Methodology

Preliminary

Team Formation Problem(TFP):

One way to look at the issue of team formation is as a puzzle with set coverage (SCP). Let a set of experts $E = E_1, E_2, \ldots\ldots, E_k$, each $E_i \in E$ and $1 < i \leq kand|E| = k$ is the number of total experts. $S = S_1, S_2, \ldots\ldots, S_n$, each $S_i \in S$ and $1 < j \leq n$ and $|S| = n$ is the total number of unique expertise. $S_j = S_1^{E_i}, S_2^{E_i}, \ldots\ldots\ldots, S_m^{E_i}$ is the set of expertise of an expert E_i where each $S_k^{E_i} \in S^{E_i},\ 1 < k < n$ and $|S^{E_i}| = 1$. Moreover, $E_i \in E$,and $S_k^{E_i} \in S$ is an expert's expertise.

The objective of the challenging task referred to as "team building" is to assemble a set of individuals from of the available search area who are all specialists in a certain field and possess a specific set of talents. This group must have all the necessary skills. According to the model developed by Lappaset al.[Lappas2009] the Interconnection Cost between any two experts, E_a and E_b is given in equation 1.

$$ICC_{ab} = 1 - \frac{(S^{E_a} \cap S^{E_b})}{(S^{E_a} \cup S^{E_b})} \quad (1)$$

Where, $ICC = 0$ means both individual a and b have same expertise and $ICC = 1$ means there is no common expertise between a and b individual.

Any project, P requires a set of skills, $S^R \subseteq S$. We can define a set of teams with one or multiple experts, $\mathrm{T} = \{T_1, \mathrm{T}_2, \ldots\ldots, \mathrm{T}_\varphi\}$, each $T_l \in T$, $1 \leq l \leq \theta$, and $|T| = \theta$. Here each team $T_l \subseteq E$ have S^R for completing the project P. Any team $T^\emptyset$ not able to complete the requirement skills of P is not consider in T and not able to find optimal team ($T^\emptyset \notin T$).

The optimized team $T^\#$ which has minimum team members and lowest interconnection cost among all the teams in T. The Objective function can be formulated as:

$$F(T^\#) = \underset{C^T}{arg\,min} \sum_{i=1}^{i=n-1} \sum_{j=i+1}^{j=n} C_{i\,j} \quad (2)$$

Where C^T is the interconnection cost of the team.

Suppose we have a software development project P, which required competencies such as P={ Data Mining, Optimization, Artificial Intelligence, Software Testing}.

Table: Database of UMP expertise

Expert's Name	Skills/Competencies
Dr. Kamal	T-Way Testing, Software Testing, Object Oriented Analysis And Design
Dr. Nasser	Software Testing, Optimization
Dr. Mohd Faizal	Machine Learning, Intrusion Detection System, UI Design, optimization.
Dr. Nabilah Filzah	Data Mining, Mobile Computing, Artificial Intelligence, Manufacturing
Dr. Md. Mustafizur Rahman	Advanced Materials-Composite Special Alloys, Computational Fluid Dynamics, Biotechnology, Artificial Intelligence, Software Engineering.
Md. Abdul Kader	Computational Intelligence, Artificial Intelligence, Optimization, Image Processing, Data Mining.

From the database Skills connection matrix can be calculated. Skill Connection Matrix is a matrix of 0 and 1. The size of SCM is ***Total number of experts*** × ***Total number of unique skills***. If an expert has the skill, then 1 otherwise 0. That means,

$$SCM_{ij} = \begin{cases} 1\ , & j \epsilon S^i \\ 0\ , & \text{otherwise} \end{cases} \quad (3)$$

Table: Skill Connection Matrix

Expertise Expert	T-Way Testing	Software Testing	Machine Learning	Data Mining	Advanced Materials-Composite Special Alloys	Computational Intelligence	Optimization	Intrusion Detection System	Mobile Computing	Computational Fluid Dynamics	Artificial Intelligence	Object Oriented Analysis And Design	UI Design	Biotechnology	Manufacturing	Image Processing	softwere engineering
Dr. Kamal	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
Dr. Nasser	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Dr. Mohd Faizal	0	0	1	0	0	0	1	1	0	0	0	0	1	0	0	0	0
Dr. Nabilah Filzah	0	0	0	1	0	0	0	0	1	0	1	0	0	0	1	0	0
Dr. Md. Mustafizur Rahman	0	0	0	0	1	0	0	0	0	1	1	0	0	1	0	0	1
Md. Abdul Kader	0	0	0	1	0	1	1	0	0	0	1	0	0	0	0	1	0

**Note: SCM is generated using Table: Database of UMP expertise

The Interconnection Cost Matrix(ICCM) can be computed by equation 1. The ICCM is a square matrix with dimension of ***Total number of experts***.

Table: Interconnection Cost Matrix

	Dr. Kamal	Dr. Nasser	Dr. Mohd Faizal	Dr. Nabilah Filzah	Dr. Md. Mustafizur Rahman	Md. Abdul Kader
Dr. Kamal	0.00	0.75	1.00	1.00	1.00	1.00
Dr. Nasser	0.75	0.00	0.75	1.00	1.00	1.00
Dr. Mohd Faizal	1.00	0.75	0.00	1.00	1.00	1.00
Dr. Nabilah Filzah	1.00	1.00	1.00	0.00	0.88	0.86
Dr. Md. Mustafizur Rahman	1.00	1.00	1.00	0.88	0.00	0.89
Md. Abdul Kader	1.00	1.00	1.00	0.86	0.89	0.00

Considering the project skills requirement, some of the possible teams can be formed as shown in the Table: Possible team formation. Each team complete the project requirements.

The interconnection cost of each team is calculated in Table: Teams size and connection cost. From the Table: Teams size and connection cost it is evident that if the team size is minimum the connection cost can also be minimum. The best team is the one in which the communication expenses between various specialists on the team are kept to a minimum. The minimum connection cost team is Team 3 and Team 8.

Table: Possible team formation

Team 1	Team 2	Team 3	Team 4
Dr. Kamal Dr. Nasser Dr. Mohd Faizal Dr. Nabilah Filzah	Dr. Kamal Dr. Mohd Faizal Dr. Nabilah Filzah	Dr. Nasser Dr. Mohd Faizal Dr. Nabilah Filzah	Dr. Kamal Dr. Mohd Faizal Dr. Nabilah Filzah Dr. Md. Mustafizur Rahman Md. Abdul Kader
Team 5	**Team 6**	**Team 7**	**Team 8**
Dr. Nasser Dr. Mohd Faizal Dr. Nabilah Filzah Dr. Md. Mustafizur Rahman Md. Abdul Kader	Dr. Kamal Dr. Nasser Dr. Mohd Faizal Md. Abdul Kader	Dr. Kamal Dr. Mohd Faizal Md. Abdul Kader	Dr. Nasser Dr. Mohd Faizal Md. Abdul Kader

Table: Teams size and connection cost

Team	Team Size	Interconnection Cost
Team 1	4	5.50
Team 2	3	3.0
Team 3	3	2.75
Team 4	5	9.63
Team 5	5	9.38
Team 6	4	5.0
Team 7	3	3.0
Team 8	3	2.75

Multiple Team Formation Problem(MTFP):

In multiple team formation problem, there need teams for multiple projects. Each project has its own requirement skills and projects are independent from each other. The projects may be sub-project of a large projects and each sub-project is assigned for a particular job. In multiple team formation problem, each task has the same search space(pool of experts), same solution representation but the tasks are mutually exclusive among themselves.

Multifactorial Multitasking Basic:

Consider a hypothetical situation where k self-contained optimal teams are required for completing k sub-projects of a software development project. Search of each optimal team can be considered as an optimization task and have to perform k optimization task concurrently. And all the task are minimization problem that means find the optimal teams with minimum Interconnection cost. The $j_t h$ task T_j have a search space X_j on which the objective function can be defined as $F_j: X_j \rightarrow \mathbb{R}$.

In addition, each team has a skills requirement, and all tasks must satisfy the requirement. In such setting MTO aims to simultaneously navigate the search space of all task and constantly building on the

implicit parallelism of population based search to rapidly reduce $\{X_1, X_2, X_3 \dots \dots . . X_k\} = \{ arg\ min \{F_1(X_j), F_2(X_j), \dots \dots \dots F_k(X_j)\}$ where X_j is the feasible solution of T_j.

To compare the evolving individuals of the population in MTO approach, we consume MFO concepts and established a set of attributes for each entity $P_i\ suchthat, i \in \{1,2, \dots, |P|\}$ in a population P [Gupta2016b].

Definition 1 (Factorial Cost): For a given task $\mathrm{T_j}$, the factorial cost FC_j^i of individual $\mathrm{P_i}$ is given by $FC_j^i = \mu \cdot \sigma_j^i + \mathrm{f_j^i}$; where μ is a large penalizing multiplier, $\mathrm{f_j^i}$ and σ_j^i are the objective value and the total constraint violation, respectively, of $\mathrm{P_i}$ with respect to $\mathrm{T_j}$. Accordingly, if $\mathrm{P_i}$ is feasible with respect to $\mathrm{T_j}$ (zero constraint violation), we have $FC_j^i = f_j^i$.

Definition 2 (Factorial Rank): The factorial rank $\mathrm{R_j^i}$ of P_i on task $\mathrm{T_j}$ is simply the index of $\mathrm{P_i}$ in the list of population members sorted in ascending order with respect to j. While assigning factorial ranks, whenever $FC_j^a = FC_j^b$ for a pair of individuals $\mathrm{P_a}$ and $\mathrm{P_b}$, the parity is resolved by random tie breaking. However, since the performance of the two individuals is equivalent with respect to the $\mathrm{j^{th}}$ task, we label them as being $j\text{-}counterparts$.

Definition 3 (Scalar Fitness): The list of factorial ranks $\{R_j^1, R_j^2, \dots, R_j^k\}$ of an individual P_i is reduced to a scalar fitness α_i based on its best rank over all tasks.

$$For\ minimization\ problem, \alpha_i = \frac{1}{min_{j \in \{1,\dots,k\}}\ \{R_j^i\}} \tag{4}$$

The 3-concept described here are also used for different purpose such as:

- Describing how population individuals interact with each other.
- Determining which solution survive in the population between successive generation.
- Classifying and sorting the entire population.

Once the fitness of every individual has been standardized according to Scala fitness, performance comparison can be straightforward manner. For example, individual P_i will be considered dominate individual P_2 in the multifactorial sense if $\alpha_1 > \alpha_2$.

The procedure describing here for comparing the individual is not absolute. As the factorial rank of individual and implicitly its Scala fitness, depends on the performance of every individual of the population, the comparison is population dependent. But the procedure guarantees that if an individual P^* attains the global optimum of any task, then $\alpha^* = 1$, which implies that $\alpha^* \geq \alpha_i$ for all $i \epsilon \{1,2,3, \dots \dots . . |P|\}$. So it can be said that the procedure is indeed consistent with ensuring definition of multifactorial optimality.

Multifactorial Optimality: *An individual* P^* is considered to be optimum in multifactorial sense if there exists at least one task in the K factorial environment which is globally optimizes.

Multitasking Optimization for Multiple Team Formation Problem(MTO-MTFP):

We propose a multitasking approach for solving multiple team formation problem inspired from shared-population multifactorial multitasking optimization. Multifactorial Multitasking adopted two important concepts, transfer optimization and multitasking learning. Transfer optimization is a paradigm that facilitates knowledge transfer across optimization problems. Multitask learning (MTL) is a subfield of machine learning in which multiple learning tasks are solved at the same time, while exploiting commonalities and differences across tasks. Transfer learning in the field of machine learning that can leverage on available knowledge on related tasks to solve the current task instead of starting from nothing.

For multifactorial multitasking, the tasks must be mutually independent from each other, one task does not relate to another task. And most importantly all the tasks must have the Similar solution representation or Unified Representation scheme. Multiple team formation problem has contained the same properties that is mandatory for multifactorial multitasking.

In multiple team formation problem, each task has the same search space(pool of experts), same solution representation but the tasks are mutually exclusive among themselves. We solve multiple team formation problem by considering each team formation as single objective optimization task. Each optimization task has the same objective function to minimize the interconnection cost among team proposed members. In our multitasking approach, we use the same population for each optimization task. In our population-based MTO-MTFP approach, we initialize the population randomly. Each individual of the initial population is updated based on the metaheuristic search operator. To evaluate our approach, we propose 2 distinct metaheuristic search operator a parameter-free Jaya, and a parameter SCA based multitasking algorithm.

The MTO-MTFP solver has several blocks which include preprocessing steps, population initialization, optimizer design, best solution selection to find the optimal teams in the social network.

Preprocessing steps:

In the preprocessing steps, raw data consists of expert names and corresponding expertise are initially separated according to the Skill Definition File(SDF). SDF extracts the unique expert's name and unique expertise from raw data and encoded into a unique numerical number so that the search processing can be performed without complexity. In the second level of preprocessing Skill Extractor(SE) extracts the Skill Connection Matrix(SCM). Skill Connection Matrix is a matrix of 0 and 1. The size of SCM is $Total\ number\ of\ experts \times Total\ number\ of\ unique\ skills$.

The Cost Extractor(CE) extracts Interconnection or Communication Cost Matrix(ICM) from raw data and Skill Connection Matrix according to equation. The ICM is a square matrix with dimension of $Total\ number\ of\ experts$. The ICM stored in a lookup matrix for further utilization.

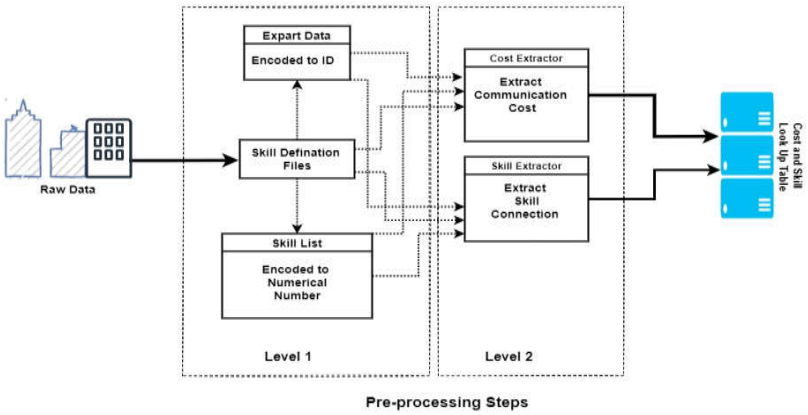


***Figure**: Preprocessing Steps of Multiple team formation system. In the first level unique expert and expertise are extracted. And in the second level Skill Connection Matrix and Interconnection Matrix are Extracted and stored into two separate look-up tables for further utilization.*

Multiple Team Formation Solver Design:

At the beginning level of the algorithm, each task's definition is encoded into a Task Requirement Matrix(TRM). TRM is a row matrix with the dimension of total unique skills and consists of 0 and 1.

$$TRM_j = \begin{cases} 1\ , & j \epsilon S \\ 0\ , & \text{otherwise} \end{cases} \quad (5)$$

Where S is the set of Unique expertise of the dataset. The TRM of the tasks are feed into the MTO optimizer for finding optimal team for each task separately.

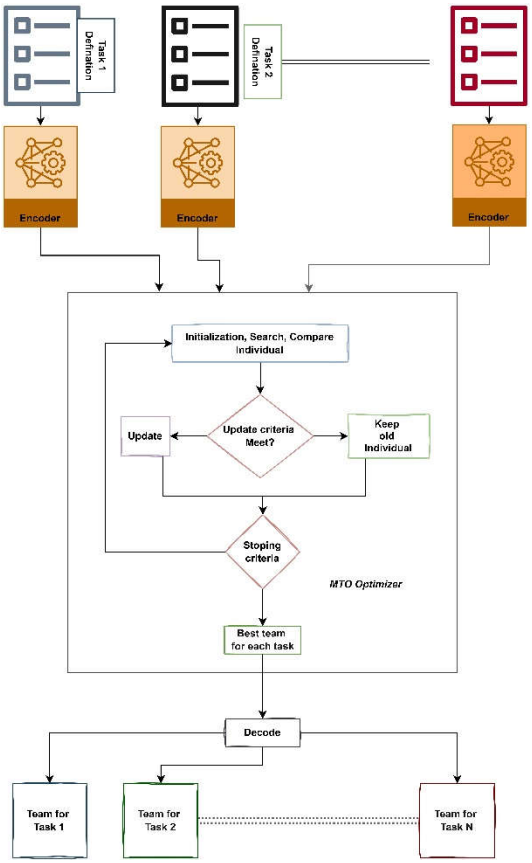


Figure (a) Multiple Team Formation algorithm overview

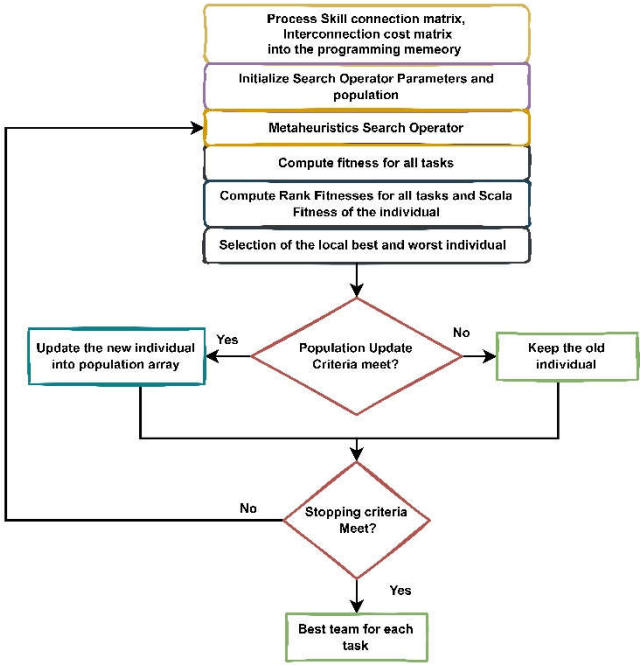


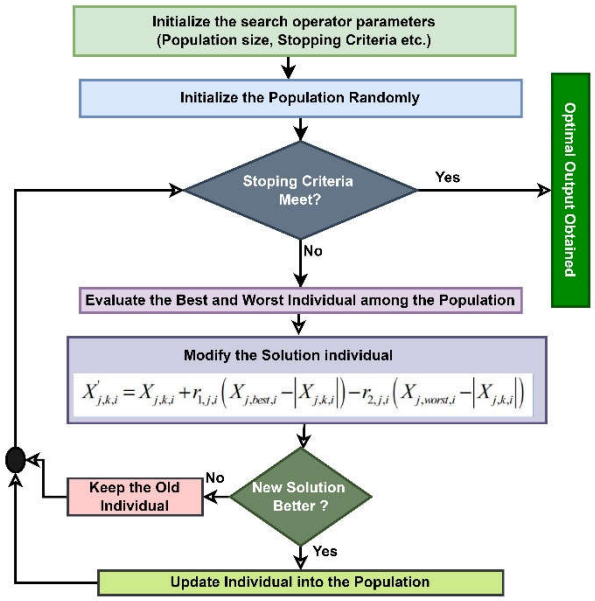
Figure (b) MTO Optimizer Algorithm Flowchart

Figure: Figure (a) is the overview of MTF algorithm where each task is encoded into Task Requirement Matrix(TRM) and feed into the MTO optimizer algorithm to find the optimal teams for each task in a single run. MTO optimizer in figure (b) initialize the search operation by preparing the skill connection and cost connection matrix, initializing search operator parameters and population. The MTO optimizer updates the population based on the population update criteria(i.e., Fitness, Rank, Scala fitness) and terminates the search operations while termination criteria satisfied. At the end, the MTO optimizer proposed the best team for each task separately. The proposed team is decoded into the readable format at the end of the MTFA.

MTO Optimizer:

At the beginning, MTO optimizer processes the Skills Connection Matrix and Interconnection Cost Matrix and stored into the programming memory. After that, MTO optimizer initialize the Metaheuristics search operator parameters(i.e., population size, stopping criteria etc.) and the Optimizer (population based) initialize the population randomly.

The Metaheuristics Search Operator(MSO) explore and exploit the search space according to their behaviors/ equations and modify the sequence/individual. We propose two metaheuristic search operators one is Jaya, a parameter free optimizer and Sine Cosine, a parameter-based search operator as MSO. The two algorithms are depicted into figure.



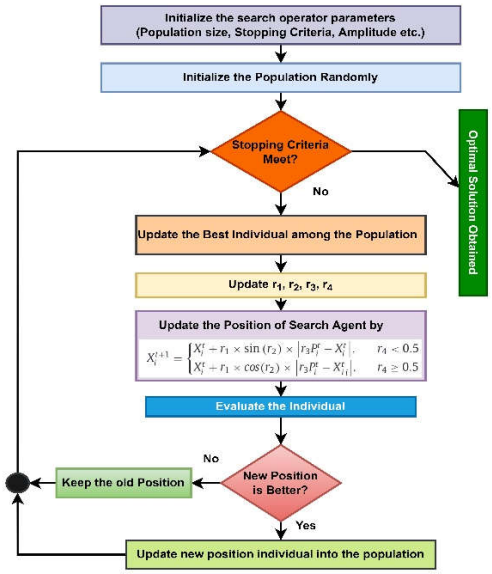


Figure (a) Jaya Optimization Algorithm *Figure (b) SCA Optimization Algorithm*

Figure: Metaheuristic Search Operator. Each operator has the same initialization process. Jaya is a parameter-free metaheuristics only need the basic parameters to be initialized. While SCA is a parameterized metaheuristic needs some parameters to be initialized. Additionally, Jaya needs both the local best and worst individual while SCA only needs local best to modify the population.

We solve team formation problem as a set covering problem. The goal of a set covering problem is to detect the fewest number of sets from a given set of elements, that include (or "cover") all of these components. The objective of team formation problem is to find minimum number of members that cover the team requirement matrix. So, our objective function only finds a sequence of the team members that cover the team requirement matrix. Additionally, we calculate two objective values of the sequence : team size and interconnection cost. So, fitness calculation means finding covered set and calculate team size and interconnection cost.

The MTO optimizer use the same sequence/ individual to calculate the fitness for every task. That means every individual of the population is evaluated separately based on the task requirement

matrices. Based on the single objective(interconnected cost) the individuals are ranked for each task. That means if there are N number of tasks to be performed simultaneously in MTO, each individual is evaluated N times based on the task requirement matrices. Additionally, there are N rank matrices for N number of tasks. The size of each $Rank\ Matrix\ =\ Population\ Size$.

MTO Optimizer calculates Scala Fitness using the Rank Matrices. As our team formation problem focused to minimize the team's interconnected cost, the Scala fitness is inverse of the minimum rank among the rank matrices of the tasks. Scala fitness is important variable is our MTO algorithm because it helps to compare the population. In Single Task Optimization, the population can easily compare based on the objective value or values. But in multitasking Optimization we must compare the population in a way that every task can get the chance to move forward towards the optimal value. Scala fitness is an effect quantity measurement that enables MTO optimizers to take decision whatever it update the individual or keep the old one into the population. Scala fitness measure the survival quality of an individual. If the new Scala fitness is greater than the old Scala fitness that means at least one of the task objective values improves from previous iteration.

Best solution selector?

Most of the metaheuristic's algorithm needs local minima or maxima for its modification of the population. The local best or worst individual guides the metaheuristic algorithm towards global minima/maxima. For single task optimization it is simply the individual which has maximum or minimum objective value/values. But for multitasking optimization algorithm, inappropriate selection of local minima can divert the population towards wrong direction of global minima/maxima. To select the local minima/maxima we propose a selection approach.

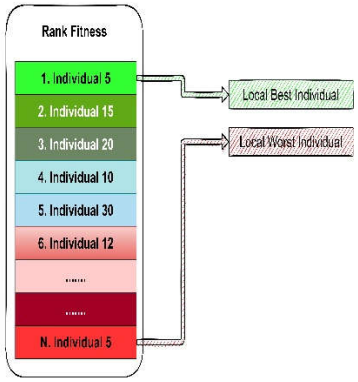
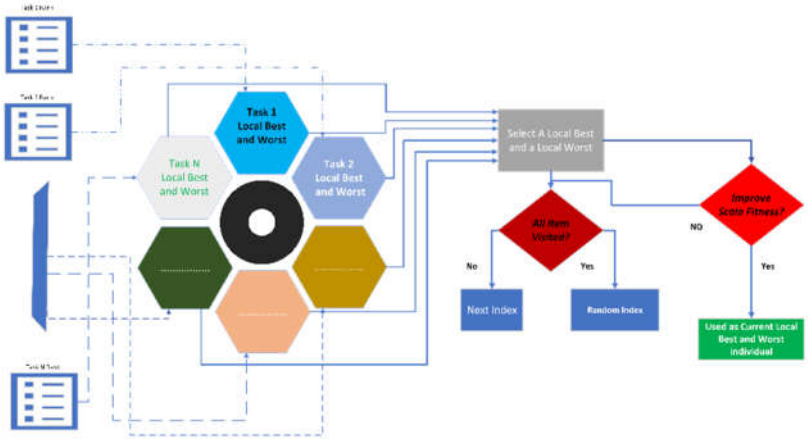


Figure (a) Local Best and Worst individual for a specific task



Figure(b) Schematic Diagram of Local Best and worst population Selector

Figure: Local Best Selector. Each task has its own rank of the population. The first individual of the rank matrix is considered as the corresponding task local best individual within an iteration. Local best selector selects a task's local best and if condition satisfied used as the current local best individual otherwise use next index. If all the local best index is tested the local best selector use random index as the current local best individual.

Each task has its own rank fitness. The first individual of the rank matrix is considered as the local best individual and last index is considered as the local worst individual for a specific task. The Local Best and Worst individuals are updated in each iteration. Local best and worst selector chooses a task's best and worst individual for the current population. If the selection improves the Scala fitness, then MTO optimizer used these as the current local best and worst. Otherwise, the selector first selects next task

until all the task's index are visited. If all the task's index are visited the selector, choose a random task's index as current best and worst individuals for the current population.

The stopping criteria for MTO optimizer is the maximum iteration number. While the maximum iteration number arises the MTO optimizer provides the best optimized team for each task as a string sequence. The Decoder decodes the string sequence into understandable information consisting of the person's names of the optimized teams for all the tasks.