American Journal of Operations Research, 2019, 9, 1-14
http://www.scirp.org/journal/ajor
ISSN Online: 2160-8849
ISSN Print: 2160-8830

An Optimization Model for Exercise Scheduling

Vardges Melkonian

Department of Mathematics, Ohio University, Athens, OH, USA
Email: melkonia@ohio.edu

How to cite this paper: Melkonian, V. (2019) An Optimization Model for Exercise Scheduling. *American Journal of Operations Research*, **9**, 1-14.
https://doi.org/10.4236/ajor.2019.91001

Received: November 14, 2018
Accepted: December 26, 2018
Published: December 29, 2018

Abstract

The paper gives an optimization model for a special type of exercise session, circuit training. Circuit training involves a series of exercises performed in rotation with minimal rest. The goal of our model is to minimize the total circuit time while accomplishing a number of training goals. Our primary model is a linear integer program; additional constraints are added for muscle group and intensity requirements. The model is implemented and tested on algebraic modeling language AMPL. Our computational results show that the model can return an exercise schedule for a typical real-life data set within a few seconds.

Keywords

Discrete Optimization, Linear Programming, Planning and Scheduling, Health Services Operations Research, Sports Operations Research

1. Introduction

Having an efficient workout routine (or an exercise session) is important for improving one's health, fitness, or athletic performance. A workout might have different goals, such as gaining strength, losing weight, increasing durability, training certain muscle groups, or some combination of those. There are also different requirements and restrictions that should be taken into account when building a workout. Some examples are listed below.

- There might be limited amount of time available for the workout; on the other hand, rest time is needed between exercises.
- The muscle groups targeted by the workout should get certain amount of work; on the other hand, the muscle groups should not be overworked.
- A certain level of intensity should be maintained during the workout.

The choice of exercises to achieve those goals while satisfying the requirements is normally determined by the exerciser or her/his personal trainer based on their experience, common sense, guidelines from other sources. But designing a workout that satisfies all those requirements and achieves the goals in an optimal way might not be an easy task. In this paper, we give an optimization model that determines the exercises and their order in the workout using rigorous mathematical calculations.

We build our model for a special type of workout called circuit training [1] [2]. Circuit training involves a series of exercises performed in rotation with minimal rest. In a standard workout, one performs several sets of one exercise and has to rest muscles between sets. These rest intervals add up. On the other hand, in a circuit training one normally alternates between exercises that use different muscle groups, and that allows the rest intervals to be relatively short. Thus, circuit training saves a lot of rest time and is valuable if the exerciser wants to accomplish a number of training goals in limited amount of time. Another benefit is that a circuit consisting of only strength exercises can still have cardio benefits because of the minimal rest between exercises.

Our model chooses a set of exercises and an order for them to minimize the total circuit time while satisfying certain muscle training and intensity requirements. We use linear integer programming as our modeling technique. Binary variables are used to express different requirements about the circuit as mathematical constraints. To the best of our knowledge, there are no previous results on using linear programming for exercise scheduling. We have implemented our model on algebraic modeling language AMPL [3]. Our computational results show that the model is time-efficient in practice. It can return an exercise schedule for a typical real-life data set within a few seconds.

The paper is organized in the following way. The primary model that minimizes the total circuit time is given in Section 2. The models with muscle group and intensity constraints are given in Sections 3 and 4 correspondingly. Section 5 gives the computational results including the complete AMPL model, a discussion on collecting data for the model and the output of the model for a sample data set. Section 6 discusses possible future directions for further developing the model.

2. The Primary Integer Programming Model for Circuit Training

In this section we develop the primary model that minimizes the total circuit time. We are given a potential set of exercises E to be included in a circuit training. Each exercise has a duration d_i. We are also given inter-exercise times t_{ij}. The goal is to choose N exercises from E and an order for them such that the total circuit time is minimized.

The inter-exercise time t_{ij} is the time needed to start exercise j after finishing exercise i. There are several reasons why the inter-exercise times should be included in the model.

1) The actual time to go from station i to station j might depend on the layout

of the exercise stations in the gym.

2) It takes some time to clean up station *i* and set up station *j*.

3) It takes time to recover from the exercise on station *i* and get ready for the exercise on station *j*. It might depend on the intensity levels of both exercises *i* and *j*. Also, more time is needed if both *i* and *j* use the same muscle group.

Our problem is a variation of the Traveling Salesman Problem (TSP). The exercises correspond to the cities in TSP, while the inter-exercise times t_{ij} correspond to the distances between pairs of cities. But our problem has also the following differences from TSP.

- We have not just inter-exercise times but also a time associated with each exercise.
- Our solution is a path, not a tour.
- Not all the exercises given in E are included in the solution but only N of them.

Correspondingly, our integer program is a variation of the classical integer program for TSP [4]. Below we discuss it in detail.

Variables.

The following three sets of variables are defined.

Let

$$x_i = \begin{cases} 1 & \text{if exercise } i \text{ included in the circuit} \\ 0 & \text{otherwise} \end{cases} \quad \text{for each } \ i \in E\,;$$

$$f_{ij} = \begin{cases} 1 & \text{if exercise } j \text{ follows exercise } i \text{ in the circuit} \\ 0 & \text{otherwise} \end{cases}$$

for each $i \in E$ and $j \in E$ such that $i \neq j$; s_i be the order of exercise *i* in the circuit, for each $i \in E$.

Our constraints will provide that the orders are $1, \cdots, N$ for the exercises that are in the circuit; while the orders are 0 for the exercises not in the circuit.

Objective function.

The objective function minimizes the sum of exercise durations and inter-exercise times for those exercises that are included in the circuit:

$$\min \sum_{i \in E} d_i x_i + \sum_{i \in E,\, j \in E : i \neq j} t_{ij} f_{ij}$$

Constraints.

Below we give the constraints along with their desciptions and necessary justifications. The constraints are grouped based on the goals they accomplish.

Note that the graph structure corresponding to the circuit is a chain that has N nodes and $N-1$ arcs, more specifically, a path connecting the first exercise in the circuit to the last exercise. Each node *i* corresponds to a selected exercise. Arc $i \to j$ is on the path if exercise *j* follows exercise *i* in the circuit.

(C1) The first two constraints provide that the path has **N nodes and $N-1$ arcs**:

$$\sum_{i \in E} x_i = N \qquad (2.1)$$

$$\sum_{i\in E,\, j\in E: i\neq j} f_{ij} = N-1 \tag{2.2}$$

(C2) The next set of constraints **connect variables x_i and f_{ij}**. For any two exercises *i* and *j*, exercise *j* can follow exercise *i* in the circuit (*i.e.*, variable f_{ij} can be 1) only if both exercises *i* and *j* are included in the circuit (*i.e.*, both x_i and x_j are 1):

$$f_{ij} \le x_i \tag{2.3}$$

$$f_{ij} \le x_j \tag{2.4}$$

Constraints given in **(C3) - (C5)** provide that the set of arcs included in the solution **form a single chain**.

(C3) The following constraints provide a **path structure for the solution**. For each exercise *i*, there is at most one exercise following it and at most one exercise proceeding it in the circuit:

$$\sum_{j\in E: i\neq j} f_{ij} \le 1 \tag{2.5}$$

$$\sum_{j\in E: i\neq j} f_{ji} \le 1 \tag{2.6}$$

Note that we cannot require being equal to exactly 1 (as in TSP) because for those exercises that are not included in the circuit the sum in the left-hand side should be zero.

(C4) The following constraints provide that there is **no cycle in the solution**. For any two exercises *i* and *j*,

$$s_i - s_j + (N+1)\cdot f_{ij} \le N \tag{2.7}$$

Let us show that it prevents any cycle in the solution. Assume the opposite: there is a cycle in the solution. Then f_{ij} = 1 for all the arcs on the cycle. Thus, if we sum inequalities (2.7) for all the arcs on the cycle then all s_i variables will be canceled and we obtain:

$$(N+1)\cdot k \le N\cdot k$$

where *k* is the number of arcs in the cycle. It is a contradiction.

(C5) Next we need to show that the constraints given above provide that **the solution is a single chain**. Suppose that the solution is a collection of k disjoint chains $C_1,\cdots,C_k$. We want to show that k = 1. Suppose the number of arcs in those chains are $a_1,\cdots,a_k$. Then the number of nodes in those chains are $a_1+1,\cdots,a_k+1$. Constraint (2.2) implies that the total number of the arcs in the chains is $a_1+\cdots+a_k = N-1$. Then the total number of the nodes in the chains is $(a_1+1)+\cdots+(a_k+1) = N-1+k$. Also note that constraints (2.3), (2.4) imply that x_i = 1 for any node in the chains. Thus,

$$\sum_{i\in E} x_i = N-1+k$$

and constraint (2.1) implies that k = 1.

Summarizing, the set of nodes and arcs corresponding to f_{ij} = 1 form a single directed chain. That chain uniquely determines the order of the selected exercises in the circuit. We denote those exercises by $e_1,\cdots,e_N$ where *i* is the order of exercise e_i in the circuit.

(C6) Our next constraints provide that variables $\boldsymbol{s_i}$ **are really the correct orders of the selected exercises in the circuit.** We want to achieve that s_i's are $1,\cdots,N$ for $e_1,\cdots,e_N$; while if exercise *i* is not included in the circuit then $s_i = 0$.

First we require that for any exercise *i*,

$$0 \le s_i \le N \tag{2.8}$$

Constraint (2.8) particularly implies that $s_i - s_j \le N$ for any two exercises *i* and *j*. Thus, constraint (2.7) will be satisfied for any exercises *i*, *j* with $f_{ij} = 0$.

We also need the following constraint for any exercise *i*:

$$s_i \le N\left(\sum_{j\in E:j\ne i}\left(f_{ij}+f_{ji}\right)\right) \tag{2.9}$$

Note that an exercise is not included in the circuit if and only if all associated f_{ij} variables are 0. Constraint (2.9) provides that if all associated f_{ij} variables are 0 for exercise *i* then the corresponding $s_i = 0$ and the exercise is not in the circuit. On the other hand, if at least one associated f_{ij} variable is 1 then s_i can take any value not exceeding *N*.

Note also that $f_{ij} = 1$ for any two consecutive exercises $i = e_k$ and $j = e_{k+1}$ in the circuit. Thus, constraint (2.7) for consecutive exercises *i* and *j* yields $s_i - s_j + (N+1) \le N$ and hence $s_j \ge s_i + 1$. In other words, the *s*-value is bigger by at least 1 for the exercise that is next in the circuit. It still allows different combinations of values for s-variables of the circuit exercises (not necessarily $1,\cdots,N$ since possible values start from 0 and can also be fractional). Thus, we also need the following constraint:

$$\sum_{i\in E} s_i = \frac{N(N+1)}{2} \tag{2.10}$$

Since $1+\cdots+N = N(N+1)/2$, then the arguments above imply that the value $N(N+1)/2$ in the right-hand side of (2.10) can only be achieved by assigning $1,\cdots,N$ to the s-variables of the circuit exercises $e_1,\cdots,e_N$.

The complete primary model.

Summarizing, we have the following model:

$$\min \sum_{i\in E} d_i x_i + \sum_{i\in E,j\in E:i\ne j} t_{ij} f_{ij}$$

subject to:

$$\sum_{i\in E} x_i = N \tag{2.1}$$

$$\sum_{i\in E,j\in E:i\ne j} f_{ij} = N-1 \tag{2.2}$$

$$f_{ij} \le x_i \tag{2.3}$$

$$f_{ij} \le x_j \tag{2.4}$$

$$\sum_{j\in E:i\ne j} f_{ij} \le 1 \tag{2.5}$$

$$\sum_{j \in E: i \neq j} f_{ji} \leq 1 \tag{2.6}$$

$$s_i - s_j + (N+1) \cdot f_{ij} \leq N \tag{2.7}$$

$$0 \leq s_i \leq N \tag{2.8}$$

$$s_i \leq N\left(\sum_{j \in E: j \neq i} \left(f_{ij} + f_{ji} \right) \right) \tag{2.9}$$

$$\sum_{i \in E} s_i = \frac{N(N+1)}{2} \tag{2.10}$$

3. The Model with Muscle Group Requirements

Let M be the set of muscle groups to be trained in the circuit. Most exercises are compound and involve more than one major muscle group at a time. Typically, there is one larger muscle group that ends up doing the majority of the work, and then one or more smaller muscle groups that are recruited secondarily [5]. To account for it, we define the following two sets of parameters we will need in our constraints.

Let w_{im} be

1) 2 if $m \in M$ is a primary muscle group used in exercise $i \in E$;

2) 1 if $m \in M$ is a secondary muscle group used in exercise $i \in E$;

3) 0 if $m \in M$ is not used in exercise $i \in E$.

Let v_{im} be

1) 1 if $m \in M$ is a muscle group used in exercise $i \in E$;

2) 0 if $m \in M$ is not used in exercise $i \in E$.

Note that v_{im} is uniquely defined by w_{im}. Still it is convenient to have both v_{im} and w_{im} for giving our constraints. But only w_{im}'s need to be given as an input to the program; v_{im}'s are uniquely calculated by w_{im}'s, as we have done in our AMPL program in Section 5.

We have the following constraints for muscle groups (in addition to the constraints given in Section 2).

(M1) An important requirement for the circuit is that the **same muscle group cannot be used as the primary group in two consecutive exercises** [1]. Correspondingly, we have the following constraint. For any muscle group $m \in M$ and any two exercises $i, j \in E$ such that $w_{im} = w_{jm} = 2$,

$$f_{ij} = 0 \tag{3.1}$$

This constraint provides that if m is a primary muscle group used for both exercises *i* and *j* then *j* does not follow *i*.

(M2) It is also important to require that **each muscle group gets fully rested after being used in two consecutive exercises** (as a primary or secondary group). The following constraint achieves that. For any muscle group $m \in M$ and any three exercises $i, j, k \in E$ such that $v_{im} = v_{jm} = v_{km} = 1$,

$$f_{ij} + f_{jk} \leq 1 \tag{3.2}$$

This constraint provides that if m is used in all three exercises as a primary or a secondary group then at least one of f_{ij} and f_{jk} is 0, and thus *i*, *j*, *k* are not three consecutive exercises.

(M2g) The same muscle group cannot be used in p consecutive exercises

We can generalize constraint (3.2) the following way. The same muscle group cannot be used in *p* consecutive exercises (but it is allowed to be used in *p* − 1 consecutive exercises). For any muscle group $m \in M$ and any *p* exercises $i_1, i_2, \cdots, i_p \in E$ such that $v_{im} = 1$ for $i_1, i_2, \cdots, i_p \in E$ (*m* is used in all those exercises as a primary or a secondary group),

$$f_{i_1 i_2} + f_{i_2 i_3} + \cdots + f_{i_{p-1} i_p} \le p - 2 \tag{3.2g}$$

Since the left-hand side includes *p* − 1 binary variables the inequality above forces that at least one of those *f*-variables to be 0 and thus exercises $i_1, i_2, \cdots, i_p$ are not consecutive exercises in the circuit.

(M3) Lower and upper bounds for the total work value on each muscle group in the circuit.

We also require that the total work on a muscle group in the circuit is within certain limits LBmuscle and UBmuscle. Then we have the following constraint for each muscle group m:

$$\text{LBmuscle} \le \sum_{i \in E} w_{im} \cdot x_i \le \text{UBmuscle} \tag{3.3}$$

The middle entry in the double inequality represents the total work on the muscle group m in the circuit.

4. The Model with Intensity Requirements

Intensity is normally measured in METs [6]. One MET is considered to be the average resting energy expenditure of a typical human being. Intensity of exercise can be expressed as multiples of resting energy expenditure. An intensity of exercise equivalent to 8 METs means that the energy expenditure of the exercise is eight times the resting energy expenditure.

As a measure for intensity we use MET values of exercises. MET values for more than 800 activities are given in [6]. In our model, exercise intensities are parameters in a certain range from minMET to maxMET.

A main requirement of a circuit is that a high-intensity exercise should be followed by a low-intensity exercise [1]. To distinguish a high-intensity exercise from a low-intensity one, we define a threshold intensity value T_MET (that can be determined by the user of the model). An exercise with MET value lower than T_MET is defined to be a low-intensity exercise, and an exercise with MET value higher than T_MET is defined to be a high-intensity exercise.

To give the constraints it is convenient to introduce another binary parameter highMET which is equal 1 if an exercise is high-intensity (higher than T_MET) and 0 otherwise. This new parameter is not an input to the program; it is uniquely determined by the MET value of an exercise and the T_MET value, and is computed by the AMPL model.

We give four sets of intensity constraints, (I1) - (I4) in Subsection 4.1. Not all of these constraints should be included in the same model. The user might include some combination of the constraints depending on the goals of the target workout. The relationships among different constraints are discussed in Subsection 4.2. In Subsection 4.3 we give suggestions on combinations of constraints to be included in alternative workouts.

4.1. Intensity Constraints

The following intensity constraints can be added to the primary model given in Section 2.

(I1) An important requirement for a circuit is that **a high-intensity exercise should be followed by a low-intensity exercise** [1]. Correspondingly, we have the following constraint. For any two exercises $i, j \in E$ such that highMET[*i*] = highMET[*j*] = 1,

$$f_{ij} = 0 \tag{4.1}$$

(I2) A stronger restriction compared to (I1) would be to have **no more than one high-intensity exercise in any three consecutive exercises**. We need the following constraint to achieve that. For any three exercises $i, j, k \in E$ such that highMET[*i*] = highMET[*k*] = 1 and highMET[*j*] = 0:

$$f_{ij} + f_{jk} \le 1 \tag{4.2}$$

Constraint (4.2) prevents *i*, *j*, *k* being consecutive exercises when highMET[*i*] = highMET[*k*] = 1. But note that (4.2) does not prevent $f_{ij} = 1$ when highMET[*i*] = highMET[*j*] = 1. Thus, (4.1) should be required along with (4.2) to achieve that there is no more than one high-intensity exercise in three consecutive ones. In that sense, (4.2) complements (4.1) to achieve the goal.

(I3) A more relaxed restriction would be requiring **at least one low-intensity exercise in any three consecutive exercises**. In other words, it is allowed to have two consecutive high-intensity exercises but not three consecutive high-intensity exercises. We need the following constraint. For any three exercises $i, j, k \in E$ such that highMET[*i*] = highMET[*j*] = highMET[*k*] = 1,

$$f_{ij} + f_{jk} \le 1 \tag{4.3}$$

The constraint provides that f_{ij} and f_{jk} cannot be 1 at the same time; but just one of them can be 1.

(I3g) We can generalize constraint (I3) the following way. **At least one low-intensity exercise is required in any *p* consecutive exercises**. For any p high-intensity exercises $i_1, i_2, \cdots, i_p \in E$,

$$f_{i_1 i_2} + f_{i_2 i_3} + \cdots + f_{i_{p-1} i_p} \le p - 2 \tag{4.3g}$$

The constraint will force that the f-variables in the left-hand side cannot be all 1 at the same time; thus, $i_1, i_2, \cdots, i_p$ cannot be consecutive exercises in the circuit.

(I4) Lower and upper bounds for the average intensity level.

One can set a goal for an average intensity level through the circuit. While it is hard to achieve an exact target value for the average intensity it is possible to have lower and upper bounds for it. Let LBavgMET and UBavgMET be the lower and upper bounds for the average intensity. Then we have the following constraints:

$$\text{LBavgMET} \le \frac{1}{N}\sum_{i\in E}\text{MET}(i)\cdot x_i \le \text{UBavgMET} \tag{4.4}$$

The middle entry in the double inequality represents the average intensity of the N exercises included in the circuit.

4.2. Relationships among Constraints (I1) - (I4)

Relationships among (4.1), (4.2), *and* (4.3).

We have the following chain of logical implications for the constraints.

[(4.1) and (4.2) together: no more than 1 high-intensity in 3 consecutive exercises] →

[(4.1): no more than 1 high-intensity in 2 consecutive exercises] →

[(4.3): no more than 2 high-intensity in 3 consecutive exercises]

Relationship between (4.4) *and other constraints*.

Constraint (4.4) gives intensity limits for the whole circuit and can be given in combination with any other constraints.

4.3. Suggestions for Including the Intensity Constraints in the Model

Not all of the above constraints should be included in the model but rather some combination of those, based on the constraint relationships discussed above. We suggest the following three combinations.

Model 1 (*low-intensity workout*).

Both constraints (4.1) and (4.2) are included to force no more than 1 high-intensity in 3 consecutive exercises. No need for constraint (4.3) since it is implied by (4.1). Constraint (4.4) is included with relatively low LBavgMET and UBavgMET values.

Model 2 (*medium-intensity workout*).

Constraints (4.1) is included but (4.2) is not to force no more than 1 high-intensity in 2 consecutive exercises. No need for constraint (4.3) since it is implied by (4.1). We might have high-intensity and low-intensity exercises alternating in this kind of circuit. Constraint (4.4) is included; LBavgMET and UBavgMET values are chosen by the user to control the average intensity of the circuit.

Model 3 (*high-intensity workout*).

Constraints (4.3) is included (but constraints (4.1) and (4.2) are not) to force no more than 2 high-intensity in 3 consecutive exercises. Constraint (4.4) is included with relatively high LBavgMET and UBavgMET values.

5. Computational Results

We give the full AMPL model for the integer program developed in previous sections in Section 5.1. The model includes both muscle group and intensity constraints discussed in Sections 3 and 4 correspondingly. The intensity constraints are given for the medium-intensity workout. Section 5.2 gives guidelines and references for choosing input data for the model. We also describe a sample data set on which the model was tested. The output for the sample data set is given in Section 5.3.

5.1. The AMPL Model

```
###### SETS AND PARAMETERS #######
set Exercises;
param duration{Exercises};
# duration of each exercise
param number_of_exercises;
# number of exercises in a circuit
param time_between_exercises{i in Exercises, j in Exercises: i!=j};
# asymmetric, equal to zero if i and j are the same
###### Muscle parameters #######
set Muscles;
param work{i in Exercises, j in Muscles};
# amount of work on muscle j from exercise i
param used{i in Exercises, j in Muscles}:= if work[i,j]==0 then 0 else 1;
# is 1 muscle j is used in exercise i, otherwise is 0
param muscle_lower_limit{j in Muscles};
# lower limit on the amount of work for muscle j
param muscle_upper_limit{j in Muscles};
# upper limit on the amount of work for muscle j
###### Intensity parameters #######
param MET{i in Exercises};
# MET value for each exercise
param T_MET;
# threshold MET value for high intensity exercises
param highMET{i in Exercises}:= if MET[i] >= T_MET then 1 else 0;
# is 1 for high-intensity exercises, 0 for low-intensity exercises
param intensity_lower_limit;
# lower bound on average intensity level in the circuit
param intensity_upper_limit;
# upper bound on average intensity level in the circuit
###### VARIABLES #######
var included{Exercises} binary;
# is 1 if exercise i is included
var next_exercise{i in Exercises, j in Exercises: i!=j} binary;
# is 1 if exercise j immediately follows exercise i in the circuit
var exercise_order{Exercises} >= 0, <= number_of_exercises;
```

```
###### OBJECTIVE FUNCTION #######
minimize total_circuit_time:
sum{i in Exercises} duration[i]*included[i] +
sum{i in Exercises, j in Exercises: i!=j} time_between_exercises[i,j]*next_exercise[i,j];
###### CONSTRAINTS #######
###### Primary constraints #######
s.t. number_of_exercises_in_circuit: sum{i in Exercises}included[i] = number_of_exercises;
s.t. number_of_following_exercises_in_circuit: sum{i in Exercises, j in Exercises: i!=j} next_exercise[i,j] = number_of_exercises - 1;
s.t. follow_exercise_only_if_included1{i in Exercises, j in Exercises: i!=j}:
next_exercise[i,j] <= included[i];
# exercise i can be followed by exercise j only if exercise i is in the schedule
s.t. follow_exercise_only_if_included2{i in Exercises, j in Exercises: i!=j}:
next_exercise[i,j] <= included[j];
# exercise i can be followed by exercise j only if exercise j is in the schedule
s.t. at_most_one_exercise_follows{i in Exercises}:
sum{j in Exercises: i!=j} next_exercise[i,j] <= 1;
s.t. at_most_one_exercise_precedes{j in Exercises}:
sum{i in Exercises: i!=j} next_exercise[i,j] <= 1;
s.t. no_cycle{i in Exercises, j in Exercises: i!=j}:
exercise_order[i] - exercise_order[j] + (number_of_exercises + 1) * next_exercise[i,j] <= number_of_exercises;
s.t. dummy_zero_if_not_included {i in Exercises}:
exercise_order[i] <= number_of_exercises * sum{j in Exercises: j!=i}(next_exercise[i,j] + next_exercise[j,i]);
s.t. sum_of_orders:
sum{i in Exercises}exercise_order[i] = number_of_exercises * (number_of_exercises + 1)/2;
###### Muscle load constraints #######
s.t. no_two_primaries_in_a_row {m in Muscles, i in Exercises, j in Exercises: i!=j and work[i,m]==2 and work[j,m]==2}: next_exercise[i,j] = 0;
s.t. no_muscle_in_three_exercises_in_a_row {m in Muscles, i in Exercises, j in Exercises, k in Exercises: i!=j and j!=k and i!=k and used[i,m]==1 and used[j,m]==1 and used[k,m]==1}:
next_exercise[i,j] + next_exercise[j,k] <= 1;
s.t. Muscle_lower_limit {j in Muscles}:
sum{i in Exercises} work[i,j] * included[i] >= muscle_lower_limit[j];
s.t. Muscle_upper_limit {j in Muscles}:
sum{i in Exercises} work[i,j] * included[i] <= muscle_upper_limit[j];
###### Intensity constraints #######
s.t. no_two_high_intensities_in_a_row {i in Exercises, j in Exercises: i!=j and
```

highMET[i]=1 and highMET[j]=1}: next_exercise[i,j] = 0;

s.t. Intensity_lower_limit:

sum{i in Exercises} MET[i] * included[i] >= intensity_lower_limit * number_of_exercises;

s.t. Intensity_upper_limit:

sum{i in Exercises} MET[i] * included[i] <= intensity_upper_limit * number_of_exercises;

5.2. Input Data

In this section we give general guidelines for choosing input data for the model. We also discuss the specific values the sets and parameters take in our sample data set.

List of Exercises

There are many sources for potential exercises to be included in a circuit. For example, [7] gives exercises grouped by muscle groups, while [8] gives a good selection of bodyweight cardio exercises.

We have included the following 27 exercises in our test data: bench press, chest fly, dips, push up, pull up, shoulder press, upright rows, bent over rows, lateral pull down, biceps curl, deadlift, squat, back extension, crunch, exercise wheel, side plank, leg press, lunge, push up and rotation, jumping jack, jumping squat, burpee, mountain climber, plyometric lunges, lunge jump, step up, battle rope. Most of them are compound exercises training the target muscle groups listed below.

Number of exercises in the circuit

[1] suggests 9 - 12 exercises to be included in a circuit, while [2] recommends 3 to 15. We have 12 exercises in our sample data set to allow using more muscle groups.

Exercise durations

[1] recommends about 30 seconds for each exercise. It often depends on the number of repetitions. In our data set, the durations are between 20 and 30 seconds.

Inter-exercise times

[1] recommends no more than 15 seconds, while [2] suggests 3 - 15 seconds. In Section 2, we have listed several factors that the inter-exercise times might depend on: 1) the actual time to move from one station to another; 2) the time to clean up one station and set up the next one; 3) the recovery time (might depend on training goals). In our data set, the inter-exercise times are between 6 and 15 seconds.

Muscle groups

[5] gives most common compound and isolation exercises along with the primary and secondary muscle groups each exercise targets. We have the following 10 muscle groups in our data set: chest, shoulders, triceps, biceps, lats, trapezius, lower back, abs, quads, glutes/hamstrings.

Intensity values

MET values for more than 800 activities are given in [6]. The exercises in our data set have MET values between 6 and 10. The threshold MET value for high intensity exercises is set at 8. Of course, a user can pick a different value based on the target intensity level of the workout.

5.3. The Output for the Sample Data

We ran the AMPL model for the sample data set described in the previous subsection on the NEOS server using the solver Gurobi [9]. The solution returned by the solver included the following exercises in this specific order: 1) pull up, 2) squat, 3) push up and rotation, 4) lunge, 5) shoulder press, 6) push up, 7) biceps curl, 8) chest fly, 9) exercise wheel, 10) jumping jack, 11) bent over rows, 12) jumping squat. The workout could be completed in 5 minutes and 30 seconds.

The solution was returned within a few seconds. Our sample data set (12 exercises to be selected out of 27) has a typical size of a real-life exercise selection problem. Thus, the model is time-efficient and can be used to make real-time decisions for choosing exercises very quickly.

6. Future Directions

We have developed an integer programming model for a special type of exercise session, circuit training. The model minimizes the total training time while accomplishing several training goals, such as satisfying certain muscle group and intensity requirements. The computational results show that the model is time-efficient in practice. Our model is deterministic: it assumes that the values assigned to all the parameters are known constants. But some parameters might possess inherent randomness in practical situations, and thus probabilistic formulations might be better suited for modeling those situations. Below we discuss several future directions for further developing the model.

Real-time schedule update

Sometimes it might be necessary to update the preset schedule. A possible reason could be that one or more exercise stations are unavailable (busy or out of order). The user can run the model by giving a new input to the program stating that those exercise stations are unavailable. The model will return an updated schedule as an output. As discussed in Section 5.3, the output can be returned within a few seconds for typical problems. Thus, it is an efficient way of rescheduling the circuit.

Stochastic models for exercise scheduling

Another variation of the original model could be taking into account the high usage of some machines. Statistics can be collected about the usage of machines at different time periods of a day. It can be as a percentage of time the machine is used at a particular time (periods of 15, 20 or 30 minutes). Thus, the user can have the probabilities of machines' availability at a given time of a day. Then the objective function could be minimizing not the actual circuit time but the expected time to finish the circuit.

Models for other exercise routines

Our model is for circuit training. But integer programming can be used for other type of workouts too. We give mathematical constraints for muscle group and intensity requirements. There might be other type of workout requirements that can be translated into mathematical constraints. Our main goal is minimizing the workout time. Another possibility is to have a fixed maximum time allotted to the workout, and to maximize a certain workout benefit (for example, burned calories) in the given time.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Klika, B. and Jordan, S. (2013) High-Intensity Circuit Training Using Body Weight: Maximum Results with Minimal Investment. *ACSM'S Health & Fitness Journal*, **17**, 8-13. https://doi.org/10.1249/FIT.0b013e31828cb1e8

[2] Johnson, J. (2017) Circuit Training: Best Time between Sets. http://www.livestrong.com/article/342216-circuit-training-best-time-between-sets/

[3] The AMPL Website. http://www.ampl.com/

[4] Papadimitriou, C.H. and Steiglitz, K. (1998) Combinatorial Optimization: Algorithms and Complexity. *Dover Publications,* 308-309.

[5] Compound Exercises vs Isolation Exercises: Which Is Best? http://www.aworkoutroutine.com/compound-exercises-vs-isolation-exercises/

[6] Ainsworth, B.E., Haskell, W.L., Hermann, S.D., Meckes, N., Basset Jr., D.R., Tudor-Locke, C., Greer, J.L., Vezina, J., Whitt-Glover, M.C. and Leon, A.S. (2011) 2011 Compendium of Physical Activities: A Second Update of Codes and MET Values. *Medicine and Science in Sports and Exercise*, **43**, 1575-1581. https://doi.org/10.1249/MSS.0b013e31821ece12

[7] A List of the Best Weight Training Exercises for Each Muscle Group. http://www.aworkoutroutine.com/list-of-exercises-for-each-muscle-group/

[8] 33 Cardio-Based Bodyweight Exercises. https://greatist.com/fitness/cardio-bodyweight-exercises

[9] The Gurobi Website. https://neos-server.org/neos/solvers/lp:Gurobi/AMPL.html