Computers, Materials & Continua
DOI: 10.32604/cmc.2022.025077
Article

Tech Science Press

An Enhanced Particle Swarm Optimization for ITC2021 Sports Timetabling

Mutasem K. Alsmadi[1,*], Ghaith M. Jaradat[2], Malek Alzaqebah[3], Ibrahim ALmarashdeh[1], Fahad A. Alghamdi[1], Rami Mustafa A. Mohammad[4], Nahier Aldhafferi[4] and Abdullah Alqahtani[4]

[1]Department of MIS, College of Applied Studies and Community Service, Imam Abdulrahman Bin Faisal University, Dammam, Saudi Arabia
[2]Department of Computer Science, Faculty of Computer Science and Informatics, Amman Arab University, Amman, Jordan
[3]Department of Mathematics, College of Science, Imam Abdulrahman Bin Faisal University, Dammam, Saudi Arabia
Basic and Applied Scientific Research Center, Imam Abdulrahman Bin Faisal University, Dammam, Saudi Arabia
[4]Computer Information Systems Department, College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University, Dammam, Saudi Arabia
*Corresponding Author: Mutasem K. Alsmadi. Email: mkalsmadi@iau.edu.sa
Received: 10 November 2021; Accepted: 31 December 2021

Abstract: Timetabling problem is among the most difficult operational tasks and is an important step in raising industrial productivity, capability, and capacity. Such tasks are usually tackled using metaheuristics techniques that provide an intelligent way of suggesting solutions or decision-making. Swarm intelligence techniques including Particle Swarm Optimization (PSO) have proved to be effective examples. Different recent experiments showed that the PSO algorithm is reliable for timetabling in many applications such as educational and personnel timetabling, machine scheduling, etc. However, having an optimal solution is extremely challenging but having a sub-optimal solution using heuristics or metaheuristics is guaranteed. This research paper seeks the enhancement of the PSO algorithm for an efficient timetabling task. This algorithm aims at generating a feasible timetable within a reasonable time. This enhanced version is a hybrid dynamic adaptive PSO algorithm that is tested on a round-robin tournament known as ITC2021 which is dedicated to sports timetabling. The competition includes several soft and hard constraints to be satisfied in order to build a feasible or sub-optimal timetable. It consists of three categories of complexities, namely early, test, and middle instances. Results showed that the proposed dynamic adaptive PSO has obtained feasible timetables for almost all of the instances. The feasibility is measured by minimizing the violation of hard constraints to zero. The performance of the dynamic adaptive PSO is evaluated by the consumed computational time to produce a solution of feasible timetable, consistency, and robustness. The dynamic adaptive PSO showed a robust and consistent performance in producing a diversity of timetables in a reasonable computational time.

Keywords: Sports timetabling; particle swarm optimization; ITC2021; round-robin tournament; dynamic adaptive

Nomenclature

PSO	Particle Swarm Optimization
DAPSO	Dynamic Adaptive Particle Swarm Optimization algorithm
DAPSO-SD	Dynamic Adaptive Particle Swarm Optimization algorithm and the Steepest Descent Heuristic
SD	Steepest Descent Heuristic
GA	Genetic Algorithm
2RR	Double Round-Robin Tournament
W	Worst (a feasible timetable with highest soft constraints violations)
Σ	Standard Deviation
LP	Linear Programming
IP	Integer Programming
ACO	Ant Colony Optimization
CP	Constraint Programming
F	Feasibility
Q1	First Quartile
Q3	Third Quartile
M	Median

1 Introduction

Timetabling is a way of distributing resources such as players and games over a fixed period of time, a timeslot. This task is difficult and exhaustive. If generating timetables is automated using metaheuristic algorithms, then this saves both time and money for the athlete's/sports federations as in the case of sports timetabling [1,2].

Swarm intelligence is one of the evolutionary computation strategies in search methodologies. It comprises a collective behavior of natural or artificial self-organized and decentralized systems [3]. It was first introduced by [4], in the field of cellular robotic systems. Swarm intelligence systems comprise an agent's population of locally interacting with each other and with their environment. The inspiration comes from nature, especially biological systems. The agents follow very simple pseudo-random rules and decentralized control structure local behavior that lead to the intelligence of global behavior. Common examples are ant colonies, bird flocking, and animal herding. Swarm intelligence refers to a general set of algorithms.

Computationally, Particle Swarm Optimization (PSO) is an approach for optimizing a problem by trying to improve a candidate solution iteratively considering a given quality measure [5,6]. It solves a problem by generating candidate solutions population (*aka*. particles) and moving these particles around in the search space according to simple mathematical formulae based on the particle's velocity and position. Every movement of a particle is affected by its local best-known location but also is guided toward the best-known locations in the search space, that are updated as better positions are detected by other particles. This will converge the swarm toward the best solutions.

A comprehensive survey of PSO variety and applications was made by [7]. Another recent comprehensive review of theoretical and experimental works on PSO has been published by [8]. According to them, PSO is technically defined as a metaheuristic that makes few or no assumptions about the problem being optimized and searches very large spaces of candidate solutions. However,

metaheuristics including PSO do not guarantee an optimal solution. Also, PSO is not a gradient-based optimization, which means it does not require the optimization problem to be differentiable as in classic optimization methods such as gradient descent heuristic.

The constraints in instances of the real-life problem are extremely diverse, where every competition has its requirements. In the ITC2021 competition, two types of constraints are considered for optimization: (i) hard constraints which represent the timetable properties that can't be violated, and (ii) soft constraints which are the preferences that should be satisfied when it is possible. Although many possible objectives of optimization are indicted in the literature, ITC2021 competition deals with problem instances just where the objective is to lessen the penalties from soft constraints that are violated. This was assumed by [9] which makes the formulation of the problem more attractive for timetabling community while retaining the empirical problems' complexity.

Scheduling or timetabling Sports events is considered a big business in hundreds of billion dollars industry, many times as big as automobile and movie industries. For example, early in the 21st century TV networks paid more than $500 million per year for football English premier league. So, rights holders wanted profitable and beneficial feasible schedules. Feasible timetables are also important for extensive players traveling, weekends ticket sales are better, and better to play division teams at the end of the season. In addition, sports timetabling has a wide range of problem types that to be considered. Thus, the idea is simply "which pairs of teams play each other and when?". This presents a challenging task for algorithms in terms of easiness or hardness when dealing with small instances. Finally, a significant theoretical background of the sports timetabling problems provides a fertile environment to test new optimization algorithms or enhance existing ones. Therefore, all of this provides a significant theoretical contribution to both sports timetabling problems as well as optimization algorithms design and enhancement.

In this vast and complex computational task, along with new technologies and scheduling needs, there is a big need to have scheduling schemes in which they are fast and fully utilize the available resources in sports timetabling tasks. The growth of sports timetabling computational complexities makes it necessary to optimize timetabling methods to better solve sports timetabling problems and to fully benefit from the available resources. This work tries to utilize the PSO to improve and optimize sports timetabling problems.

The main goal of this work is to better utilize PSO in sports timetabling problems (e.g., round-robin tournament). To achieve this goal, the following objectives have to be met:

1. Explore recent literature reviews concerning sports timetabling problems.
2. Evaluate the inertia weight factor that alters PSO performance.
3. Propose an enhanced PSO algorithm to better solve sports timetabling problems.
4. Evaluate the proposed algorithm in ITC2021 sports timetabling.

The contribution of this paper is developing a Dynamic Adaptive Particle Swarm Optimization algorithm (DAPSO) to present the performance enhancement of the basic PSO algorithm to optimize the round-robin sports games timetabling by completely satisfying hard constraints and minimizing the violations of soft constraints. Also, a hybrid DAPSO has been proposed to produce a sub-optimal timetable of three categories of datasets dedicated to the sports timetabling problem (provided by ITC2021) in a 2-round-robin fashion. The proposed algorithm is considered as a hybridization of the Dynamic adaptive PSO (DAPSO) algorithm and the Steepest Descent Heuristic; named as DAPSO-SD. It is expected to outperform the traditional PSO and SD algorithms, as well as similar swarm-based and population-based approaches in terms of producing feasible timetables, and speed.

Technically, the main considered issue is to strike a balance between diversity and quality of the search process around the neighborhood of feasible timetables produced by the proposed DAPSO-SD. In other words, not to diversify the exploration around a feasible timetable too much, while not intensifying the exploitation of a better-quality timetable too much. So, this helps the search process to escape local optima and relief the stagnation.

The rest of this paper is organized as follows: the second section demonstrates related literature. The third section discusses the proposed model. computational results and discussion are illustrated in the fourth section. Finally, the fifth section is the conclusion and future works.

2 Literature Review

Scheduling or timetabling task is one of the most important steps in managing and manipulating resources for improving the capabilities of a computational system. Different experiments show that having an optimal solution is extremely difficult but having a sub-optimal solution using heuristic or metaheuristics algorithms is achievable. For demonstrating this difficulty, reference [10] compared three metaheuristics for task scheduling in the cloud environment. They are PSO, genetic algorithm (GA), and a modified PSO version for efficient task scheduling. The goal was to generate an optimal schedule in order to minimize the completion time of task execution.

Until recently, sports schedules were mostly constructed by hand which is time-consuming (with 10 teams, there are numerous possible schedules), many constraints (including television networks, teams, cities), and no new constraints can be added.

Sports timetabling is an old traditional task and has widely spread in the field of computing, research, and industry especially in the last decade. Many heuristics have been implemented to resolve sports timetabling problems such as Linear Programming (LP), Integer Programming (IP), PSO, GA, Ant Colony Optimization (ACO), and others. It is known that the use of optimization techniques (including metaheuristics) can easily adapt when new constraints are added or change the structure of the problem's formulation. So, all professional sports agencies and some institutes construct their sports timetables using optimization.

2.1 Studies Based on Round-Robin Tournaments for Sport Timetabling Problems

A variety of studies and implementations of sport timetabling problems have been conducted in the first decade of the millennium as well as the last two decades of the 20th century. Concerning several constraints, several studies mainly focused on the formulation and settings of an efficient generation of a feasible timetable.

Reference [11] discussed the impact of premature sets from completing a round-robin schedule and how they determine the minimum size of a premature set. The author constructed a round-robin schedule via two methods, the circle method and the greedy algorithm. The circle method continues rotating to get all the slots depending on an initial set of games that have all differences. On the other hand, the greedy algorithm sets games in lexicographic order, where slots are in cyclic order. It then repeatedly assigns each game to either the current slot or the next slot it can feasibly generate. Their study also considered some additional requirements such as the carry-over effect and how to balance it, venues, fixed or prohibited games, and the objective function. They ended up using integer programming and constraint programming algorithms for properly handling the additional requirements and improving the problem's formulation. Similar studies were presented by [12] and [13].

Others proposed GAs for solving the sports timetabling problem such as [14], while [15] tackled multi-objective versions of the problem. Other worth mentioning studies are conducted by [16] and [17] who developed a framework for the problem solver and comprehensive revision of the potential problem formulations, respectively. Many studies with the best results are often obtained by hybrid methods of IP, Constraint Programming (CP), and metaheuristics such as [11]. Devising optimal tournament timetables is crucial to players, cities, fans, teams, security force, TV channels, and other sponsors. Fair and balanced timetables for all teams, satisfying many soft and hard constraints, are the main issue for the attractiveness and the confidence in the professional league tournaments outcome. Therefore, this study is motivated to propose a hybridization between two metaheuristics.

2.2 Time-Constrained Double Round-Robin Tournaments ITC2021[1]

The ITC2021 competition aims to stimulate the development of solvers for the round-robin timetables' construction, meaning that every team plays against every other team a fixed number of times. Besides the round-robin format, there is a lot of other timetabling formats. One example is the knock-out format where teams are linked in pairs and each game's loser is removed. However, round-robin tournaments are the most studied (see [18]) and common format (see [19]). Most sports competitions organize a double round-robin tournament (2RR) where teams meet twice but single, triple, and even quadruple round-robin tournaments as well occur. According to [20], round-robin tournaments are of two types: (i) time-constrained timetables and (ii) time-relaxed timetables. A timetable is time-constrained when it uses the minimal number of the needed time slots, otherwise, it is time-relaxed. This competition only uses time-constrained double round-robin tournaments with an even teams' number. Under this setting, the total number of time slots is equal to the total number of games per team, henceforth each team plays just one game per time slot. For more information, please refer to [21] and [22]. For a comprehensive survey on round-robin implementations for scheduling tasks, please refer to [1]. For more information on the problem's specification and solving please refer to [23–28].

2.2.1 ITC2021–Sports Timetabling Problem Description and File Format

Reference [9] provided a comprehensive description of the ITC2021 – sports timetabling problem and its instance file format. They explained the regulation and procedures for the international timetabling competition. They illustrated the metadata of the problem and its instances, its resources, its structure, and most importantly its hard and soft constraints. Hard constraints include capacity, game, break, fairness, and separation constraints. All of which are subjected to an objective function penalized for soft constraints. For more details, please refer to [9] and the official website of the competition[2] including problem instances' three categories. It is also worth checking the work of [9] that describes the benchmark and file format of a round-robin-based sports timetabling problem. An excellent implementation of round-robin classification and format for the sports timetabling was conducted by [1].

To simplify the process of timetabling sports games, consider four components of the problem: input, scope, constraints, and output. The input fetches a set of teams and a set of timeslots; while the output generates a timetable consisting of fixtures that determine for each game on which timeslot it is to be played. The scope is either based on 2RR, time-constrained timetables, or phased timetables. The most challenging component of the problem is the constraints. Instances are subject to several constraints which are hard and soft constraints. They are mainly concerned with: (i) all hard constraints must be satisfied towards feasibility; (ii) minimizing penalties of violated soft constraints towards optimality.

2.2.2 Problem Instances

The tournament is structured as 2RR with 18, 20, or 22 teams; time-constrained, and phased or no symmetry. Constraints come in 9 types gathered into 5 groups including capacity, break, fairness and attractiveness, game, and separation.

Reference [1] have presented constraints violation penalties which are formulated as each constraint $c \in C$ has its XML notation and precise description, and description of the deviation vector (for hard constraints) is denoted as the sum of the overall elements in the deviation vector, where $D_c = [\, d1\ d2\ \ldots\ dq]$, $\alpha_c = \sum_{k=1}^{q} d_k$. The same description is meant for the violations of soft constraint with the p_c penalty. The objective is to minimize $\sum_{c \in Csoft} p_c \alpha_c$, while $\alpha_c = 0$ for all $c \in C_{hard}$.

Due to the milestone of the competition regarding submissions and the availability of their instances, they consider in the first place the solution value as the only criteria that matters to them in the competition. The rules set are drawn by the organizers in the simplest form such as no computation time or technology restrictions, only solutions in their accredited XML format will be considered, and the same version of the algorithm must be used for all instances.

2.3 PSO

PSO is considered a population-based stochastic optimization approach, which is inspired by social behavior such as bird flocking. It shares many similarities with evolutionary computation techniques like GA [29]. The algorithm is first initialized with a population of random solutions and searches for feasible solutions and then optimal solutions by updating generations. Generally, unlike GA, PSO has no evolution operators such as mutation and crossover. In PSO, candidate solutions, called particles, fly through search space by following the current optimum particles. The advantages of PSO are easier to implement and has fewer parameters to adjust than e.g., GA. Originally, the concept was initiated as a simplified social system simulation, by graphically simulating the choreography of a fish school or bird block. They also found that the particle swarm model can be utilized as an optimizer [30].

In place of employing genetic operators, as in the case of GAs, every particle in the PSO adjusts its flying by its group and own experiences. Each particle is handled as a point in a dimensional space and is operated as in the following equation [30]:

$$v_{id} = v_{id} + c_1 * rand()(p_{id} - x_{id}) + c_2\ rand()(p_{gd} - x_{id}) \tag{1}$$

$$x_{id} = x_{id} + v_{id} \tag{2}$$

where *c1* and *c2* are positive constants and *rand* () is a random function in the range [0,1], $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$ denotes the *i*th particle, $Pi = (p_{i1}, p_{i2}, \ldots, p_{iD})$ is the best preceding position of the particle (which gives the best fitness value), g denotes the index of the best particle among all population's particles, and $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$ is the position change rate (velocity) for particle *i*. Eq. (1) describes how the velocity is dynamically updated. Eq. (1) is divided into 3 components; the momentum, the cognitive, and the social component. In the 1st component, the velocity cannot be abruptly changed: it is attuned based on the current velocity. The 2nd component denotes the learning from its own flying experience. The 3rd component represents the learning from the group learning flying experience. Most optimization algorithms applications are designed for static problems. Eq. (2) illustrates updating the position of the "flying" particles. Many real-world systems keep changing their state frequently, these changes require frequent, sometimes continuous, re-optimization. It has been established that PSO can be effectively applied to optimizing and tracking dynamic systems [30].

The first added parameter into the original PSO for performance and adaptation enhancement is the inertia weight ω. The PSO dynamic equation with inertia weight is changed to be [31]:

$$v_{id} = \omega v_{id} + c_1 * rand()(p_{id} - x_{id}) + c_2\ rand()(p_{gd} - x_{id}) \tag{3}$$

$$x_{id} = x_{id} + v_{id} \tag{4}$$

An adaptive PSO algorithm was developed by [32] and implemented on timetabling a sports training program. Their approach utilized physiological constraints optimization models. However, this version of PSO is not presented clearly in their work, in addition, their implementation is far from the scope of this study, which is the sports timetabling.

3 The Proposed Model

To overcome the problems of the standard PSO algorithm such as low accuracy and premature convergence, the proposed enhanced PSO initializes the process by a random sequence, which is used to balance the diversity of particles. Then, an effective diagnosis mechanism of premature is adopted to determine local convergence and algorithm correction is achieved by random mutation, which could activate the particles in stagnation and make them escape from a local optimum. Simulation experiments are conducted to show the feasibility and effectiveness of the proposed enhanced PSO.

The enhanced PSO algorithm is proposed to construct feasible timetables of the ITC2021-sports timetabling problem and try to solve it by introducing the iterative selection operator. The improvement is mainly focusing as much as possible on avoiding falling into a local optimum.

One of the challenging issues is to completely satisfy hard constraints while satisfying as much as possible soft constraints. Many versions of enhanced PSO algorithms are among the best scheduling algorithm in timetabling. However, PSO algorithms randomly generate the first population. As randomness decreases the probability of the algorithm to converge to the best solution. The proposed enhanced PSO is to overcome the population diversity trade-off.

Sports timetabling is a very important part of the timetabling problems in a variety of industries. Aiming at the characteristics of sports timetabling and considering all stakeholders. The proposed improvement to the enhanced PSO algorithm is based on adaptive weights. The algorithm uses adaptive weights to make the weight change with the increase of the number of iterations and introduces random weights in the later stage, which avoids the situation that the algorithm may be trapped in the local optimum when it comes to the late stage. Applying the algorithm to sports timetabling can achieve a better timetabling plan.

Enhanced PSO

In the standard PSO, every particle is a candidate solution to the numerical optimization problem in a D dimensional space. Each particle in this search space has a velocity assigned to it and a position. The particle's position is represented by $x_i = x_{i1}, x_{i2}, \ldots, x_{iD}$. The particle's velocity is represented as $v_i = v_{i1}, v_{i2}, \ldots, v_{iD}$. Every particle has a local memory ($pBest$) which saves the best position that is so far found by the particle. A globally shared memory ($gBest$) saves the best global position found so far. This data contributes to the particle's flying velocity, by the below equations [33]:

$$v_i = v_i + \varphi_1 * rand * (pBest_i - x_i) + \varphi_2 * rand * (gBest - x_i) \tag{5}$$

$$x_i = x_i + v_i \tag{6}$$

where, φ_1 and φ_2 are constants defining the relative effects of the social and personal experiences. Defining an upper bound for the velocity component enhances the approach's performance. Eq. (6) illustrates the particle position update.

Reconsidering an inertia factor (introduced by [33]) for Eq. (5) which enhances the performance, since it adjusts the velocity over time and enhances the precision of the particles search. Eq. (5) can be rewritten as:

$$v_i = \theta * v_i + \varphi_1 * rand * (pBest_i - x_i) + \varphi_2 * rand * (gBest - x_i) \tag{7}$$

Where θ is the inertia factor and rand is a random number uniformly distributed in [0,1]. Furthermore, K is a constriction factor that is reconsidered for more effective control and velocities constraining. Then Eq. (7) is modified as [33]:

$$v_i = K * (v_i + \varphi_1 * rand * (pBest_i - x_i) + \varphi_2 * rand * (gBest - x_i)) \tag{8}$$

Here K can be expressed as:

$$K = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \tag{9}$$

where $\varphi = \varphi_1 + \varphi_2$. The value $\varphi > 4$, prevents the system explosion, which can happen when the velocities of the particle increase without control. It is known that the successes of constriction factor and inertia equations are problem-dependent. Consequently, this study considered implementing the proposed enhanced PSO with both equations, and the best result is taken.

Specifically, the algorithm relies on the particle distance to its globally closest worst and globally closest best. Reducing the inertia means that during this period, the proposed enhanced PSO increases the importance of the cognitive and social factors and concentrates less on the particle's actual velocity. Sometimes when the particle is trapped in local optima, it decreases linearly so inertia leaves the particle with a velocity less than what the particle needs for escaping those optima. This is the reason for the inertia value decreasing linearly until reaching a certain threshold. At that point, the inertia is reset to its upper bound. Now, an upper and lower inertia bound value is defined. To achieve a balanced diversification, some randomness is added to the lower and upper bounds in addition to the decreasing factor.

This work proposes a modified algorithm in which particles are directed far from the closest worst position. It corrects their positions to the supposed locations, before updating their positions. This can be achieved by updating the value of inertia weight randomly to prevent the particles from being stuck in their local or global search. It is known that settings w near 1 facilitates global search, and settings it in the range (0.2 to 0.5) facilitates the rapid local search. Some researchers recommended the value of inertia weight to be 0.8. However, the strategy of decreasing w is an optimal setting for many problems; where the swarm explores the search domain at the starting of the run and turns it into a local search at the end of the run.

This research uses the following equations to calculate the cognitive and social acceleration coefficient. After choosing the closest worst and best components from the external memories regarding the particle, the similarity of the particle to each of these elements is measured. Eqs. (9) and (10) calculate the similarity between the particle (c_1) and the closest best and the similarity between

the particle (c_2) and the closest worst respectively [33].

$$c_1 = \frac{\sqrt{\sum_{i=1}^{D} (CB - x_i)^2}}{(x_u - x_l)} \tag{10}$$

$$c_2 = \frac{\sqrt{\sum_{i=1}^{D} (CW - x_i)^2}}{(x_u - x_l)} \tag{11}$$

Here x_u and x_l are the lower and upper bound values for every dimension. In the experiments of this research work, the range of this value has been equal for all dimensions. C, which is the position correction coefficient, is defined in Eq. (12) [33].

$$c = \frac{|c_1 - c_2|}{(c_1 + c_2)} \tag{12}$$

Accordingly, a hybrid metaheuristic is proposed in this work namely dynamic adaptive particle swarm optimization with steepest descent heuristic (DAPSO-SD) to overcome issues of the original PSO easily via providing a multiple phase approach for solving each sub-problem. See Fig. 1 demonstrating the generic pseudocode of the proposed method.

Algorithm: DAPSO-SD.
1. **Begin**
2. Randomly **initialize** the particles within their ranges and set all velocities to zero
3. Define external memory for best positions (*Bests*). **Initialize** to all zeros
4. **Until** a certain number of iterations is reached, or convergence **do**:
5. **For each** particle x_i **do**
6. $x_i = x_i + v_i$.
7. **If** particle position at each dimension exceeds its allowed range:
8. x_i = random from range
9. *Val* = evaluate particle
10. **Update** the local best // steepest descent
11. **If** iteration count > max(size),
12. **For each** particle x_i
13. *CB* = closest best to particle.
14. **Calculate** c_1 and c_2 according to Eqs. 10 and 11
15. **Calculate** *C* according to Eq. 12
16. **For each** dimension
17. With probability *p*
18. **Update** the position as per Eqs. 5 and 6
19. **If** $\omega \geq lower\ bound + \varepsilon$
20. $\omega = \omega - \Delta\omega - \varepsilon$ // decrease the inertia
21. **Else**
22. $\omega \geq lower\ bound - \varepsilon$ // resetting the inertia
23. **End**

Figure 1: A generic pseudocode of DAPSO-SD

The inertia value is decreased linearly until a certain threshold is reached. Hence, resetting the inertia to its upper bound. Here, an upper and lower bound value for the inertia is defined. For achieving diversification, some randomness is added to the lower and upper bounds in addition to the decreasing factor ($\Delta\omega$).

Enhanced PSO Implementation

In general, the proposed algorithm starts with generating a random population of feasible timetables, where their hard constraints have zero violations. Then it selects a number of solutions based on

their fitness values from those feasible ones, where their soft constraints are less violated towards zero if existed. Those selected solutions are then permutated locally to generate new feasible solutions with probably lesser soft constraints violations. The proposed algorithm considers permutations made to the order of teams and the determination of the round each team plays. It keeps an elite portion of the population in a steady-state reproduction, where a group of elite solutions is the local optimum to start with the permutations towards optimality (zero violations of hard and soft constraints).

The proposed algorithm implements direct and indirect encodings for the problem-solving representation. The direct encoding is only used to represent a timetable as a n $\times$ n array with some complex operator. On the other hand, the indirect encoding is used to perform a permutation of teams in a n $\times$ n array and orders the teams in a 2RR fashion (home and away) which influences the produced timetable towards optimality.

For example, let us say, there are 9 rounds of games, so each team plays each other once. Then it assigns each game a location (timeslot), based on a series of constraints. Each generated timetable is then evaluated based on the fitness value. Permutations are performed based on the index of the round assignment for each team that violates a constraint. The whole process is iteratively repeated by generating a population of timetables and reproducing them until a sub-optimal is found. For choosing the home teams, first, the algorithm iterates through unscheduled games generated in the population initialization step. Initially, using a degree heuristic to generate feasible timetables but not necessarily optimal, meaning that by choosing higher degree first by assigning a value to the variable which is involved in the highest number of constraints on the other variables that are unassigned. Here, variables are the sequences of games. The fitness evaluation of generated timetable is calculated based on one of three ratings: travel time, consecutive, and location balance.

For generating elite solutions, permutations and reproductions are based on either order of games or perturbation of teams. A permutation is simply the neighbors around a feasible timetable with minimum soft violations. However, it is known that most randomly swapping position or order of timetable's elements do not lead to feasible double round-robin timetables, where possible permutations need to be controlled such as swap locations for a pair of games, swap 2 slots of games, swap timetable of 2 teams, partially swap 2 slots, and partially swap 2 teams.

The criteria of searching either neighborhood structure based on the permutation rate and weight which is it turns depend on the soft constraints and sometimes on the size of the dataset itself. Where, for example, given a round-robin schedule then assign home/away to minimize breaks. Therefore, the DAPSO-SD is proposed which could solve this issue easily via providing a multiple phase approach for solving each sub-problem. The first phase finds home/away feasible patterns, where one sequence per team is conducted. The second phase assigns games consisting only of a feasible home/away pattern. The third phase assigns teams to entries. These phases deploy dividing the problem into subproblems for faster solving or satisfying each constraint. They may guarantee flexibility and robustness.

For a proper selection of neighborhood structures, the DAPSO-SD employs the steepest descent heuristic for selecting a structure based on the soft constraint at hand for converging the exploitation of a solution space towards optimality. This local search is chosen based on a preliminary experiment conducted by the authors, where the steepest descent heuristic proved reliable over similar heuristics.

4 Computational Results and Discussion

DAPSO-SD has been tested over 54 instances from ITC2021[3]. The DAPSO-SD also comprises of examining the influence of utilizing an elite pool together with its implicit solution recombination.

Hence, the aim is to explore the effect of some parameters on the DAPSO-SD performance and how it maintained a balance between diversity and quality. The performance is examined by testing it on the sports timetabling problem in terms of efficiency, consistency, effectiveness, and generality.

4.1 Experimental Setup

Parameters demonstrated in Tab. 1 are experimentally determined and rely on the literature. For example, the population size in a generic swarm optimization metaheuristic is preferred to be relatively small [34]. There are many instances provided for the sports timetabling problem. Therefore, it is determined to test DAPSO-SD on new instances provided by ITC2021.

Table 1: Parameter's settings of DAPSO-SD

Parameters description	Value
MaxIt - Maximum number of iterations	100,000
nPop - Population size (swarm size)	50
ω - Inertia coefficient	1
wdamp - Damping ratio of inertia coefficient	0.99
c1 - Personal acceleration coefficient	2
c2 - Social acceleration coefficient	2
e - Elite percentage	0.2
c - Culled percentage	0.4
Pw - Permutation weight	0.2
Pr - Permutation rate	0.2
Number of non-improvement iterations	100
Controlling ratio (exploration *vs.* exploitation)	1.0$\in$ [1.0, 1.1]
Importance of constraints (penalty)	2.0
Number of employed neighborhood structures per solution	5
Elite pool size	10
Similarity measurement	Least similar = best diverse
Selection	Roulette wheel
Search update	Dynamic, once a better solution is found
Local search routine	Steepest descent heuristic

Parameters tuning of DAPSO-SD is a combinatorial problem. It is relatively simple to find for a specific parameter an approximate optimum, but because they are all correlated, the problem becomes highly non-linear and harder. Consequently, this research is concerned with testing different general enhancements for the PSO instead of intensively fine-tuning each parameter for getting a small

improvement for a particular instance. Improved solutions would likely be reached by utilizing a set of instance-dependent parameters. Though, the goal is to design a robust solver which can solve a variety of instances for the problem of sports timetabling efficiently.

To test the consistency of DAPSO-SD, it is implemented 25 times for every instance with iterations number as a stopping requirement which is a relaxed running time, or a predetermined running time in seconds, or once a feasible solution is found. Using NetBeans IDE 8.2, a Java code is developed which represents the DAPSO-SD algorithm on a Core i7 machine with 16 GB of RAM. In this experimental setup two different values for inertia weight ($\omega = 1$; and $\omega = \omega * 0.95$) are used. By this setup, it showed the ability to reduce the cost of the PSO algorithm and alter the value of inertia weight to be decreased in each iteration. These parameters are tuned as calibrating reduction rate, elite pool size, and non-improvement number of iterations. Then predefining the operator of perturbation. Relying on preliminary testing, it is noticed that these parameter settings produce satisfying results. Refer to Tab. 1. In this study, the size of the elite pool in the DAPSO-SD metaheuristic was fixed intentionally. As for the update strategy, it was maintained. The purpose of inertia is to minimize soft constraints violations as much as possible. When varying the value of inertia weight, this research tries to expand search space and span the space to increase the convergence.

The official website[4] of the ITC2021 overviews the problem instances and specifications that are available in 4 groups showing the number of teams, slots, and constraints classification for each instance.

4.2 Experimental Results

Test results obtained for the sports timetabling problem ITC2021 are shown in Tab. 2. This table also shows descriptive statistics for DAPSO-SD:

- *Lower/Upper bounds*: determined by the organizers and published on their official website.
- feasibility (f): a feasible timetable that has the hard constraints violation = 0.
- first ($Q1$) and third quartiles ($Q3$).
- standard deviation (σ) and median (m).
- worst (w): a feasible timetable with the highest soft constraints violations.

For comparison purposes at the outset, the original PSO is applied and compared to DAPSO-SD. They are applied 25 times for each instance using the same parameters settings with a relaxed computational time. Notice that, the original PSO does not have an external memory and diversification and intensification mechanisms.

Results shown in Tabs. 2–5 indicate that the original PSO is unable to produce feasible timetables for most instances. Therefore, an enhancement of the structure of the original version is required, which is achieved by developing DAPSO-SD metaheuristic outperforming the original PSO across all instances. In which the quality of the solution produced for most instances is considered poor compared to Van Bulk's approach, noticing that the results of the Van Bulk's shown are the best values among the whole number of trials. Although the organizers presented the lower and upper bounds of their obtained results for each instance, they didn't demonstrate the number of trials implemented for each instance.

The same local search has been used in both DAPSO-SD and original PSO. The hybridization in DAPSO-SD including the local search routine and the selection mechanism has a greater effect on enhancing the quality of solutions (see f and w in Tabs. 2–5) while maintaining a search diversity (see σ and w in Tabs. 2–5). On the other hand, the original PSO performs only an opened global particles

update. This is the main drawback of the algorithm which hinders the search process from being balanced between quality and diversity for a smooth and controlled convergence towards optimality. For this exact reason, a hybrid enhanced PSO version is developed in this research work, DAPSO-SD, to control the convergence by balancing space exploration and solution exploitation.

Table 2: The best fitness values obtained by ITC2021 organizer based on Lower/Upper bounds [35], the original PSO *vs.* DAPSO-SD for Early datasets

Instance	Lower/ Upper bounds	PSO						DAPSO-SD					
		f	*Q1*	*σ*	*m*	*Q3*	*w*	*f*	*Q1*	*σ*	*m*	*Q3*	*w*
1	(0, 0)	0, 1088	1522.5	898.5	2453	3079.5	4112	**0, 0**	109	137.7	260	350.5	440
2	(0, 0)	0, 1088	1870	1288.9	2755	4429	4967	**0, 0**	0	.000	0	0	0
3	(0, 0)	*inf*	-	-	-	-	-	**0, 0**	42.5	87.17	161	221.5	270
4	(0, 0)	*inf*	-	-	-	-	-	**0, 0**	0	.000	0	0	0
5	(0, 0)	0, 3185	3185	.000	3185	3185	3185	**0, 0**	360	261.7	479	819.5	1088
6	(0, 0)	*inf*	-	-	-	-	-	**0, 0**	38	51.83	104	138	176
7	(0, 0)	*inf*	-	-	-	-	-	**0, 0**	733.5	874.2	1132	1945	3185
8	(0, 0)	*inf*	-	-	-	-	-	**0, 0**	0	.000	0	0	0
9	(0, 0)	0, 178	178	.000	178	178	178	**0, 0**	0	.000	0	0	0
10	(0, 0)	0, 4967	6081.5	1517.2	8081	9193	-	**0, 0**	31	40.7	67	98.5	131
11	(0, 0)	0, 4112	5863	1758.9	6764	9005.5	-	**0, 0**	908.5	1564.2	2081	3629	4967
12	(0, 0)	*inf*	-	-	-	-	-	**0, 0**	176	151.2	297	385.5	555
13	(0, 0)	0, 440	959	903.03	1673	2548.5	3400	**0, 0**	24	41.7	58	107	131
14	(0, 0)	0, 176	378	310.8	704	981	1130	**0, 0**	23.5	24.7	43	70.5	86
15	(0, 0)	*inf*	-	-	-	-	-	**0, 0**	1450.5	882.9	2030	2715.5	3400

Note: (-) means that no feasible timetable has been obtained, therefore, no descriptive statistical calculation is available (*inf*) means infeasible timetable.

Table 3: The best fitness values obtained by ITC2021 organizer based on Lower/Upper bounds [35], the original PSO *vs.* DAPSO-SD for Middle datasets

Instance	Lower/ Upper bounds	PSO						DAPSO-SD					
		f	*Q1*	*σ*	*m*	*Q3*	*w*	*f*	*Q1*	*σ*	*m*	*Q3*	*w*
1	(0, 0)	0	1173	3101.5	2223.5	4704	6830.5	**0, 0**	0	0	0	.000	0
2	(0, 0)	*inf*	-	-	-	-	-	**0, 0**	0	0	282.5	320.6	655
3	(0, 0)	*inf*	-	-	-	-	-	**0, 0**	0	0	1315	1011.5	2120
4	(0, 0)	0	1088	3241.5	2456.05	5382	7659	**0, 0**	0	0	0	.000	0
5	(0, 0)	*inf*	-	-	-	-	-	**0, 0**	0	0	249.5	175.4	404
6	(0, 0)	*inf*	-	-	-	-	-	**0, 0**	0	0	40	65.14	129

(Continued)

Table 3: Continued

Instance	Lower/Upper bounds	PSO						DAPSO-SD					
		f	*Q1*	σ	*m*	*Q3*	*w*	*f*	*Q1*	σ	*m*	*Q3*	*w*
7	(0, 0)	*inf*	-	-	-	-	-	**0, 0**	0	0	993	1603.5	1890
8	(0, 0)	*inf*	-	-	-	-	-	**0, 0**	0	0	0	.000	0
9	(0, 0)	*inf*	-	-	-	-	-	**0, 0**	0	0	342.5	413.9	612
10	(0, 0)	*inf*	-	-	-	-	-	**0, 0**	0	0	0	.000	0
11	(0, 0)	*inf*	-	-	-	-	-	**0, 0**	0	0	34	57389	89
12	(0, 0)	*inf*	-	-	-	-	-	**0, 0**	0	0	340.5	331.2	619
13	(0, 0)	*inf*	-	-	-	-	-	**0, 0**	0	0	22	46.58	53
14	(0, 0)	*inf*	-	-	-	-	-	**0, 0**	0	0	994.5	1137.8	1946
15	(0, 0)	*inf*	-	-	-	-	-	**0, 0**	0	0	18	27.28	43

Table 4: The best fitness values obtained by ITC2021 organizer based on Lower/Upper bounds [35], the original PSO *vs.* DAPSO-SD for Late datasets

Instance	Lower/Upper bounds	PSO						DAPSO-SD					
		f	*Q1*	σ	*m*	*Q3*	*w*	*f*	*Q1*	σ	*m*	*Q3*	*w*
1	(0, 0)	*inf*	-	-	-	-	-	**0, 0**	0	0	23.5	25.79	44
2	(0, 0)	*inf*	-	-	-	-	-	**0, 0**	0	0	74.5	79.21	147
3	(0, 0)	*inf*	-	-	-	-	-	**0, 0**	0	0	145	156.1	268
4	(0, 0)	0	873	3430	2266.1	5762	7386	**0, 0**	0	0	0	.000	0
5	(0, 0)	*inf*	-	-	-	-	-	**0, 0**	0	0	1785	1163.3	2685
6	(0, 0)	*inf*	-	-	-	-	-	**0, 0**	0	0	143.5	154.7	251
7	(0, 0)	*inf*	-	-	-	-	-	**0, 0**	0	0	22.5	66.26	55
8	(0, 0)	*inf*	-	-	-	-	-	**0, 0**	0	0	717.5	986.4	1821
9	(0, 0)	*inf*	-	-	-	-	-	**0, 0**	0	0	41	55.83	112
10	(0, 0)	*inf*	-	-	-	-	-	**0, 0**	0	0	1698	921.9	2020
11	(0, 0)	*inf*	*inf*	-	-	-	-	**0, 0**	0	0	160	162.3	293
12	(0, 0)	*inf*	*inf*	-	-	-	-	**0, 0**	0	0	837.5	913.8	1526
13	(0, 0)	*inf*	*inf*	-	-	-	-	**0, 0**	0	0	988.5	1063.3	1992
14	(0, 0)	*inf*	*inf*	-	-	-	-	**0, 0**	0	0	156.5	132.4	278
15	(0, 0)	*inf*	*inf*	-	-	-	-	**0, 0**	0	0	36	23.63	56

Table 5: The best fitness values obtained by ITC2021 organizer based on Lower/Upper bounds [35], the original PSO *vs.* DAPSO-SD for Test datasets

Inst-ance	Lower/Upper bounds	PSO						DAPSO-SD					
		f	*Q1*	σ	*m*	*Q3*	*w*	*f*	*Q1*	σ	*m*	*Q3*	*w*
1	(0, 1066)	0	4535	5495.5	1669.4	7227	8846.5	-	**0**	**1066**	1066	.000	1066
2	(0, 176)	0	1066	1679.5	1021.2	2064	3125.5	4535	**0**	**176**	176	.000	176

(Continued)

Table 5: Continued

Instance	Lower/Upper bounds	PSO						DAPSO-SD					
		f	*Q1*	σ	*m*	*Q3*	*w*	*f*	*Q1*	σ	*m*	*Q3*	*w*
3	(0, 1253)	0	4535	6251	1606.7	7719	9107	-	**0**	**1253**	1253	.000	1253
4	(0, 4535)	0	4535	5799.5	1634.6	7415	8968	-	**0**	**4535**	4535	.000	4535
5	(0, 2)	0	66	657	3335.8	5101	7969	-	**0**	**2**	2	.000	2
6	(0, 0)	-	-	-	-	-	-	-	**0**	**0**	1164	3327.6	4365
7	(0, 0)	-	-	-	-	-	-	-	**0**	**0**	2549.5	2654.3	4183
8	(0, 0)	-	-	-	-	-	-	-	**0**	**0**	0	.000	0
Demo	(0, 0)	0	0	.00	.909	1	2	2	**0**	**0**	0	.000	0

Since the competition is not closed yet, it could not compare DAPSO-SD's results against those competitor optimization approaches, they are not reported or published yet. So, it is limited to compare DAPSO-SD's results to the original PSO and ITC2021 organizer [35] reported results. As shown in Tabs. 2–4, for all instances our DAPSO-SD has obtained feasible timetables (see *f*) with lower and upper bounds that are identical to lower and upper bounds (see lower/upper bounds) that are predetermined by the organizers of ITC2021. This means that the DAPSO-SD can meet the predetermined bounds which validate its feasibility and its quality. On the other hand, the original PSO has failed to meet those lower and upper bounds for all 54 instances, hence, infeasibility is dominant. However, to demonstrate the hardness of the instances towards optimality or higher quality timetables, the organizers have provided the test instances shown in Tab. 5, where the bounds vary and the upper bound could be lowered towards 0. It depends on future implementations that can be contributed by the competitors. Thus, results obtained by our DAPSO-SD are also identical to the predetermined bounds with stable performance. So, getting better results and the lowest bounds could be achieved if we run our DAPSO-SD for a longer computational time. So far, no competitors have reported their detailed results (e.g., number of runs, median, average, best, worst, standard deviation, upper and lower quartiles), hence, no fair comparison against our DAPSO-SD is possible. Briefly, based on measures from Tabs. 2–5, our DAPSO-SD showed a consistent, effective, and efficient performance.

Tabs. 2–5 show the results obtained by DAPSO-SD based on parameters presented in Tab. 1 are applied over 54 ITC2021 instances. Best results obtained by DAPSO-SD are presented in bold type which are outperforming the original PSO and is the same as lower and upper bounds.

The DAPSO-SD can produce feasible timetables for all instances on most runs. From Tab. 4, it can be seen that DAPSO-SD has obtained feasible timetables across all instances *w.r.t.* hard constraints satisfaction (violations = 0), while in terms of quality, the algorithm has obtained similar timetables to the lower and upper bounds. The experiments ran 25 times for each instance across all 54 instances. The computational time is set to 15–90 s for small-size instances and 900–9000 s for large instances of the size. They are predetermined experimentally with the least possible and fair ratio. Of course, if DAPSO-SD ran for a longer time, it might be able to obtain better quality timetables or perhaps obtain the optimal ones.

4.3 Results Discussion

It is shown that the algorithm is fairly consistent in producing converged feasible results for almost all instances. The closer the value of the median to the best-obtained result, the more consistent the approach. The effectiveness of DAPSO-SD may be due to efficient search space exploration and effective solution space exploitation. This may present the ability of DAPSO-SD to maintain a balance between the diversity and quality of the search. For clarification of the analysis, this research work relies on the fact that the standard deviation expresses to us how tightly a set of values is clustered around the average of those same values. In which, a standard deviation is a measure of dispersal, or variation, in a group of numbers. Hence, it gives us some indication of the consistency of the algorithm.

For translating the consistency of the algorithm, Tab. 6 presents the statistical tests and descriptions, and factors that indicate the performance of DAPSO-SD for all instances across 25 trial runs. Results of the *t*-test statistical comparison of DAPSO-SD against the original PSO. Generally, the *t*-test statistical comparison is carried out with 24 degrees of freedom at a 0.05 level of significance.

Table 6: Paired Samples *t*-test (PSO *vs.* DAPSO-SD)

Instance	*t*-test	*df*	*p*-value	Mean Difference	σ	*Std*. Error Mean	95% Confidence Interval of the Difference	
							Lower	Upper
Early[a]	2.8309	24	.0254	1904.25	1902.56	672.658	313.67	3494.83
Middle[a]	26.600	24	.0239	1130.50	60.10	42.500	590.49	1670.51
Late∗	-	-	-	-	-	-	-	-
Test[b]	1.8998	24	.1159	1284.17	2308.37	675.948	−453.41	3021.75

Note: a by conventional criteria, this difference is considered to be statistically significant. b by conventional criteria, this difference is considered to be not statistically significant. ∗ t cannot be computed because the standard deviation is 0.

Note the mean difference for each instance group shows the average value of the differences between the best results obtained across the 25 runs for both presented algorithms. To find the results of the difference in the means of instances, the *p*-value is found for the test. The *p*-value refers to the *t*-test for Equality of Means. For example, the *p*-value is .0239 for the Middle instances group in Tab. 6, which implies that the difference in means is statistically highly significant at p-value $\leq .01$, significant at p-value $\leq .05$, and marginally significant at p-value $\leq .1$ levels.

The exploration in DAPSO-SD is performed by utilizing an elite pool that contains a collection of elite solutions found during the search in order to prevent premature convergence of the search. The elite solutions in the memory are utilized by the diversification mechanism to regenerate new solutions. The diversification mechanism reinitializes the generation of particles to divert the search perhaps toward the global solution when it stagnates. The memory provides good solutions that may lead to the global solution. Intensive solution exploitation is performed by further exploring the elite solution's neighbors in order to significantly enhance the quality of the solution.

The diversification mechanism assists inertia boundaries to further diversify the particle search by exploring different regions of the search space, whilst the intensification mechanism assists the updating the position to further intensify the search around the elite solution space. This may eventually maintain a balance between diversity and quality of the search. These mechanisms proved to be effective and significant in producing good quality results after all. In contrast, the original PSO

has no effective diversity control mechanism, and it totally relies on the local search routine to improve the quality of solutions, which is not enough to maintain a balanced convergence. In the PSO, elite solutions permutations are performed implicitly based on the changes of the position. Specifically, the premature convergence occurred in the intensification phase, where the neighborhoods of an elite solution are randomly explored. Therefore, the solution recombination relies on limited information about the elite solution. This prevents it from estimating the effectiveness of performing a permutation to the solution. In addition, the population size in the PSO is maintained relatively small. Hence, some information about the current elite solution is lost during the search in successive iterations where it does not guarantee an effective convergence toward optimality.

In general, DAPSO-SD, besides intensification and diversification mechanisms, elitism is used to manage the position updates, where just the best particle in each iteration is used for updating the position. Therefore, they are considered crucial guidance of the search. The diversification mechanism avoids premature convergence of the search, which is accomplished by the inertia and generating new solutions from those in the elite pool, whilst the intensification mechanism intensifies the search around an elite solution, which is accomplished by the position update and adding good quality solutions into the elite pool. Finally, this research work has contributed the following:

- A balance between search intensification and diversification is maintained by hybridizing PSO with: dynamic update strategy; diversification and intensification mechanisms; elite pool; and steepest descent heuristic.
- Effective exploitation via the elite pool to generate a new good quality population of particles. It is dynamically updated by replacing worse diverse solutions stored in the pool with new better diverse solutions.
- DAPSO-SD produced feasible timetables for the majority of instances.

5 Conclusions

An enhanced version of the PSO named DAPSO-SD has been proposed, which is a hybrid version containing a dynamic and adaptive diversification mechanism utilizing an elite pool and the steepest descent local search routine. It is observed that the search converges stability and very quickly towards feasibility. This proposes that once the local search is employed a smaller population size is more effective in search space exploration, also this may give better performance than an equivalent number of neighborhood structures from solutions that are randomly constructed. That is why this research work employed the strategy of generating new solutions from the elite pool instead of scratch. In addition, for each iteration, randomly updating the value of inertia weight has a significant role to prevent the particles from being stuck in their local or global search.

Briefly described, the obtained results proved the superiority of DAPSO-SD over the original PSO concerning generality, efficiency, consistency, and particularly in terms of the tested instances. This is mainly factored by the usage of the elite pool in generating diverse and high-quality timetables that are also consistent as opposed to the original PSO. These observations are confirmation of the capacity of DAPSO-SD in generating feasible outcomes across all instances. So, DAPSO-SD can be used for further research on sports timetabling problems and complex scheduling problems.

Although the performance of DAPSO-SD is fairly sufficient regarding a guided convergence toward feasibility and sub-optimality, it had some drawbacks. The limitation is the albescence of a clear and meaningful representation of positions regarding the team assignments. It is hard to estimate the effects of improvements made to a global optimum. This is most unlikely to be achieved due to the implicit solution recombination operators and the indirect solution representation used in the PSOs.

Where these operators are very useful in performing a guided sampling of the search space using only the information about the structure of the global optimum. Then, it can only be applied to the indirect representation, which is not suitable for measuring the diversity of the search in terms of measuring similarities between solutions in the population within a Hamming or Euclidean space (e.g., as in GAs). From the author's perspective, any PSO considers the neighborhood space due to the fact that positions are indirectly representing the team-timeslot assignments in a timetable using floating numbers to indicate feasibility or cost. In future work, it is possible to introduce a new approach concerning the drawbacks of DAPSO-SD, such as the diversity measurements of the population in order to determine the right degree of search diversity. This can be achieved by measuring similarities within a Euclidean space in order to perform explicit recombination of elite solutions.

Funding Statement: This work was supported by Deanship of Scientific Research at Imam Abdulrahman Bin Faisal University, under the Project Number 2019-383-ASCS.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] D. Van Bulck, D. Goossens, J. Schönberger and M. Guajardo, "RobinX: A three-field classification and unified data format for round-robin sports timetabling," *European Journal of Operational Research*, vol. 280, pp. 568–580, 2019.

[2] X. Yi, D. Goossens and F. T. Nobibon, "Proactive and reactive strategies for football league timetabling," *European Journal of Operational Research*, vol. 282, pp. 772–785, 2020.

[3] Y. Feng, S. Deb, G. -G. Wang and A. H. Alavi, "Monarch butterfly optimization: A comprehensive review," *Expert Systems with Applications*, vol. 168, pp. 114418, 2021.

[4] J. Wang and G. Beni, "Cellular robotic system with stationary robots and its application to manufacturing lattices," in *Proc. IEEE Int. Symp. on Intelligent Control*, Albany, NY, USA, pp. 132–137, 1989.

[5] E. H. Houssein, A. G. Gad, K. Hussain and P. N. Suganthan, "Major advances in particle swarm optimization: Theory, analysis, and application," *Swarm and Evolutionary Computation*, vol. 63, pp. 100868, 2021.

[6] R. Alayi, M. Mohkam, S. R. Seyednouri, M. H. Ahmadi and M. Sharifpur, "Energy/economic analysis and optimization of on-grid photovoltaic system using CPSO algorithm," *Sustainability*, vol. 13, pp. 12420, 2021.

[7] Y. Zhang, S. Wang and G. Ji, "Review Article: "A comprehensive survey on particle swarm optimization algorithm and its applications," in *Hindawi Publishing Corporation Mathematical Problems in Engineering*, Hindawi Publishing Corporation: United Kingdom, vol. 2015, Article ID 931256, pp. 38, 2015.

[8] M. R. Bonyadi and Z. Michalewicz, "Particle swarm optimization for single objective continuous space problems: A review," *Massachusetts Institute of Technology Evolutionary Computation*, vol. 25, no. 1, pp. 1–54, 2017.

[9] D. Goossens, J. Belien, M. Davari and D. Van Bulck, "The international timetabling competition 2021: Sports timetabling," *in INFORMS Annual Meeting*, 2020. http://hdl.handle.net/1854/LU-8706747

[10] S. Abdi, S. A. Motamedi and S. Sharifian, "Task scheduling using modified PSO algorithm in cloud computing environment," in *Int. Conf. on Machine Learning, Electrical and Mechanical Engineering*, Dubai (UAE), pp. 8–12, 2014.

[11] M. A. Trick, "Integer and constraint programming approaches for round-robin tournament scheduling," in *Int. Conf. on the Practice and Theory of Automated Timetabling*, Gent, Belgium, pp. 63–77, 2002.
[12] A. Aggoun and A. Vazacopoulos, "Solving sports scheduling and timetabling problems with constraint programming," in *Economics, Management and Optimization in Sports*, ed: Springer, Berlin, Heidelberg, pp. 243–264, 2004.
[13] C. C. Ribeiro, "Sports scheduling: Problems and applications," *International Transactions in Operational Research*, vol. 19, pp. 201–226, 2012.
[14] J. Schönberger, D. C. Mattfeld and H. Kopfer, "Memetic algorithm timetabling for non-commercial sport leagues," *European Journal of Operational Research*, vol. 153, pp. 102–116, 2004.
[15] A. R. Duarte and C. C. Ribeiro, "Referee assignment in sports leagues: approximate and exact multi-objective approaches," *in 19th Int. Conf. on Multiple Criteria Decision Making*, Auckland, pp. 58–60, 2008.
[16] M. Gröbner, P. Wilke and S. Büttcher, "A standard framework for timetabling problems," in *Int. Conf. on the Practice and Theory of Automated Timetabling*, Gent, Belgium, pp. 24–38, 2002.
[17] G. Kendall, S. Knust, C. C. Ribeiro and S. Urrutia, "Scheduling in sports: An annotated bibliography," *Computers & Operations Research*, vol. 37, no. 1, pp. 1–19, 2010.
[18] S. Knust, "Classification of literature on sports scheduling," 2020. [Online]. Available: http://www2.inf.uos.de/knust/sportssched/sportlit_class/. Accessed: 25/08/2021.
[19] D. R. Goossens and F. C. Spieksma, "Soccer schedules in Europe: An overview," *Journal of Scheduling*, vol. 15, pp. 641–651, 2012.
[20] G. L. Nemhauser and M. A. Trick, "Scheduling a major college basketball conference," *Operations Research*, vol. 46, pp. 1–8, 1998.
[21] D. Van Bulck, D. Goossens, J. Schönberger and M. Davari, "ITC2021 – sports timetabling problem description and file format," 2020a. [Online]. Available: https://www.sportscheduling.ugent.be/ITC2021/images/OrganizationITC2021_V7.pdf.
[22] D. Van Bulck, D. Goossens, J. r. Schonberger and M. Guajardo, "An instance data repository for the round-robin sports timetabling problem," *Management and Labour Studies*, vol. 45, pp. 184–200, 2020.
[23] D. Van Bulck and D. Goossens, "Relax-fix-optimize heuristics for time-relaxed sports timetabling," *INFOR: Information Systems and Operational Research*, vol. 59, no. 4, pp. 623–638, 2021.
[24] M. Carlsson, M. Johansson and J. Larson, "Scheduling double round-robin tournaments with divisional play using constraint programming," *European Journal of Operational Research*, vol. 259, pp. 1180–1190, 2017.
[25] D. Van Bulck, D. R. Goossens and F. C. Spieksma, "Scheduling a non-professional indoor football league: A tabu search based approach," *Annals of Operations Research*, vol. 275, pp. 715–730, 2019.
[26] G. Cocchi, A. Galligari, F. P. Nicolino, V. Piccialli, F. Schoen *et al.,* "Scheduling the Italian national volleyball tournament," *INFORMS Journal on Applied Analytics*, vol. 48, no. 3, pp. 271–284, 2018.
[27] G. Durán, M. Guajardo, F. Gutiérrez, J. Marenco, D. Sauré *et al.,* "Scheduling the main professional football league of Argentina," *INFORMS Journal on Applied Analytics*, vol. 51, pp. 361–372, 2021.
[28] D. Van Bulck, D. Goossens, J. Belien and M. Davari, "The fifth international timetabling competition (itc 2021): Sports timetabling," in *MathSport International*, pp. 117–122, 2021. https://biblio.ugent.be/publication/8716704/file/8716705
[29] M. Alzaqebah, S. Jawarneh, R. M. A. Mohammad, M. K. Alsmadi, I. Al-marashdeh *et al.,* "Hybrid feature selection method based on particle swarm optimization and adaptive local search method," *International Journal of Electrical & Computer Engineering*, vol. 11, pp. 2414–2422, 2021.
[30] A. M. Madureira, N. Sousa and I. Pereira, "Swarm intelligence for scheduling: a review," in *Second Int. Conf. on Business Sustainability (BS'2011)*, Madureira, Sousa, Pereira, 2011.
[31] J. Bansal, P. Singh, M. Saraswat, A. Verma, S. Jadon *et al.,* "Inertia weight strategies in particle swarm optimization," *IEEE*, Salamanca, Spain, pp. 640–647, 2011.
[32] N. Kumyaito, P. Yupapin and K. Tamee, "Planning a sports training program using adaptive particle swarm optimization with emphasis on physiological constraints," *BMC Research Notes*, vol. 11, pp. 1–6, 2018.

[33] W. A. Hassan, M. B. Fayek and S. I. Shaheen, "PSOSA: An optimized particle swarm technique for solving the urban planning problem," in *Proc. Int. Conf. on Computer Engineering and Systems*, Cairo, Egypt, pp. 401–405, 2006.

[34] G. Jaradat, M. Ayob and I. Almarashdeh, "The effect of elite pool in hybrid population-based meta-heuristics for solving combinatorial optimization problems," *Applied Soft Computing*, vol. 44, pp. 45–56, 2016.

[35] ITC2021-Organizers, "Sport scheduling research group," 2021. [Online]. Available: https://www.sportscheduling.ugent.be/ITC2021/instances.php. Accessed: 27/10/2021.