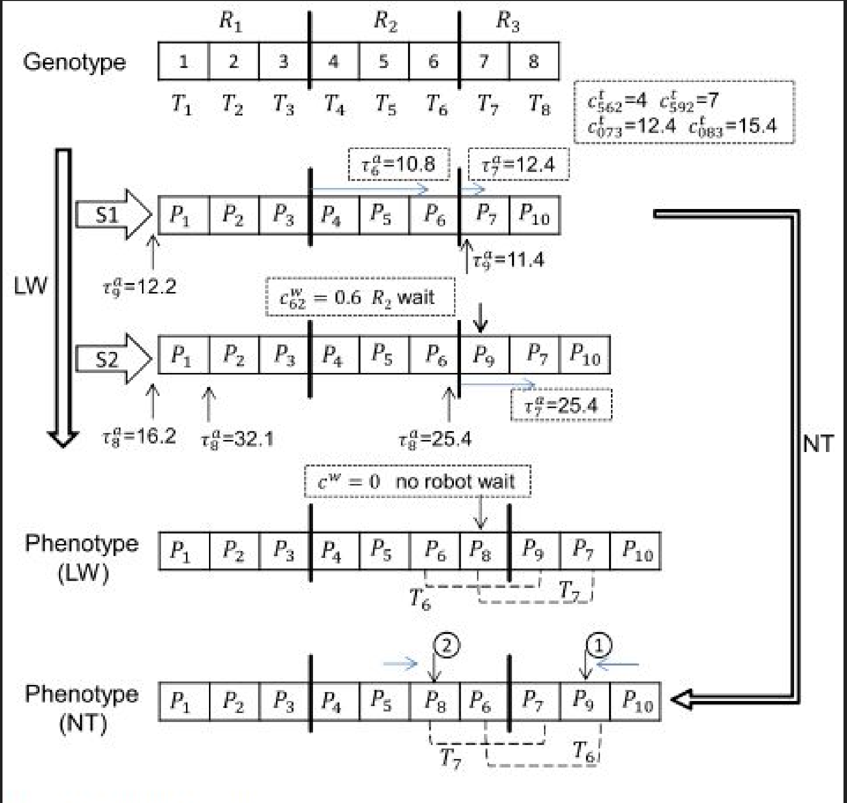
if $\tau^a_\beta \geq \tau^a_\alpha$, i.e., robot R_k waits for c^w at the inspection position of P_α, $c^w_{\alpha k} = c^w$; otherwise, robot R_s waits for $|c^w|$ at the inspection position of P_β, $c^w_{\beta s} = |c^w|$. For simplicity, exhaustive enumeration is used, i.e., all possible positions in the chromosome are tested. In the case of more than one two-robot task, the enumeration is carried out starting from the two-robot task that one robot meets firstly. In the following, positions in the chromosome are classified as "active" and "inactive" positions. P_β can be inserted in "active" positions and cannot be inserted in "inactive" positions. At the beginning of the LW decoding, all positions in the chromosome are active, i.e., P_β can be inserted and tested in any position.

(F1): Calculate the arriving time τ^a_α for all two-robot tasks T^T; let positions of Z, which are assigned to two-robot tasks, be inactive;
(F2): Sort T^T in ascending order by τ^a_α;
(F3): For the first two-robot task of T^T, insert its P_β to the first active position of $Z' = (Z \backslash Z_k)$, calculate c^w, and repeat until all active positions of Z' are tested;
(F4): Insert P_β to the position that provides minimal c^w such that new Z is produced;
(F5): Let positions that are before P_α or the insertion position of P_β be inactive, delete the first two-robot task from T^T, and recalculate τ^a_α;
(F6): Repeat (F2)–(F5) until all two-robot tasks are decoded. For instance, in Fig. 12, P_9 is inserted before P_7 with $c^w_{62} = 0.6$, P_8 is inserted after P_6 with $c^w = 0$.

Coding Strategies

5.2.2 Nearest-task (NT) decoding:

- ➢ We identify the subtask, P_X with the spatially closest position to P_B in the task sequences of other robots.
- ➢ It will insert P_B before or after P_X with 50% probability.
- ➢ 2 robot tasks are decoded as per the sequence of the chromosome of Z.



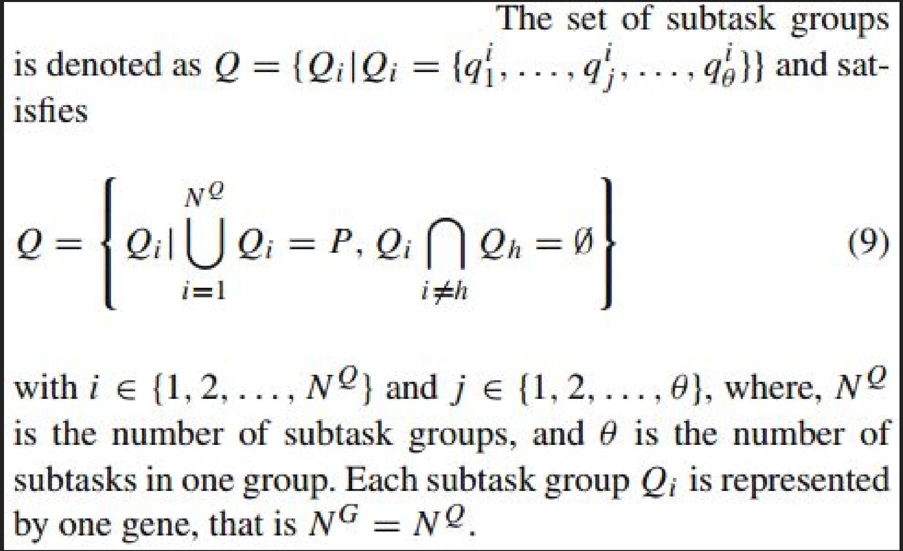
R_1 R_2 R_3
Genotype
1 2 3 4 5 6 7 8
T_1 T_2 T_3 T_4 T_5 T_6 T_7 T_8
c^t_{562}=4 c^t_{592}=7
c^t_{073}=12.4 c^t_{083}=15.4
τ^a_6=10.8
τ^a_7=12.4
S1
P_1 P_2 P_3 P_4 P_5 P_6 P_7 P_{10}
τ^a_9=11.4
τ^a_9=12.2
LW
c^w_{62} = 0.6 R_2 wait
S2
P_1 P_2 P_3 P_4 P_5 P_6 P_9 P_7 P_{10}
τ^a_7=25.4
τ^a_8=16.2
τ^a_8=32.1
τ^a_8=25.4
NT
c^w = 0 no robot wait
Phenotype (LW)
P_1 P_2 P_3 P_4 P_5 P_6 P_8 P_9 P_7 P_{10}
T_6 T_7
2
1
Phenotype (NT)
P_1 P_2 P_3 P_4 P_5 P_8 P_6 P_7 P_9 P_{10}
T_7 T_6

Fig. 12 Task-based coding

Coding Strategies

5.3 Combination-based (CB) coding:

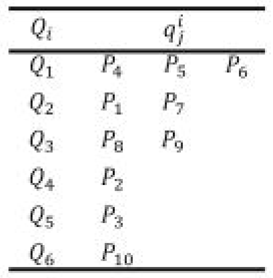
- ➢ Builds small temporal subtask groups.
- ➢ First considers each subtask as independent and then does “task combination”. N^S is maximal number of subtasks in one group and H is the temporal constraint of grouping.



The set of subtask groups is denoted as $Q = \{Q_i | Q_i = \{q_1^i, \dots, q_j^i, \dots, q_\theta^i\}\}$ and satisfies

$$Q = \left\{ Q_i \middle| \bigcup_{i=1}^{N^Q} Q_i = P, Q_i \bigcap_{i \neq h} Q_h = \emptyset \right\} \quad (9)$$

with $i \in \{1, 2, \dots, N^Q\}$ and $j \in \{1, 2, \dots, \theta\}$, where, N^Q is the number of subtask groups, and θ is the number of subtasks in one group. Each subtask group Q_i is represented by one gene, that is $N^G = N^Q$.



Q_i		q_j^i	
Q_1	P_4	P_5	P_6
Q_2	P_1	P_7	
Q_3	P_8	P_9	
Q_4	P_2		
Q_5	P_3		
Q_6	P_{10}		

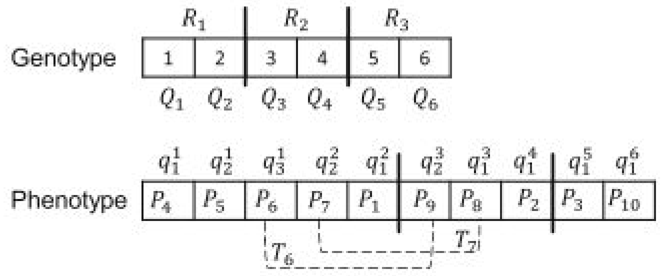
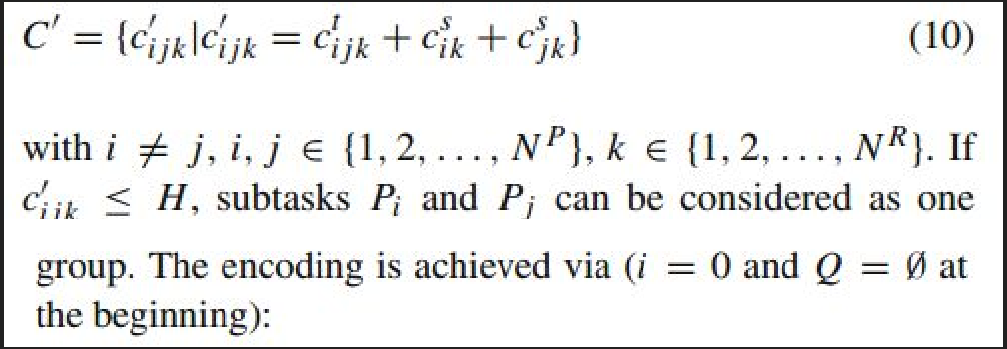


Fig. 13 Combination-based coding

Coding Strategies

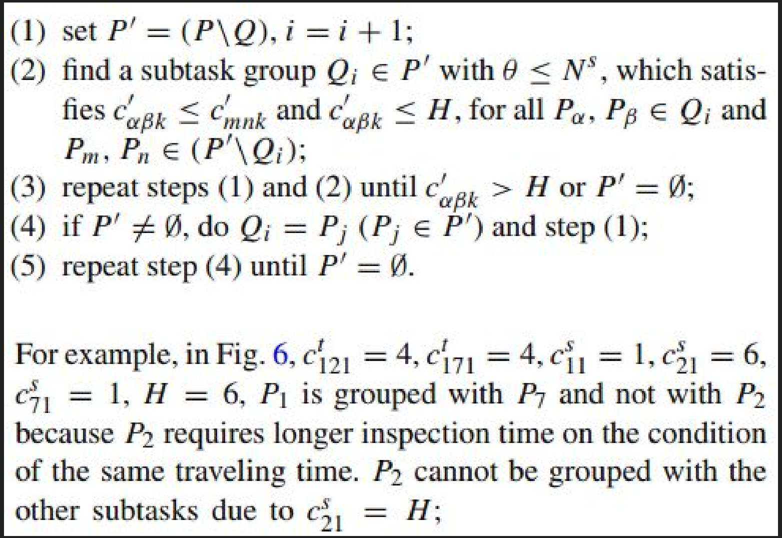
5.3.1 Task combination and encoding strategy:

- Value of H is determined by matrix C’: H belongs to C’ and satisfies that 5% of the elements in C’ are less than H.



$$C' = \{c'_{ijk} | c'_{ijk} = c^t_{ijk} + c^s_{ik} + c^s_{jk}\} \tag{10}$$

with $i \neq j$, $i, j \in \{1, 2, \ldots, N^P\}$, $k \in \{1, 2, \ldots, N^R\}$. If $c'_{ijk} \leq H$, subtasks P_i and P_j can be considered as one group. The encoding is achieved via ($i = 0$ and $Q = \emptyset$ at the beginning):



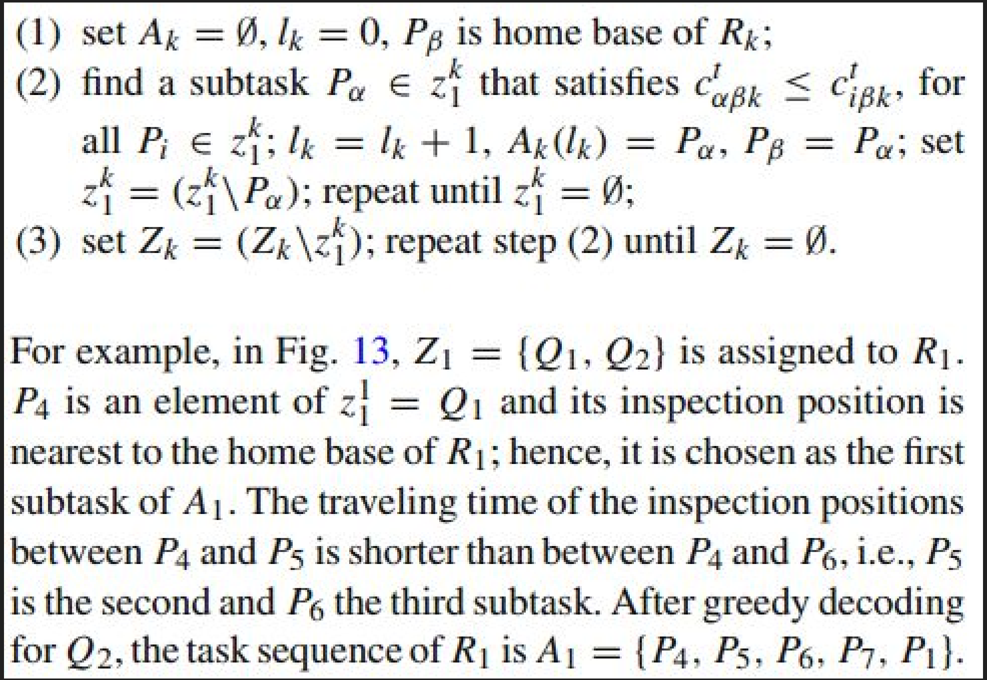
(1) set $P' = (P \backslash Q)$, $i = i + 1$;
(2) find a subtask group $Q_i \in P'$ with $\theta \leq N^s$, which satisfies $c'_{\alpha\beta k} \leq c'_{mnk}$ and $c'_{\alpha\beta k} \leq H$, for all $P_\alpha, P_\beta \in Q_i$ and $P_m, P_n \in (P' \backslash Q_i)$;
(3) repeat steps (1) and (2) until $c'_{\alpha\beta k} > H$ or $P' = \emptyset$;
(4) if $P' \neq \emptyset$, do $Q_i = P_j$ ($P_j \in P'$) and step (1);
(5) repeat step (4) until $P' = \emptyset$.

For example, in Fig. 6, $c^t_{121} = 4$, $c^t_{171} = 4$, $c^s_{11} = 1$, $c^s_{21} = 6$, $c^s_{71} = 1$, $H = 6$, P_1 is grouped with P_7 and not with P_2 because P_2 requires longer inspection time on the condition of the same traveling time. P_2 cannot be grouped with the other subtasks due to $c^s_{21} = H$;

Coding Strategies

5.3.2 Greedy Decoding:

- For any robot R_K its subtask sequence A_K is decoded by this algorithm:



(1) set $A_k = \emptyset$, $l_k = 0$, P_β is home base of R_k;

(2) find a subtask $P_\alpha \in z_1^k$ that satisfies $c_{\alpha\beta k}^t \leq c_{i\beta k}^t$, for all $P_i \in z_1^k$; $l_k = l_k + 1$, $A_k(l_k) = P_\alpha$, $P_\beta = P_\alpha$; set $z_1^k = (z_1^k \backslash P_\alpha)$; repeat until $z_1^k = \emptyset$;

(3) set $Z_k = (Z_k \backslash z_1^k)$; repeat step (2) until $Z_k = \emptyset$.

For example, in Fig. 13, $Z_1 = \{Q_1, Q_2\}$ is assigned to R_1. P_4 is an element of $z_1^1 = Q_1$ and its inspection position is nearest to the home base of R_1; hence, it is chosen as the first subtask of A_1. The traveling time of the inspection positions between P_4 and P_5 is shorter than between P_4 and P_6, i.e., P_5 is the second and P_6 the third subtask. After greedy decoding for Q_2, the task sequence of R_1 is $A_1 = \{P_4, P_5, P_6, P_7, P_1\}$.

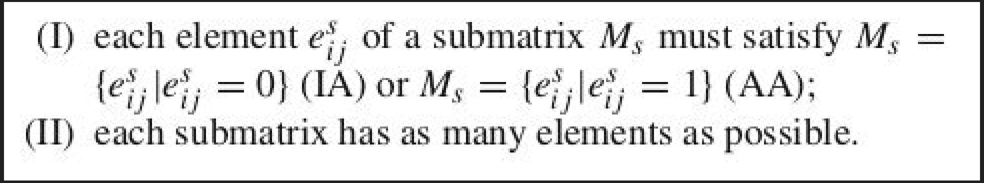
Coding Strategies

5.4 Decomposition-based (DB) coding:

➢ We build spatial local subtask groups with the information we have about the geometry of the target problems.

5.4.1 Task decomposition and encoding strategy:

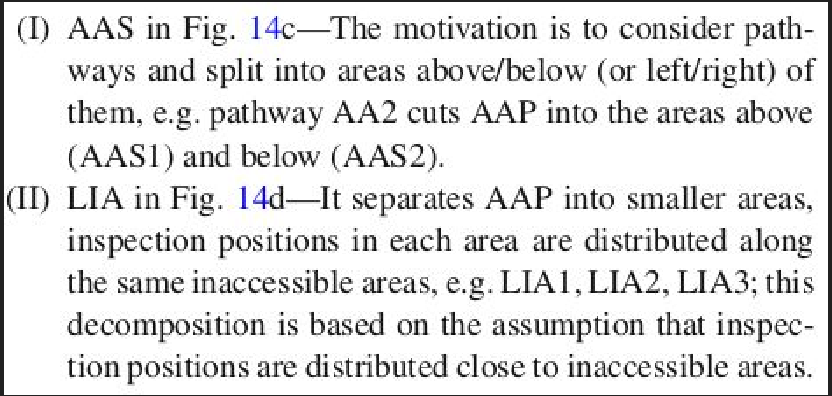
➢ Inspection area is split into accessible (AA) and inaccessible areas (IA). Grid map, E, is separated into sub-matrices (M = M_s | M_s belongs to E) by starting from cell e_{11}



(I) each element e^s_{ij} of a submatrix M_s must satisfy $M_s = \{e^s_{ij} | e^s_{ij} = 0\}$ (IA) or $M_s = \{e^s_{ij} | e^s_{ij} = 1\}$ (AA);
(II) each submatrix has as many elements as possible.

Coding Strategies

- Each accessible region with inspection positions, (AAP) is decomposed in 2 ways:



(I) AAS in Fig. 14c—The motivation is to consider pathways and split into areas above/below (or left/right) of them, e.g. pathway AA2 cuts AAP into the areas above (AAS1) and below (AAS2).

(II) LIA in Fig. 14d—It separates AAP into smaller areas, inspection positions in each area are distributed along the same inaccessible areas, e.g. LIA1, LIA2, LIA3; this decomposition is based on the assumption that inspection positions are distributed close to inaccessible areas.

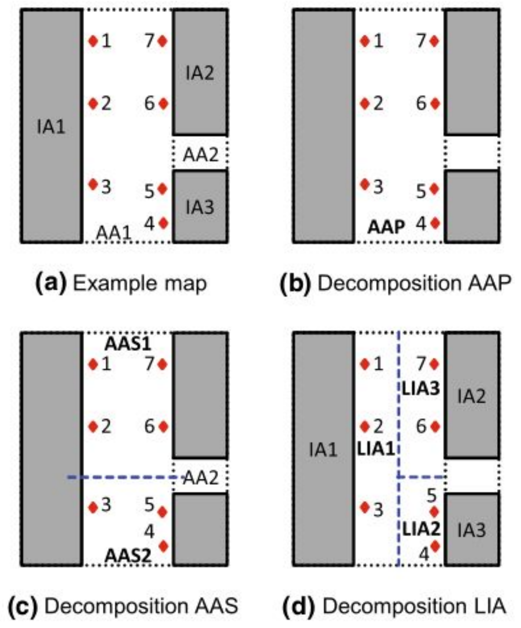
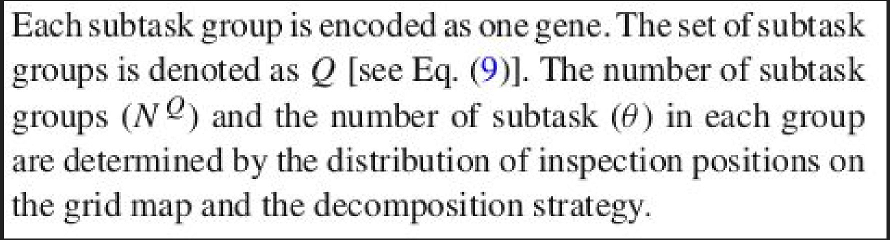


Fig. 14 Example for tasks decomposition strategies



Each subtask group is encoded as one gene. The set of subtask groups is denoted as Q [see Eq. (9)]. The number of subtask groups (N^Q) and the number of subtask (θ) in each group are determined by the distribution of inspection positions on the grid map and the decomposition strategy.

Coding Strategies

5.4.2 Decoding strategy:

- ➢ Three move modes (MM) of robots are designed to accomplish subtasks in one group. They inspect objects along the solid lines.

- ➢ **MM1** - start and end at top.
- ➢ **MM2** - start and end at the bottom
- ➢ **MM3** - start at top and end at the bottom

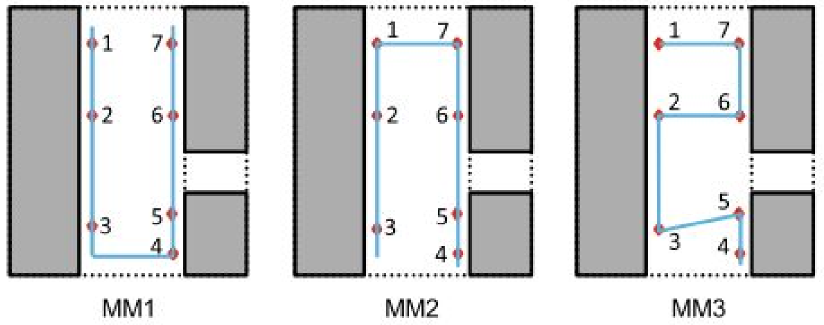
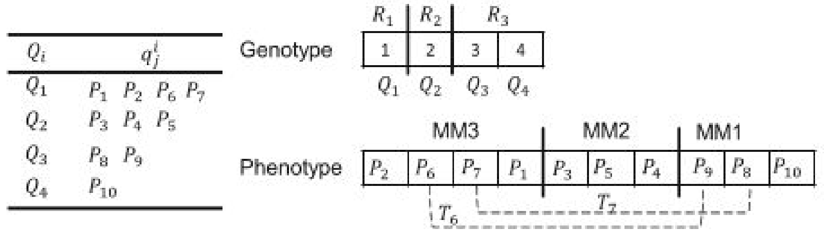


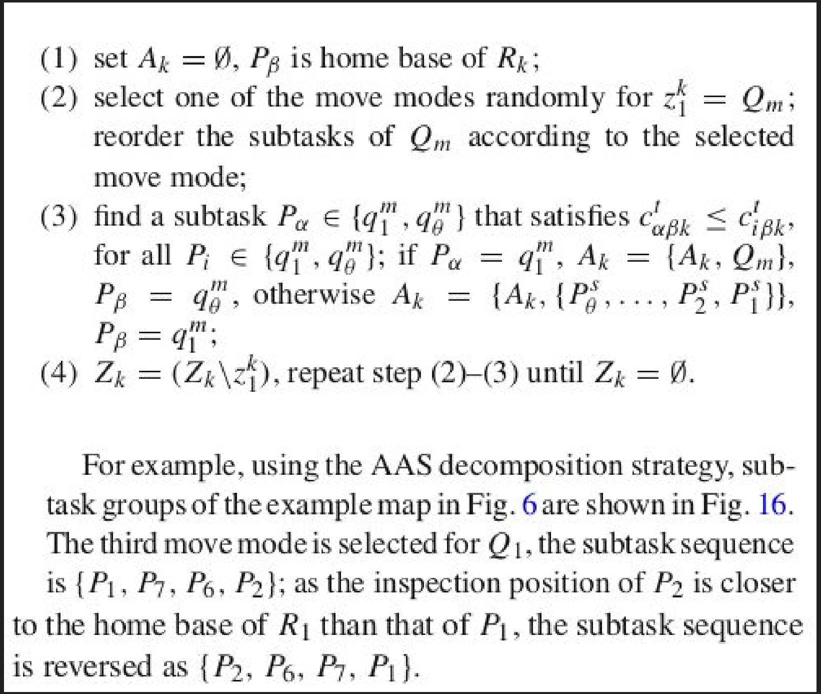
Fig. 15 Three move modes (MM) for the inspection positions in one gene



Q_i	q_j^i
Q_1	P_1 P_2 P_6 P_7
Q_2	P_3 P_4 P_5
Q_3	P_8 P_9
Q_4	P_{10}

Fig. 16 Decomposition-based coding

Coding Strategies



(1) set $A_k = \emptyset$, P_β is home base of R_k;
(2) select one of the move modes randomly for $z_1^k = Q_m$; reorder the subtasks of Q_m according to the selected move mode;
(3) find a subtask $P_\alpha \in \{q_1^m, q_\theta^m\}$ that satisfies $c^t_{\alpha\beta k} \le c^t_{i\beta k}$, for all $P_i \in \{q_1^m, q_\theta^m\}$; if $P_\alpha = q_1^m$, $A_k = \{A_k, Q_m\}$, $P_\beta = q_\theta^m$, otherwise $A_k = \{A_k, \{P_\theta^s, \ldots, P_2^s, P_1^s\}\}$, $P_\beta = q_1^m$;
(4) $Z_k = (Z_k \backslash z_1^k)$, repeat step (2)–(3) until $Z_k = \emptyset$.

For example, using the AAS decomposition strategy, subtask groups of the example map in Fig. 6 are shown in Fig. 16. The third move mode is selected for Q_1, the subtask sequence is $\{P_1, P_7, P_6, P_2\}$; as the inspection position of P_2 is closer to the home base of R_1 than that of P_1, the subtask sequence is reversed as $\{P_2, P_6, P_7, P_1\}$.

Comparison of coding strategies

- In general, smaller number of genes give better results. So, CB and DB have higher efficiency but require more time in encoding and decoding.
- Bad task grouping may lead to a poor search space.
- DB makes an assumption that the inspection area is like a grid.

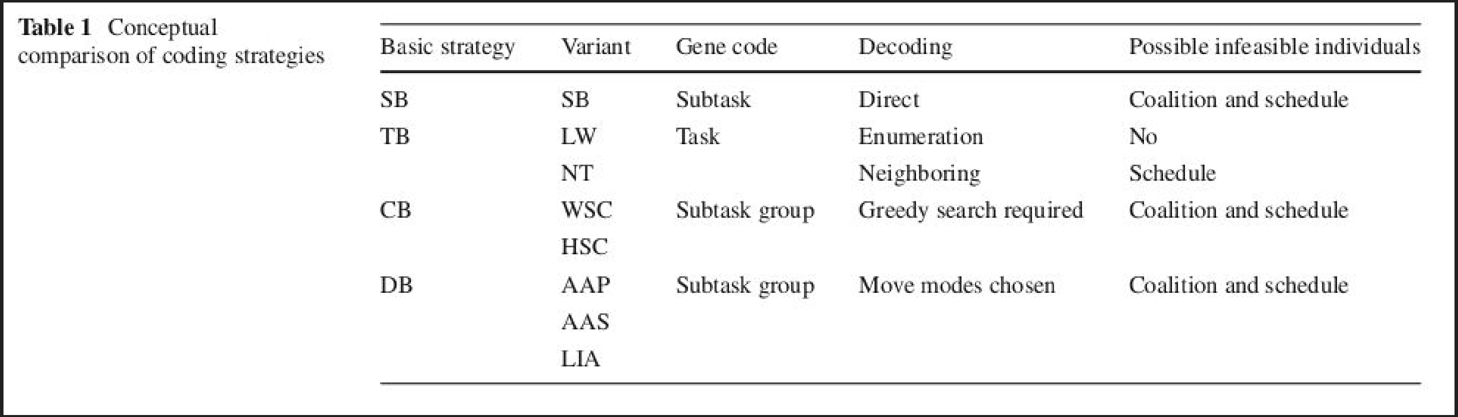


Table 1 Conceptual comparison of coding strategies

Basic strategy	Variant	Gene code	Decoding	Possible infeasible individuals
SB	SB	Subtask	Direct	Coalition and schedule
TB	LW	Task	Enumeration	No
	NT		Neighboring	Schedule
CB	WSC	Subtask group	Greedy search required	Coalition and schedule
	HSC			
DB	AAP	Subtask group	Move modes chosen	Coalition and schedule
	AAS			
	LIA			

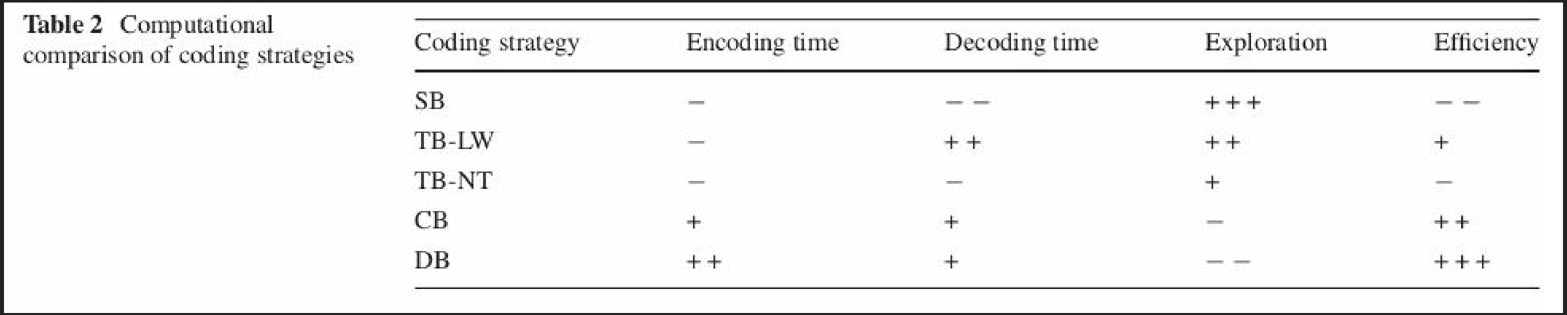


Table 2 Computational comparison of coding strategies

Coding strategy	Encoding time	Decoding time	Exploration	Efficiency
SB	−	− −	+ + +	− −
TB-LW	−	+ +	+ +	+
TB-NT	−	−	+	−
CB	+	+	−	+ +
DB	+ +	+	− −	+ + +

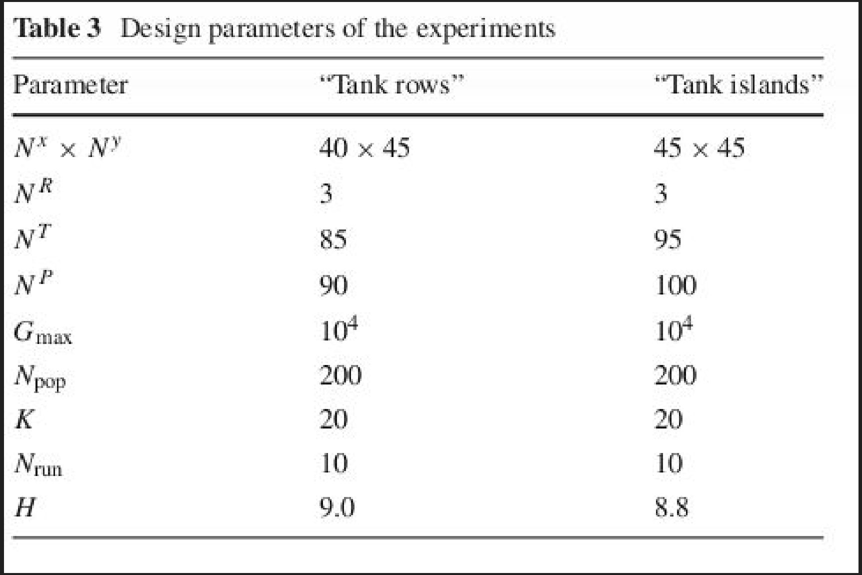


Table 3 Design parameters of the experiments

Parameter	"Tank rows"	"Tank islands"
$N^x \times N^y$	40×45	45×45
N^R	3	3
N^T	85	95
N^P	90	100
$G_{\max}$	10^4	10^4
N_{pop}	200	200
K	20	20
N_{run}	10	10
H	9.0	8.8

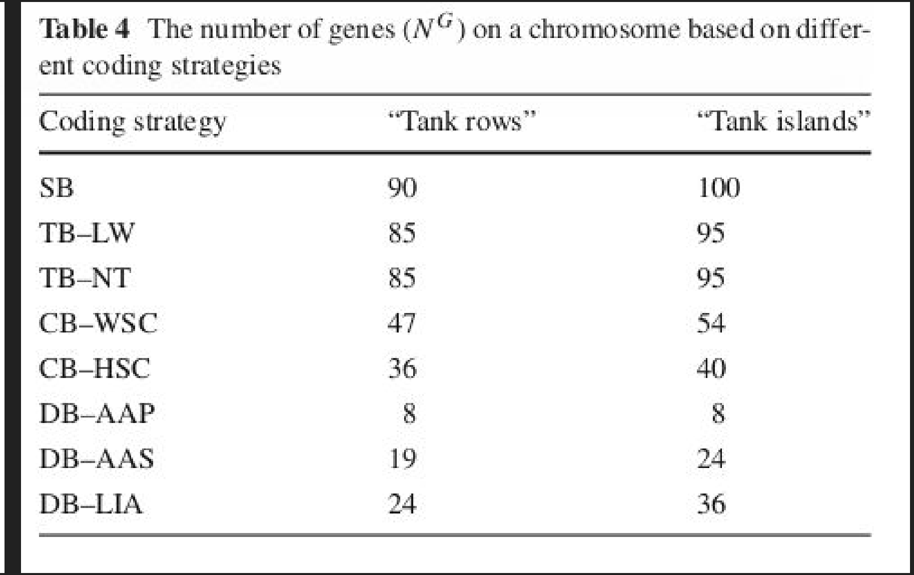


Table 4 The number of genes (N^G) on a chromosome based on different coding strategies

Coding strategy	"Tank rows"	"Tank islands"
SB	90	100
TB–LW	85	95
TB–NT	85	95
CB–WSC	47	54
CB–HSC	36	40
DB–AAP	8	8
DB–AAS	19	24
DB–LIA	24	36

Case studies

- Tested in 2 scenarios: “tank rows” and “tank islands”.
- 10 independent runs of each algorithm performed with $N_{pop} = 200$, $G_{max} = 10^4$.
- Average algorithm performance is measured by: $R_P = ((J_{mean} - J_{opt})/J_{opt})*100$, J_{mean} is average completion time, J_{opt} is best completion time.

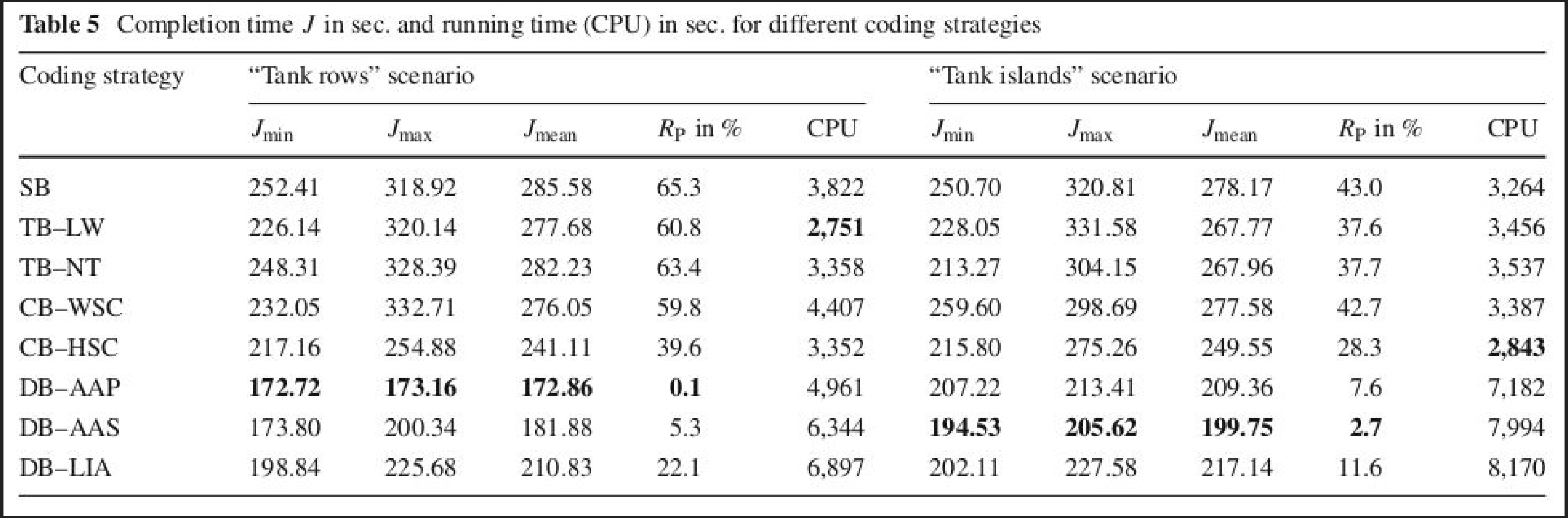


Table 5 Completion time J in sec. and running time (CPU) in sec. for different coding strategies

Coding strategy	“Tank rows” scenario					“Tank islands” scenario				
	J_{min}	J_{max}	J_{mean}	R_P in %	CPU	J_{min}	J_{max}	J_{mean}	R_P in %	CPU
SB	252.41	318.92	285.58	65.3	3,822	250.70	320.81	278.17	43.0	3,264
TB–LW	226.14	320.14	277.68	60.8	**2,751**	228.05	331.58	267.77	37.6	3,456
TB–NT	248.31	328.39	282.23	63.4	3,358	213.27	304.15	267.96	37.7	3,537
CB–WSC	232.05	332.71	276.05	59.8	4,407	259.60	298.69	277.58	42.7	3,387
CB–HSC	217.16	254.88	241.11	39.6	3,352	215.80	275.26	249.55	28.3	**2,843**
DB–AAP	**172.72**	**173.16**	**172.86**	**0.1**	4,961	207.22	213.41	209.36	7.6	7,182
DB–AAS	173.80	200.34	181.88	5.3	6,344	**194.53**	**205.62**	**199.75**	**2.7**	7,994
DB–LIA	198.84	225.68	210.83	22.1	6,897	202.11	227.58	217.14	11.6	8,170

Case studies

- ➢ In tank rows scenario best solution is obtained by DB-AAP.
- ➢ In tank islands scenario best solution is obtained by DB-AAS.
- ➢ Reduction of number of genes improve the performance and in both the above methods it is nearly reduced by 80%.
- ➢ Reducing the number of genes isn't suitable in all cases, ex - tank islands with DB-AAP.
- ➢ SB coding is worst due to large search space and limited computational resources.
- ➢ TB-LW is better than TB-NT in the tank rows scenario.
- ➢ When we compare our results with GA's without local search, we see our memetic algorithm performs better. The improvements is noticeable in the CB and DB codings
- ➢ CB and DB improve very slightly after 1000 generations, so more than one decomposition strategy can be used in the future.
- ➢ SB and TB codings are not better than CB and DB even with $G_{max} = 5*10^5$.

- ➢ For a fixed N_{pop} and G_{max} the time complexity of the GA with SB,TB-NT, CB, DB is *$O(l+mn)$* and for TB-LW it is *$O(mnl)$*, where, m = number of robots a task requires (in our case it is 2), n = number of multi-robot tasks, l = number of tasks.

Thank You.