Proceedings of the 7th WSEAS International Conference on Evolutionary Computing, Cavtat, Croatia, June 12-14, 2006 (pp39-44)

Computer-Based Dietary Menu Planning

BARBARA KOROUŠIĆ SELJAK
Computer Systems Department
Jožef Stefan Institute
Jamova 39, 1000 Ljubljana
SLOVENIA
http://csd.ijs.si/korousic/korousic.html

Abstract: - Nutrition is one of many disciplines, where intractable optimization problems naturally arise and require special attention of meta-heuristic methods. The problem of generating an optimal dietary menu that considers nutrient and non-nutrient requirements can be solved by evolutionary algorithms in an efficient and effective way. The menu optimality may be defined by several factors, such as cost, food functionality and season, as well as aesthetic standards for taste, consistency, color, temperature, shape, and method of preparation. In this paper, we present a multi-objective and multi-constrained evolutionary algorithm for planning and optimization of daily menus. It is based on the Elitist Non-Dominated Sorting Genetic Algorithm [2] and the Baldwinian repair algorithm [6]. The paper also presents an example of planning a daily menu by this method to demonstrate the approach.

Key-Words: - multi-objective and multi-constrained optimization, knapsack problem, evolutionary algorithm, Elitist Non-Dominated Sorting Genetic Algorithm, repair algorithm, dietary-menu planning.

1 Introduction

Dietary menu planning is an intractable optimization problem because of many constraints and objectives. People decide what to eat in highly personal way often based on behavioral or social motives. As the knowledge about nutrition and its health impact grows, decisions become more complex and a support of a computer-based method is required.

The dietary menu planning problem is constrained by nutrient and non-nutrient requirements. Objectives may include cost, seasonal quality, functional quality, and aesthetic standards for taste, consistency, color, temperature, shape, and method of preparation. This is a linear-programming problem because it can be formulated as minimizing the objectives, given limited resources and competing constraints. We can specify the objectives as linear functions and the constraints as linear equalities or inequalities. A simplified version of the problem considering basic nutrient requirements and one objective of cost was firstly solved using a calculator in 1941 [1]. Since then the linear programming methods have improved significantly, producing cost-optimized menus. However, difficulties have been encountered in using numerical representations for qualitative factors, such as taste, consistency, color, temperature, shape, and method of preparation.

In this paper, we present an evolutionary-computation approach to the dietary-menu planning problem. We applied the Elitist Non-Dominated Sorting Genetic Algorithm (NSGA-II) [2] to generate daily menus considering constraints on nutrient and non-nutrient requirements and nine objectives of low cost, high seasonal quality and functionality, and low deviations from uniformly distributed aesthetic standards for taste, consistency, color, temperature, shape, and method of preparation into account.

2 Problem Formulation

Mathematically, dietary menu planning reduces to a multi-objective and multi-constrained (multi-dimensional) knapsack problem (MDKP) that is easy to formulate, yet its decision problem is NP-complete. It means that only by using a heuristic optimization method a solution can be found quickly (in a polynomial time).

We define the problem as follows: *Given food items of different values and volumes, find the most valuable composition that fits in a knapsack of fixed volumes. Values are defined subjectively with respect to food functionality, seasonal availability, cost, taste, consistency, color, temperature, shape and method of preparation. Knapsack volumes are defined by the weakly correlated diet-planning principles*.

Food items are selected from a nutritional database that integrates nutritional data of more than 7000 (national and world-wide) foods. We consider the D-A-CH diet-planning principles established by the European nutrition societies [3].

Many other real-world problems can be formulated as a MDKP, for example, the capital budgeting problem, allocating processors in a distributed computer system, project selection, and cutting stock problem.

2.1 Multi-dimensional Knapsack Problem

We are given a knapsack of m volumes C_k, $k = 1, 2, \ldots m$, and n meal items. Each item i has nine values $v_{ik} \in I^+, v_{ik} > 0,\ k = 1,2,...9$, and m volumes $\omega_{ik} \in R^+, \omega_{ik} > 0,\ k = 1,2,...m$, one for each capacity. We are looking for a composition of t items, $t \le n$, such that $\sum_{i=1}^{t} \omega_{ik} x_i \Phi C_k$ (Φ can be $\le$ or $\ge$, $k = 1,2,...m$, $t \le n$), and for which the total values $\sum_{i=1}^{n} v_{ik} x_i,\ k = 1,2$ are maximized, while $\sum_{i=1}^{n} v_{ik} x_i,\ k = 3$ and

$$\left(\sum_{j=1}^{n_{a_l}} \left| \sum_{i=1}^{n} h_{lj}(x_i) - \frac{\sum_{i=1}^{n} h(x_i)}{n_{a_l}} \right| \right) - \sum_{i=1}^{n} h(x_i),\ l = 4,5,...9,$$

are minimized, where n_{a_l} is the number of possible states of an aesthetic standard l. The functions used in the above objective function are defined as follows:

$$h_{lj}(x_i) = \begin{cases} 0, if\ x_i = 0 \\ 1, if\ x_i > 0 \wedge v_{il} = \text{j} \end{cases},\ h(x_i) = \begin{cases} 0,\ if\ x_i = 0 \\ 1,\ otherwise \end{cases},$$

$i = 1,2,...,n, l = 4,5,...9$. The parameter x_i is a binary variable.

2.2 Metods for Solving MDKPs
Exact algorithms that deliver optimum solutions to multi-dimensional knapsack problems in pseudo-polynomial time are based on the branch-and-bound and the dynamic programming approaches. On the other hand, heuristic methods with time complexity bounded by a polynomial in the size parameters of the problem have been known for many decades. A comprehensive review of the multi-constrained 0-1 knapsack problem and the associated heuristic algorithms is given by Chu and Beasley [4]. Some of the ideas are also applicable to non-0-1 MDKPs.

3 Problem Solution
In our case, a knapsack denotes a daily menu that is composed of five meals. By default meals include a breakfast, a morning snack, a lunch, an afternoon snack, and a dinner. However, this composition does not bias the method and can be modified to suit the specific menu-planning problem.
Because the menu planning is a multi-dimensional problem with many infeasible solutions (that violate at least one constraint), we decided to solve it by evolutionary algorithms that have been shown to be well suited for solving problems characterized by local minima. Although evolutionary algorithms search through an arbitrary search space by random decisions, they are far from random search routines. Modeling the random decisions using Markov chain analysis, it was shown that evolutionary algorithms can converge to globally optimum solutions [5].

3.1 Evolutionary Algorithm
We have applied a powerful evolutionary algorithm NSGA-II that is well suited for finding multiple trade-off optimal solutions to multi-objective and multi-constrained problems. The trade-off optimal solutions cannot be improved upon without hurting at least one of the objectives, and are called Pareto-optimal solutions [2].

3.2 Encoding
We encode candidate solutions of the daily menu-planning problem by integer-valued coding. In our representation, a chromosome contains five data, one for each meal. They carry the information (an identification code) about the meals. Considering approximately 15-100 meals per each meal group, the number of possible solutions is roughly estimated to 50^5.

3.3 Populations
In our implementation, the algorithm NSGA-II starts the evolution from a population of either random candidate solutions or solutions known from experience. The population's size is N and remains constant over all generations.

3.4 Fitness evaluation
In each generation, the fitness of the population is evaluated using the following objective functions:

$$f_k(\vec{x}) = \frac{1}{\sum_{i=1}^{n} v_{ik} x_i},\ k = 1,2,$$

$$f_3(\vec{x}) = \sum_{i=1}^{n} v_{i3} x_i,$$

$$f_l(\vec{x}) = \left(\sum_{j=1}^{n_{a_l}} \left| \sum_{i=1}^{n} h_{lj}(x_i) - \frac{\sum_{i=1}^{n} h(x_i)}{n_{a_l}} \right| \right) - \sum_{i=1}^{n} h(x_i), \quad (1)$$

$$h_{lj}(x_i) = \begin{cases} 0, if\ x_i = 0 \\ 1, if\ x_i > 0 \wedge v_{il} = \text{j} \end{cases},$$

$$h(x_i) = \begin{cases} 0,\ if\ x_i = 0 \\ 1,\ otherwise \end{cases},\ i = 1,2,...,n,\ l = 4,5,...9,$$

where v_{i1} denotes the functionality of the meal item i, v_{i2} its quality in the season, v_{i3} the cost, v_{i4} the taste, v_{i5} the consistency, v_{i6} the color, v_{i7} the temperature, v_{i8} the shape, v_{i9} the method of preparation, and n_{a_l} the number of possibilities for the lth aestetic standard. The meal parameters $v_{ij}, j = 1,...,9$, are estimated from the parameters of food items included into a given meal i. The aim of the evolutionary algorithm is to minimize the objective functions of (1).

3.5 Infeasible Solutions
A candidate solution may be highly fit but infeasible if it

violates at least one of the following problem constraints:

- The energy provided by the meal has to be within the lower limit and the upper limit:

$$g_1(\vec{x}) = \sum_{i=1}^{5} \omega_{iE} x_i \geq 0{,}9E,$$
$$g_2(\vec{x}) = \sum_{i=1}^{5} \omega_{iE} x_i \leq 1{,}1E, \tag{2}$$

  where ω_{iE} denotes the energy of a given meal item *i*, x_i a binary value of the item *i*, and *E* the daily requirement for energy.

- The basic nutrients (i.e., proteins, lipids and carbohydrates) need to be balanced:

$$g_3(\vec{x}) = \sum_{i=1}^{5} \omega_{iP} 4x_i \geq 0{,}1E,$$
$$g_4(\vec{x}) = \sum_{i=1}^{5} \omega_{iP} 4x_i \leq 0{,}15E,$$
$$g_5(\vec{x}) = \sum_{i=1}^{5} \omega_{iL} 9x_i \geq 0{,}15E,$$
$$g_6(\vec{x}) = \sum_{i=1}^{5} \omega_{iL} 9x_i \leq 0{,}3E, \tag{3}$$
$$g_7(\vec{x}) = \sum_{i=1}^{5} \omega_{iC} 4x_i \geq 0{,}55E,$$
$$g_8(\vec{x}) = \sum_{i=1}^{5} \omega_{iC} 4x_i \leq 0{,}75E,$$

  where ω_{iP}, ω_{iL}, ω_{iC} denote the quantity of proteins, lipids and carbohydrates, respectively, in a given meal item *i*. Because the quantities are expressed in grams, conversion factors (4 for proteins and carbohydrates, and 9 for lipids) are required to attain to calories. We applied usual balancing factors for adults (0,1 and 0,15 for proteins, 0,15 and 0,3 for lipids, and 0,55 and 0,75 for carbohydrates) but may be changed.

- Simple sugars should account for only 10 percent or less of the total energy intake:

$$g_9(\vec{x}) = \sum_{i=1}^{5} \omega_{iS} 4x_i \leq 0{,}1E. \tag{4}$$

- The daily intake of saturated fatty acids should be limited to 10 percent of the total energy intake:

$$g_{10}(\vec{x}) = \sum_{i=1}^{5} \omega_{iF} 9x_i \leq 0{,}1E. \tag{5}$$

- The recommended daily intake of the dietary fiber is 10 grams per 1000-kalorie energy intake and should not exceed 40 grams:

$$g_{11}(\vec{x}) = \sum_{i=1}^{5} \omega_{iV} x_i \geq 10\frac{E}{1000}$$
$$g_{12}(\vec{x}) = \sum_{i=1}^{5} \omega_{iV} x_i \leq 40 \tag{6}$$

- The minimum and the maximum sodium requirements for adults in Slovenia are set at 550 and 2400 milligrams per day, respectively [9]:

$$g_{13}(\vec{x}) = \sum_{i=1}^{5} \omega_{iNa} x_i \geq 500$$
$$g_{14}(\vec{x}) = \sum_{i=1}^{5} \omega_{iNa} x_i \leq 2400 \tag{7}$$

3.5.1 Repair Algorithm

We decided to repair a certain part of infeasible solutions in each generation to speed up the procedure of finding an optimal solution. We applied the Baldwinian repair, where replacement is used only to evaluate the fitness values of each solution [6]. Certain critical meals were 'replaced' with more appropriate ones. Critical meals do not satisfy the constraints on major food groups (i.e., breads, cereal, rice, and pasta / vegetables / fruits / milk, yogurt, and cheese / meat, poultry, fish, beans, eggs, and nuts / fats, oils, and sweets). Namely, a daily menu has to be composed of a certain number of foods from each major food group. There may be limitations on frequency of red meat, fish, potatoe etc.

3.6 Selection

In order to form a new generation, a binary tournament approach is applied. Solutions from both – the parent and the previous offspring – populations can take part in the tournament if they are sorted by two attributes, i.e., a non-domination rank and a crowding distance. Initially, the offspring population is an empty set.

First, solutions are sorted by the fast non-dominated sorting approach of the NSGA-II. In this approach, best non-dominated solutions become elites of identical importance, forming Pareto-optimal fronts. Solutions are non-dominated if none solution is better than the others with respect to all equally important objectives. Because every solution from the population is checked with a partially filled population for domination, the maximum time complexity of the non-dominated sorting approach is $O(4mN^2)$, where $2N$ is the population size and *m* the number of objectives.

Then, solutions are sorted according to their crowding distances. A crowding distance is a measure of the search space around a chosen solution, which is not occupied by any other solution in the population. Its computation requires sorting of the populations according to each objective function value in their ascending order of magnitude. Thereafter, for each objective function, the boundary solutions (solutions with the smallest and the largest function values) are assigned an infinite distance value. All other solutions are assigned a distance value equal to the absolute difference in the function values of two adjacent solutions. This calculation is continued with other objective functions. The overall crowding distance value is calculated as the sum of individual distance values corresponding to each objective. The maximum time complexity of this sorting approach is $O(2mN \log 2N)$.

A solution *i* wins a tournament with another solution *j* if both solutions are feasible or infeasible and any of the following conditions are true:

- It has a better non-domination rank than solution *j*.
- Having the same non-domination rank, it has better crowding distance than solution *j*.

The first condition makes sure that solution *i* lies on a better Pareto front than solution *j*. The second condition resolves the tie of both solutions being on the same non-dominated front by deciding on their crowded distance. The one residing in less crowded area wins. If one solution is feasible and the other is not, the feasible one wins the tournament.

Performing *N* tournaments, we obtain a new parent population of size *N*. Other *N* solutions from the least important Pareto fronts having a smaller crowding distance are discarded.

3.7 Crossover and Mutation

Solutions from the new parent population are mated pair-wise (using a two-point crossover operator) and mutated to create a new offspring population of size *N*. This completes one iteration.

Mutation is performed on randomly selected elements of the chromosome. The mutation rate is set to be a small value that linearly decreases with iterations. The selected elements are mutated by replacing the meal with a meal of the same type, or a food item from certain randomly selected food groups (such as fruits, salads, breads, oils) with alternatives.

4 Evaluation of the Method

As a demonstration, we applied the NSGA-II to a problem of planning daily menus for people without specific dietary requirements in a local hospital.

In Tab. 1, we list the parameters used to generate daily menus by the NSGA-II. We ran the algorithm for 25 times to obtain the experimental results presented in Tab.2. In Fig.1, a part of the feasible search space, whose shape is depicted for three objectives but actually modified by nine objectives, is presented. In Fig.2, it can be seen how solutions in point of cost, seasonal quality, and functionality converge to the Pareto-optimal solutions with time (evaluations). Tab.3 gives a subset of the analysis results of a daily menu generated by the NSGA-II.

5 Conclusion

In this paper, we have presented a computer-based method for daily menu planning that considers many constraints and nine objectives. It is based on the NSGA-II evolutionary algorithm that selects an optimal combination of meals to form a daily menu of low cost, high seasonal and functional quality, and high aesthetic standards. The method is used within a dietary menu planning tool applied as a Web-based application [7] that incorporates the strength of meta-heuristic evolutionary algorithms and linear programming methods. We have also presented the repairing algorithm that was required for maintaining the feasibility of solutions. The experimental results showed that the approach distinguishes with efficiency and effectiveness.

As the problem of dietary menu-planning belongs to the multi-dimensional knapsack problems, the method could be useful for other intractable problems from this wide group.

Parallel implementation of the multi-level NSGA-II for dietary menu planning deserves future attention.

Table 1. The parameters of the evolutionary algorithm.

Parameter	The daily-menu level
Chromosome length	5
Population size	100
Crossover probability	0,7
Mutation probability	0,2-0,01
Selection type	Binary tournament selection
Crossover type	Two-point crossover
Mutation type	Linear descending mutation
Number of iterations	90

Table 2. The experimental results of the evolutionary algorithm.

Percentage of infeasible solutions in each new generation	89 %
Percentage of successfully repaired infeasible solutions	65 %

	Cost (€)	Quality in season	Functional quality
Best result	3,08	18	0
Median	9,7	28	6
Worst result	22,8	48	12
Mean value	9,7	28,3	5,8
Standard deviation	3,1	4,7	3,4

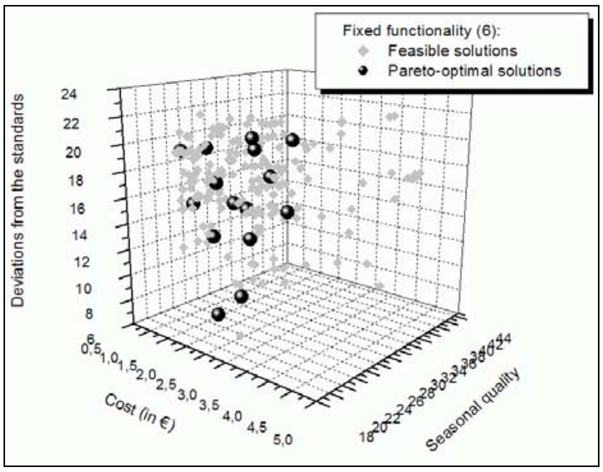

Figure 1. A part of the problem's search space.

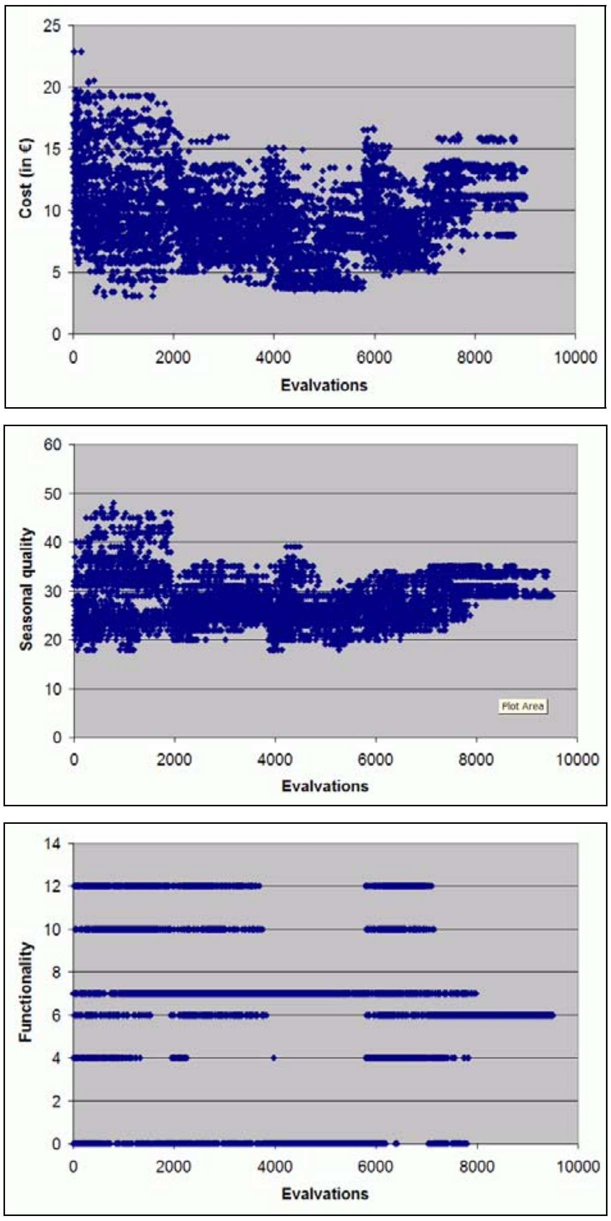

Figure 2. The objective function values.

Table 3. The analysis results of a low-cost and high-quality appetizing weekly menu generated by computer.

	Mean daily values	DACH	Goal achieved
Energy (kcal)	2036	2000	102 %
Proteins (% of energy)	16	> 10	✓
Lipids (% of energy)	28	< 30	✓
Carbohydrates (% of energy)	56	> 50	✓
Simple sugars % of energy)	4,5	< 10	✓
Saturated fats (% of energy)	6,6	< 10	✓
Ratio of omega-6 to omega-3 fatty acids	3,9:1	5:1	✓
Dietary fibre (g)	33,6	30-40	✓
Cholesterol (mg)	160	300	✓
Sodium (mg)	2500	550-2400	104 %
Breads, cereal, rice, and pasta (no. of units)	11,2	11	102 %
Vegetables (no. of units)	4,7	5	94 %
Fruits (no. of units)	3	3	100 %
Milk, yogurt, and cheese (no. of units)	2	2	100 %
Meat, poultry, fish, beans, eggs, and nuts (no. of units)	1,9	2	95 %

6 Acknowledgments

This work represents joint research between the Computer Systems Department, at the Jožef Stefan Institute, and the Department of Food Science and Technology, at the Biotechnical Faculty, University of Ljubljana. The author gratefully acknowledges the contributions of Prof. Dr. Dražigost Pokorn, and the support provided by the Slovenian Ministry of Health during the course of this work.

References:

[1] E. F. Eckstein, *Menu Planning*, Third Edition, AVI Publishing Company, Inc., Westport, Connecticut: 1983.

[2] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*, John Wiley & Sons, Ltd.: 2001, ISBN 0-471-87339-X.

[3] *Die Referenzwerte für die Nährstoffzufuhr, D-A-CH Referenzwerte der DGE, ÖGE, SGE/SVE*, 1. Auflage, Umschau Braus Gmbh, Verlagsgesellschaft, Frankfurt/Main: 2002, ISBN 3-8295-7114-5.

[4] P. C. Chu and J. E. Beasley, "A Genetic Algorithm for the Multidimensional Knapsack Problem", *Journal of heuristics*, 4, Kluwer Academic Publishers: 1988, pp. 63-86.

[5] C. L. Karr, I. Yakushin and K. Nicolosi, "Solving inverse initial-value, boundary-value problems via genetic algorithm", *Engineering Applications of Artificial Intelligence*, 13(6), 2000, pp. 625-633.

[6] H. Ishibuchi, S. Kaige and K. Narukawa, "Comparison between Lamarckian and Baldwinian Repair on Multiobjective 0/1 Knapsack Problems", In *C. A. Coello Coello et al (Eds.): EMO 2005, Lecture Notes in Computer Science 3410*, Springer-Verlag Berlin Heidelberg: 2005, pp. 370-385.

[7] http://optijed.ijs.si/ (in Slovene).