A Fully Heuristic Framework for the Sociometric Multiple Team Formation Problem

Ignacio Martínez Hernández
Metaheurísticas
Doctorado en Sistemas de Ingeniería
Universidad de Talca
imartinez17@alumnos.utalca.cl

***Abstract*—This paper revisits the Multiple Team Formation Problem (MTFP), which optimizes the allocation of human resources to multiple projects considering skill requirements and sociometric affinities. We propose a fully heuristic framework that replaces the exact Constraint Programming solver used in the original formulation with a Randomized Greedy Constructive Heuristic (RGCH). A comprehensive benchmark is conducted, comparing seven algorithms, including a Genetic Algorithm, Variable Neighborhood Search (VNS), and a novel Stochastic Local Search (SLS). Experimental results across three instance scales reveal a notable shift in performance dominance: while VNS proves most efficient for small problems, SLS consistently outperforms both VNS and population-based approaches in medium and large-scale scenarios. Statistical analysis confirms the superiority of SLS in large instances ($p < 10^{-6}$), demonstrating that persistent, focused local search can be more effective than complex metaheuristic mechanisms for high-dimensional MTFP landscapes. This work provides a flexible, solver-free alternative for the MTFP and challenges the assumption that more sophisticated algorithms always yield better results as problem size increases.**

***Index Terms*—Metaheuristics, Variable Neighborhood Search, Stochastic Local Search, Genetic Algorithm, Sociometry, Heuristic Optimization.**

I. Introduction

In the modern competitive landscape, organizations are increasingly required to execute multiple projects simultaneously to maximize resource utilization and meet market demands [1]. Consequently, the allocation of human resources has evolved from assigning individuals to isolated tasks to forming collaborative teams where social interaction plays a pivotal role in performance and productivity [1].

The problem of grouping experts to satisfy specific skill requirements while optimizing a social or functional metric is known in the literature as the *Team Formation Problem* (TFP). While early approaches focused on single-team formation or full-time assignments, real-world scenarios often involve part-time allocations across concurrent projects. To address this, Gutiérrez et al. proposed the *Multiple Team Formation Problem* (MTFP), incorporating fractional dedication levels and sociometric matrices to quantify interpersonal affinity [1].

The original resolution method for the MTFP relied on a hybrid approach combining Variable Neighborhood Search (VNS) with exact Constraint Programming (CP) solvers to handle the feasibility of sub-problems. While effective, the dependence on exact solvers for internal operations can introduce computational overhead and software dependencies that limit the flexibility of the algorithm.

In this work, we revisit the MTFP with a focus on a **fully heuristic framework**. We propose replacing the exact CP component with a *Randomized Greedy Constructive Heuristic* (RGCH) designed to exploit the problem's decomposition property efficiently. The main contributions are summarized as follows:

1) The formulation of a constructive heuristic that guarantees feasible solutions for the MTFP without using external solvers.
2) A comprehensive benchmark comparing seven algorithms ranging from simple baselines (Deterministic Greedy, Random Search) to trajectory-based methods (Hill Climbing, Tabu Search [2]) and population-based approaches (Genetic Algorithm [3]).
3) An empirical validation of the Variable Neighborhood Search (VNS) performance when powered by purely heuristic local search mechanisms [4].

II. Mathematical Formulation

The Multiple Team Formation Problem (MTFP) is modeled as a quadratic discrete optimization problem. The goal is to allocate a set of available people to a set of projects, satisfying skill requirements while maximizing the collective sociometric efficiency of the resulting teams.

A. Sets and Parameters

Let $\mathcal{H} = \{h_1, \ldots, h_n\}$ be the set of available people and $\mathcal{P} = \{p_1, \ldots, p_m\}$ be the set of projects to be executed. The workforce is categorized by a set of skills $\mathcal{K} = \{k_1, \ldots, k_f\}$. We define $\mathcal{Q}_k \subset \mathcal{H}$ as the subset of people who possess skill k. It is assumed that each person has exactly one primary skill, such that $\cup_k \mathcal{Q}_k = \mathcal{H}$ and $\mathcal{Q}_k \cap \mathcal{Q}_{k'} = \emptyset$ for $k \neq k'$.

The sociometric affinity between individuals is given by a matrix $S = [s_{ij}]_{n \times n}$, where $s_{ij} \in \{-1, 0, 1\}$ represents the predisposition of person i to work with person j (negative, neutral, or positive).

Each project p_l requires a specific amount of effort for each skill, denoted by $r_{kl} \in \mathbb{R}_{\geq 0}$. Additionally, projects may have different priorities represented by weights w_l, such that $\sum_{l \in \mathcal{P}} w_l = 1$.

B. Decision Variables

The problem allows fractional time allocations. Let $\mathcal{D} = \{d_1, \ldots, d_t\}$ be a discrete set of allowable dedication fractions (e.g., $\{0.0, 0.25, 0.5, 0.75, 1.0\}$). The decision variables are defined as $x_{il} \in \mathcal{D}$, representing the fraction of time person h_i is assigned to project p_l.

C. Objective Function

The objective is to maximize the *Global Efficiency* (E), defined as the weighted sum of individual project efficiencies. The efficiency of a single project p_l, denoted as e_l, is calculated based on the accumulation of interpersonal interactions weighted by the simultaneous time allocation of the pair involved:

$$e_l = \frac{1}{2}\left(1 + \frac{\sum_{i\in\mathcal{H}}\sum_{j\in\mathcal{H}} s_{ij}x_{il}x_{jl}}{\left(\sum_{k\in\mathcal{K}} r_{kl}\right)^2}\right) \tag{1}$$

Equation (1) normalizes the efficiency to the interval $[0, 1]$. The quadratic term $x_{il}x_{jl}$ ensures that sociometric impact is considered only when both individuals are present in the same project, scaled by their shared time intensity.

The complete optimization model is formulated as follows:

$$\text{Maximize} \quad E = \sum_{l\in\mathcal{P}} w_l e_l \tag{2}$$

$$\text{Subject to:} \quad \sum_{l\in\mathcal{P}} x_{il} \leq 1 \quad \forall h_i \in \mathcal{H} \tag{3}$$

$$\sum_{h_i\in\mathcal{Q}_k} x_{il} = r_{kl} \quad \forall p_l \in \mathcal{P}, \forall k \in \mathcal{K} \tag{4}$$

$$x_{il} \in \mathcal{D} \quad \forall h_i \in \mathcal{H}, \forall p_l \in \mathcal{P} \tag{5}$$

Constraint (3) ensures that no individual is assigned more than 100% of their working time across all projects. Constraint (4) enforces that the sum of dedications for a specific skill within a project exactly matches the requirement r_{kl}. Constraint (5) restricts allocations to the discrete domain $\mathcal{D}$.

III. Proposed Methodology

This study addresses the Multiple Team Formation Problem (MTFP) defined in [1]. While the original approach relies on Constraint Programming (CP) to resolve feasibility sub-problems within a Variable Neighborhood Search (VNS) framework, our work proposes a fully heuristic approach. We substitute the exact CP solver with a randomized constructive heuristic to reduce computational dependency on commercial solvers while maintaining the exploration capabilities of metaheuristics.

A. Problem Decomposition and Encoding

The MTFP requires allocating a set of people $\mathcal{H}$ to a set of projects $\mathcal{P}$, maximizing a sociometric objective function. We exploit the *decomposition property* described in [1], which states that since each person possesses a single skill, the feasibility of the allocation can be solved independently for each skill category $k \in \mathcal{K}$.

We represent a solution X as a collection of sub-solutions $X = \{S_1, S_2, \ldots, S_f\}$, where S_k contains the allocation variables for all individuals possessing skill k.

B. Randomized Greedy Constructive Heuristic (RGCH)

To replace the exact CP solver, we developed a Randomized Greedy Constructive Heuristic. This algorithm constructs a feasible sub-solution for a specific skill k efficiently:

1) **Reset:** All current allocations for individuals with skill k are cleared.
2) **Shuffle:** The subset of people Q_k is randomly shuffled to introduce stochasticity.
3) **Greedy Assignment:** The algorithm iterates through projects requiring skill k. For each requirement, it assigns the maximum feasible dedication level $d \in \mathcal{D}$ to the first available candidate, ensuring that the capacity constraint $\sum_l x_{il} \leq 1$ is never violated.

This heuristic serves as the core engine for neighbor generation and mutation operators in the proposed metaheuristics.

Algorithm 1 Randomized Greedy Constructive Heuristic (RGCH)

Input: Project Requirements R, Dedication Levels $\mathcal{D}$, People in skill k (Q_k).
Output: Feasible allocation X_k for skill k.
$X_k \leftarrow$ zero matrix {Reset all allocations for skill k}
$Capacity \leftarrow 1.0$ for all $h_i \in Q_k$ {Initial capacity: 100%}

$Candidates \leftarrow Q_k$
Shuffle($Candidates$) {Introduce randomization}
for each project p_l in $\mathcal{P}$ **do**
  $Req \leftarrow R_{kl}$ {Required effort for skill k in project l}
  $CurrentFill \leftarrow 0.0$
  for each person h_i in $Candidates$ **do**
    if $CurrentFill \geq Req$ **then**
      break
    end if
    $Needed \leftarrow Req - CurrentFill$
    $Cap_i \leftarrow Capacity(h_i)$
    $ValidLevels \leftarrow \{d \in \mathcal{D} \mid d \leq Needed \text{ and } d \leq Cap_i\}$
    if $ValidLevels$ is not empty **then**
      $Assignment \leftarrow \max(ValidLevels)$ {Greedy Choice}
      if $Assignment > 0$ **then**
        $X_{i,l} \leftarrow Assignment$
        $CurrentFill \leftarrow CurrentFill + Assignment$
        $Capacity(h_i) \leftarrow Capacity(h_i) - Assignment$
      end if
    end if
  end for
end for
return X_k

C. Neighborhood Structure (N^1)

Both trajectory-based algorithms and the local search phase of VNS rely on the same neighborhood definition, denoted as the **1-Skill Neighborhood** (N^1). Given a solution X, a neighbor $X' \in N^1(X)$ is generated by:

1) Randomly selecting a single skill category $k \in \mathcal{K}$.
2) Removing all current assignments associated with skill k.
3) Reconstructing the assignment for skill k using the RGCH described in Section III-B.

This operator guarantees that every neighbor generated is feasible, avoiding the need for penalty functions or repair mechanisms for invalid solutions.

D. Baseline Algorithms

To establish a performance floor, two baseline algorithms were implemented [5]:

- **Deterministic Greedy (DG):** A project-oriented heuristic that fills requirements sequentially without stochasticity. It provides a lower bound for solution quality.
- **Random Search (RS):** An iterative process that generates independent feasible solutions using the RGCH and selects the best one found within the computational budget.

E. Trajectory-Based Metaheuristics

We implemented three trajectory-based methods to analyze the impact of exploration strategies on the N^1 neighborhood [5]..

1) Standard Hill Climbing (HC): This algorithm implements a *First-Improvement* strategy with local convergence criteria [5]. At each step, it samples a subset of neighbors from $N^1(X)$. The first neighbor X' that strictly improves the current efficiency ($E(X') > E(X)$) is accepted. The algorithm terminates immediately if no improving neighbor is found within the sample, assuming a local optimum has been reached.

2) Stochastic Local Search (SLS): Unlike standard HC, this variant relies on computational persistence rather than early termination. It generates a single random neighbor $X' \in N^1(X)$ per iteration. If X' improves the current solution, it is accepted. Crucially, the process does **not stop** at local optima; instead, it continues running until the total computational budget (NFE) is exhausted, allowing the stochastic nature of the RGCH operator to eventually find escape paths from local basins of attraction.

3) Tabu Search (TS): To explore the search space more aggressively, TS generates a candidate list of neighbors at each iteration and selects the best one (*Best-Improvement*), even if it degrades the objective function [2]. To prevent cycling, a *Tabu List* records the indices of recently modified skills. A move involving a tabu skill is forbidden unless it satisfies the *aspiration criterion* (i.e., the neighbor improves the global best found so far).

4) Comparison to Prior Work: Our approach differs fundamentally from the Local Search (LS) presented in the original MTFP formulation by Gutiérrez et al. [1]. While their LS and VNS algorithms utilized an external **Constraint Programming (CP) solver** (e.g., IBM ILOG CP Optimised) to guarantee the feasibility of neighbor solutions, our method operates on a purely heuristic basis. We replace the exact CP solver—used as a black-box feasibility engine—with the **Randomized Greedy Constructive Heuristic (RGCH)**. This substitution creates a fully heuristic algorithm (SLS) with enhanced portability, making it independent of commercial software licenses, and improving scalability by drastically reducing the overhead associated with calling an exact solver during the search process.

F. Population-Based Algorithm (GA)

We propose a Genetic Algorithm (GA) designed to exploit the problem's decomposition property [6]:

- **Initialization:** The population is seeded with feasible solutions generated by RGCH.
- **Skill-Based Crossover:** We implemented a specialized Uniform Crossover that exchanges complete blocks of skill assignments between parents. Since constraints are independent per skill, this operator guarantees that if both parents are feasible, the offspring will structurally remain feasible.
- **Mutation:** A skill k is selected with probability p_m, and its assignment is reconstructed using the RGCH (effectively applying a random move in N^1).

G. Variable Neighborhood Search (VNS)

Finally, we replicated the VNS metaheuristic as the primary solver, following [1], [4]. The algorithm systematically explores neighborhoods N^k of increasing size:

1) **Shaking:** A perturbed solution is generated by simultaneously reconstructing k distinct skills ($k = 1, \ldots, f$) using the RGCH.
2) **Local Search:** The Stochastic Local Search procedure (Section III-E2) is applied to the perturbed solution for a limited number of iterations (intensification phase).
3) **Neighborhood Change:** If the local search improves the global best, the search centers on this new solution and resets $k = 1$. Otherwise, it expands the shaking intensity ($k \leftarrow k + 1$).

IV. Experimental Setup

To evaluate the performance of the proposed algorithms, we conducted a comprehensive computational study. All algorithms were implemented in Python using the `pymoo` library for evolutionary components and `numpy` for vectorized operations.

A. Test Instances

We generated three classes of instances to test scalability and robustness, following the structure defined in [1]:

- **Small instance:** ($|\mathcal{H}| = 20, |\mathcal{P}| = 3$) for validation.

- **Medium instance:** ($|\mathcal{H}| = 100, |\mathcal{P}| = 10, |\mathcal{K}| = 10$) designed to replicate the largest scenario presented in the original study.
- **Large-scale instance:** ($|\mathcal{H}| = 200, |\mathcal{P}| = 20$) to assess performance under high dimensional complexity.

In all cases, the sociometric matrix density (positive relationships) was set to 30%, consistent with the "Group Type II" defined in the literature.

B. Parameter Tuning and Budget

To ensure a fair comparison between population-based and trajectory-based methods, we standardized the computational effort by defining a fixed **Computational Budget** in terms of Function Evaluations (NFE). The budget was scaled according to the instance complexity: 20,000 NFE for the Small instance, 50,000 NFE for the Medium instance, and 100,000 NFE for the Large-scale instance. Table I summarizes the configuration for each algorithm.

TABLE I: Algorithmic Parameters Configuration

Algorithm	Parameter	Value
Common	**Computational Budget**	**Scaled (20k–100k)**
	Independent Runs	30
Genetic Alg.	Population Size	100
	Generations	Budget / 100
	Crossover Prob.	0.9
	Mutation Prob.	0.2
Tabu Search	Candidate List Size	$\max(20, 2 \cdot \lvert\mathcal{K}\rvert)$
	Tabu Tenure	$\max(1, \lvert\mathcal{K}\rvert/2)$
VNS	Shaking Intensity (k)	Dynamic ($1 \rightarrow \lvert\mathcal{K}\rvert$)
	Local Search Iter.	50 per shake
Stochastic LS	Iteration Strategy	Exhaustive Budget
	Neighbors per Iter.	1
Standard HC	Neighbors Sample	$\max(20, 2 \cdot \lvert\mathcal{K}\rvert)$
	Stop Criterion	Local Optimum

C. Performance Metrics

The algorithms are evaluated based on three key metrics:

1) **Mean Efficiency (E_{mean}):** The average global efficiency across 30 independent runs.
2) **Best Efficiency (E_{best}):** The absolute maximum efficiency found across all runs.
3) **Convergence Speed:** Analyzed through evolution curves of the objective function over the NFE consumption.

D. Computational Environment

The experiments were conducted on a workstation equipped with an **AMD Ryzen 7 8700G** processor (8 cores, 16 threads, up to 5.1 GHz) and **32 GB** of RAM, running **Windows 11 Pro**.

To optimize the experimental phase, the 30 independent runs for each algorithm were executed in parallel using Python's `multiprocessing` module, fully utilizing the available multi-core architecture. It is important to note that the reported execution time ($Time$) corresponds to the specific wall-clock time consumed by each individual process, ensuring that the metric remains independent of the parallel scheduling overhead.

E. Statistical Analysis Framework

Comparing stochastic metaheuristics requires rigorous statistical validation beyond mean values. To mitigate variance caused by random initialization, we employed a **paired sample design**, where all algorithms were executed using the same set of 30 random seeds for each instance [7].

The analysis follows a robust protocol adapted to the nature of the problem:

a) Normality Verification: Although the sample size ($N = 30$) theoretically approaches the Central Limit Theorem threshold [8], the efficiency metric in the MTFP is bounded ($E \in [0, 1]$). High-performing algorithms inherently exhibit **negatively skewed distributions** (ceiling effect) as they approach the theoretical optimum, violating the normality assumption required for parametric tests. We formally verify this deviation using the **Shapiro-Wilk test** [9].

b) Hypothesis Testing: Given the anticipated non-normality, we adopt the **Wilcoxon Signed-Rank Test** as the primary method for pairwise comparison [10]. We select the algorithm with the highest mean efficiency as the reference (A_{best}) and compare it against each competitor (A_{comp}). Let d be the vector of differences between paired runs. We test the following one-sided hypotheses:

$$H_0 : \text{median}(d) \leq 0 \quad \text{(No improvement)}$$
$$H_1 : \text{median}(d) > 0 \quad (A_{best} \text{ is superior})$$

The significance level is set to $\alpha = 0.05$. The interpretation of the p-value determines the final verdict:

- **Statistical Superiority ($p < 0.05$):** We reject H_0, confirming that the performance gap is real and A_{best} significantly outperforms the competitor.
- **Statistical Equivalence ($p \geq 0.05$):** We fail to reject H_0, indicating insufficient evidence to claim a difference in solution quality. In this scenario, algorithms are considered **competitive**, and secondary metrics—specifically *computational time*—become the deciding factor for efficiency assessment.

V. Results and Discussion

This section presents the comparative analysis of the seven algorithms across the three defined scenarios. We analyze solution quality (Mean and Best Efficiency), stability (Standard Deviation), computational cost, and convergence behavior.

A. Summary of Performance

Table II summarizes the performance metrics obtained from 30 independent runs per algorithm. The results indicate a clear trend: while VNS excels in small instances, the **Stochastic Local Search (SLS)** becomes the dominant strategy as the problem size increases, surpassing both the Genetic Algorithm (GA) and VNS.

TABLE II: Experimental Results Summary.

Algorithm	**Small instance** ($N = 20$)			**Medium instance** ($N = 100$)			**Large-scale instance** ($N = 200$)		
	$Mean \pm Std$	$Best$	Time(s)	$Mean \pm Std$	$Best$	Time(s)	$Mean \pm Std$	$Best$	Time(s)
Greedy	0.694 ± 0.000	0.694	**0.01**	0.646 ± 0.000	0.646	**0.01**	0.590 ± 0.000	0.590	**0.01**
Hill Climbing	0.801 ± 0.028	0.855	0.01	0.775 ± 0.020	0.812	0.32	0.658 ± 0.011	0.672	3.68
Random Search	0.878 ± 0.011	0.912	8.50	0.788 ± 0.006	0.800	1129.1	0.659 ± 0.004	0.669	16692.9
Tabu Search	0.891 ± 0.009	0.912	4.99	0.829 ± 0.004	0.841	149.6	0.692 ± 0.003	0.699	1104.7
Genetic Alg. (GA)	0.903 ± 0.015	0.932	11.77	0.848 ± 0.008	**0.862**	250.6	0.694 ± 0.003	0.700	2036.3
VNS	$\mathbf{0.907 \pm 0.017}$	**0.939**	5.23	0.837 ± 0.006	0.849	163.0	0.689 ± 0.004	0.697	1284.5
Stoch. LS (Ours)	0.896 ± 0.018	**0.939**	4.98	$\mathbf{0.852 \pm 0.004}$	0.861	148.1	$\mathbf{0.700 \pm 0.004}$	**0.708**	1103.4

Analysis of Peak Performance ($Best$): Beyond average performance, it is crucial to analyze the peak solutions found (E_{best}). In the Small instance, VNS and Stochastic LS (SLS) found the global best known solution (0.939). In the Medium instance ($N = 100$), the **Genetic Algorithm** found the absolute best outlier solution (0.862). However, in the Large-scale instance ($N = 200$), SLS regained dominance in both average and peak performance (0.708), demonstrating that for very large instances, the proposed trajectory-based framework explores the search space more effectively than population-based operators.

B. Performance on Small instance ($|\mathcal{H}| = 20$)

In the small instance, the experimental results reveal a *performance saturation*. Figure 1 illustrates this behavior. The improvement curve is steep for all stochastic methods, effectively saturating the search space within the first 20% of the computational budget.

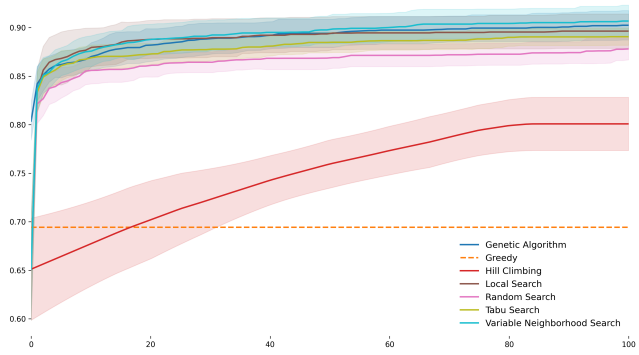

Fig. 1: Convergence profile for the Base Case ($|\mathcal{H}| = 20$). VNS and GA (top cluster) exhibit rapid convergence and high stability. The visual overlap and narrow confidence intervals support the statistical finding that there is no significant difference in quality between VNS and GA. Hill Climbing (Red) shows significant volatility (wide shaded area) during the initial phase.

Statistically, the Shapiro-Wilk test rejected the normality assumption for VNS ($p = 0.039$). The Wilcoxon Signed-Rank test confirmed that VNS is **statistically superior** to Tabu Search ($p < 0.001$) and baselines. However, the comparison against the Genetic Algorithm yielded a p-value of 0.187 (≥ 0.05), indicating **no statistically significant difference** in solution quality.

Given this comparable performance, the computational cost becomes the deciding factor. VNS demonstrated a **2.2x speedup** over the GA (5.23s vs. 11.77s), establishing itself as the most efficient choice for small instances due to its lower overhead.

C. Performance on Medium instance ($|\mathcal{H}| = 100$)

In the medium-sized instance, a divergence in performance becomes visible. Unexpectedly, the **Stochastic Local Search (SLS)** emerged as the dominant strategy (0.852 ± 0.004).

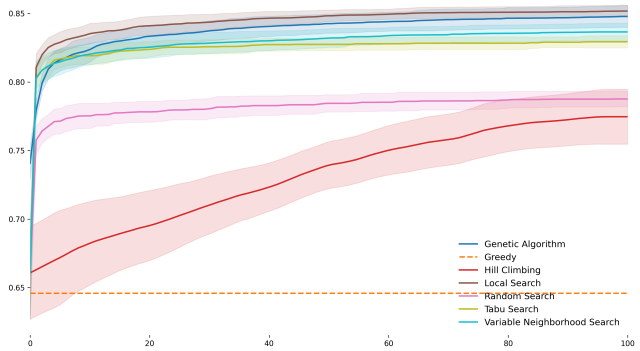

Fig. 2: Convergence profile for the Medium instance ($|\mathcal{H}| = 100$). A clear separation appears: Local Search (brown) achieves the highest final efficiency and maintains a high degree of stability (narrow shaded area). VNS and GA show signs of early stagnation and are surpassed by the persistent, simple exploration of SLS.

Regarding the statistical distribution, and contrary to the base case, the Shapiro-Wilk test **did not reject the normality assumption** for the top algorithms ($p > 0.05$). This indicates that at this intermediate complexity level, the solution qualities are symmetrically distributed around the mean, avoiding the skewness caused by the "ceiling effect" in simpler instances.

Despite this normality, we maintained the use of the non-parametric Wilcoxon Signed-Rank test to ensure methodological consistency across all experimental scenarios. The test confirms that the dominance of SLS is statistically significant compared to the Genetic Algorithm ($p = 0.0035$) and VNS

($p < 0.001$). Furthermore, SLS remained faster (148s) than the GA (250s).

D. Scalability and Large-scale instance ($|\mathcal{H}| = 200$)

In the large-scale scenario, the robustness of the proposed heuristic framework was rigorously tested. The results confirm the dominance of **Stochastic Local Search (SLS)** as the most scalable method.

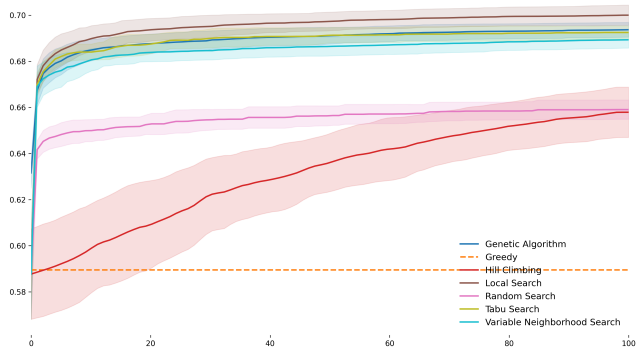

Fig. 3: Convergence profile for the Large-scale instance ($|\mathcal{H}| = 200$). The performance gap widens drastically. The Genetic Algorithm flattens out prematurely, suggesting a loss of population diversity. In contrast, SLS (brown) maintains the highest curve and continues improving throughout the execution budget, confirming its superior scalability and exploration capability in high-dimensional landscapes.

Table II shows that SLS achieved a mean efficiency of **0.700**, whereas the Genetic Algorithm dropped to 0.694 and VNS to 0.689. Statistical analysis (Table III) confirms the superiority of SLS ($p < 10^{-6}$ vs GA). This validates that in high-dimensional MTFP landscapes, the simple yet focused exploitation of the N^1 neighborhood—without the overhead of population management or aggressive shaking—is the most effective strategy.

TABLE III: Statistical Significance Analysis (Wilcoxon Signed-Rank Test) for the Stress Test Scenario.

Comparison	p-value	Verdict
Stoch. LS vs. GA	1.17×10^{-6}	Superior
Stoch. LS vs. VNS	1.86×10^{-9}	Superior
Stoch. LS vs. Tabu	8.20×10^{-8}	Superior
Stoch. LS vs. Random	9.31×10^{-10}	Superior

The Paradox of Statistical Significance: While the numerical difference between the mean efficiencies of SLS (0.7000) and its closest competitor (GA, 0.6937) appears negligible, the Wilcoxon test detected this gap as highly significant ($p < 1.17 \times 10^{-6}$). This result is due to the power of the paired sample design and the low standard deviation ($Std \approx 0.004$) observed across all top algorithms, which indicates high stability. This stability renders the small, persistent advantage of SLS across the 30 paired runs sufficient to definitively reject the null hypothesis, confirming that the superior scalability of the heuristic framework is not attributable to random chance.

VNS Performance Discussion: The underperformance of VNS relative to the pure SLS is likely attributed to the design choice concerning the internal Local Search engine. In our implementation, the LS phase inside VNS was limited to only 50 evaluations per shake (intensification). This fixed, shallow exploitation, combined with the aggressive diversification introduced by the N^k shaking operator, prevented VNS from fully converging to the local optima discovered. The results strongly suggest that the MTFP landscape favors methods with exhaustive, persistent exploitation of the N^1 neighborhood, validating the design of the standalone Stochastic Local Search (SLS) as the most effective variant of the proposed heuristic framework.

VI. Conclusion and Future Work

This study presented a fully heuristic framework for the Multiple Team Formation Problem (MTFP), successfully eliminating the computational dependencies on external exact solvers. The core contribution, the Randomized Greedy Constructive Heuristic (RGCH), proved effective, generating feasible solutions efficiently (in less than 0.01s) and powering a suite of metaheuristics.

Contrary to our initial hypothesis, the Variable Neighborhood Search (VNS) did not maintain its overall superiority as the problem size increased. Instead, the simple yet persistent **Stochastic Local Search (SLS)** emerged as the most robust and scalable algorithm. SLS achieved statistically superior results compared to the Genetic Algorithm (GA) in the stress test ($p < 10^{-6}$), while being significantly faster (approximately 2x) than the GA.

This suggests that for the highly constrained and decomposable search space of the MTFP, intensive exploitation of the well-designed N^1 neighborhood is more effective than the aggressive exploration (shaking) or population management strategies. The persistence of SLS in a single neighborhood proved vital for avoiding stagnation in the complex landscape of high-dimensional instances.

The main limitations of this work are the assumption of single-skilled individuals and the use of synthetic instances. Future work should focus on two main directions: (1) **Problem Extension:** Adapting the RGCH and the neighborhood structure to handle individuals with multiple skills and overlapping proficiencies, and (2) **Algorithmic Recovery:** Exploring adaptive or hybrid mechanisms to enhance the performance of VNS, potentially by using a self-adjusting shaking intensity, to capitalize on its high performance observed in the small instances.

REFERENCES

[1] J. H. Gutiérrez, C. A. Astudillo, P. Ballesteros-Pérez, D. Mora-Melià, and A. Candia-Véjar, "The multiple team formation problem using sociometry," vol. 75, pp. 150–162. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0305054816301198

[2] F. Glover, "Tabu search—part i," vol. 1, no. 3, pp. 190–206. [Online]. Available: https://pubsonline.informs.org/doi/10.1287/ijoc.1.3.190

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," vol. 6, no. 2, pp. 182–197. [Online]. Available: http://ieeexplore.ieee.org/document/996017/

[4] P. Hansen, N. Mladenović, and J. A. Moreno Pérez, "Variable neighbourhood search: methods and applications," vol. 175, no. 1, pp. 367–407. [Online]. Available: http://link.springer.com/10.1007/s10479-009-0657-6

[5] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*, third edition, global edition ed., ser. Prentice Hall series in artificial intelligence. Pearson.

[6] Z. Michalewicz, *Genetic algorithms + data structures: = evolution programs ; with 36 tables*, 3rd ed. Springer.

[7] D. C. Montgomery, *Design and analysis of experiments*, tenth edition, EMEA edition ed. Wiley.

[8] M. H. DeGroot and M. J. Schervish, *Probability and statistics*, 4th ed. Addison-Wesley.

[9] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," vol. 52, no. 3, p. 591. [Online]. Available: https://www.jstor.org/stable/2333709?origin=crossref

[10] F. Wilcoxon, "Individual comparisons by ranking methods," vol. 1, no. 6, p. 80. [Online]. Available: https://www.jstor.org/stable/10.2307/3001968?origin=crossref