B, Schlenkrich M, Smith JC, Stote R, Straub J, Watanabe M, Wiórkiewicz-Kuczera J, Yin D, Karplus M (1998) All-atom empirical potential for molecular modeling and dynamics studies of proteins. J Phys Chem B 102:3586–3616
20. Momany FA, McGuire RF, Burgess AW, Scheraga HA (1975) Energy parameters in polypeptides. VII. Geometric parameters, partial atomic charges, nonbonded interactions, hydrogen bond interactions, and intrinsic torsional potentials for the naturally occurring amino acids. J Phys Chem 79:2361–2381
21. Némethy G, Gibson KD, Palmer KA, Yoon CN, Paterlini G, Zagari A, Rumsey S, Scheraga HA (1992) Energy parameters in polypeptides. 10. Improved geometrical parameters and nonbonded interactions for use in the ECEPP/3 algorithm, with application to proline-containing peptides. J Phys Chem 96:6472–6484
22. Rajgaria R, McAllister SR, Floudas CA (2006) Development of a novel high resolution C^{α}–C^{α} distance dependent force field using a high quality decoy set. Proteins: Structure, Function, Bioinformatics 65:726–741
23. Samudrala R, Moult J (1998) An all-atom distance-dependent conditional probability discriminatory function for protein structure prediction. J Molec Biol 275:895–916
24. Scott WRP, Hunenberger PH, Trioni IG, Mark AE, Billeter SR, Fennen J, Torda AE, Huber T, Kruger P, VanGunsteren WF (1997) The GROMOS biomolecular simulation program package. J Phys Chem A 103:3596–3607
25. Subramaniam S, Tcheng DK, Fenton J (1996) A knowledge-based method for protein structure refinement and prediction. In: States D, Agarwal P, Gaasterland T, Hunter L, Smith R (eds) Proceedings of the 4th International Conference on Intelligent Systems in Molecular Biology. AAAI Press, Boston, pp 218–229
26. Tanaka S, Scheraga HA (1976) Medium- and long-range interaction parameters between amino acids for predicting three-dimensional structures of proteins. Macromolecules 9:945–950
27. Tobi D, Elber R (2000) Distance-dependent, pair potential for protein folding: Results from linear optimization. Proteins: Structure, Function, Bioinformatics 41:40–46
28. Tobi D, Shafran G, Linial N, Elber R (2000) On the design and analysis of protein folding potentials. Proteins: Structure, Function, Bioinformatics 40:71–85
29. Zhang Y, Skolnick J (2004) Automated structure prediction of weakly homologous proteins on a genomic scale. Proc Natl Acad Sci USA 101:7594–7599

Domination Analysis in Combinatorial Optimization

GREGORY GUTIN
Department of Computer Science, Royal Holloway, University of London, Egham, UK

MSC2000: 90C27, 90C59, 68Q25, 68W40, 68R10

Article Outline

Keywords and Phrases

Domination analysis; Domination number; Domination ratio

Introduction

Exact algorithms allow one to find optimal solutions to NP-hard combinatorial optimization (CO) problems. Many research papers report on solving large instances of some NP-hard problems (see, e. g., [25,27]). The running time of exact algorithms is often very high for large instances (many hours or even days), and very large instances remain beyond the capabilities of exact algorithms. Even for instances of moderate size, if we wish to remain within seconds or minutes rather than hours or days of running time, only heuristics can be used. Certainly, with heuristics, we are not guaranteed to find optimum, but good heuristics normally produce near-optimal solutions. This is enough in most applications since very often the data and/or mathematical model are not exact anyway.

Research on CO heuristics has produced a large variety of heuristics especially for well-known CO problems. Thus, we need to choose the best ones among them. In most of the literature, heuristics are compared in computational experiments. While experimental analysis is of definite importance, it cannot cover all possible families of instances of the CO problem at hand and, in particular, it practically never covers the hardest instances.

Approximation Analysis [3] is a frequently used tool for theoretical evaluation of CO heuristics. Let $\mathcal{H}$ be a heuristic for a combinatorial minimization problem P and let $\mathcal{I}_n$ be the set of instances of P of size n. In approximation analysis, we use the performance ratio $r_H(n) = \max\{f(I)/f^*(I) : I \in \mathcal{I}_n\}$, where $f(I)(f^*(I))$ is the value of the heuristic (optimal) solution of I. Unfortunately, for many CO problems, estimates for $r_{\mathcal{H}}(n)$ are not constants and provide only a vague picture of the quality of heuristics. Moreover, even constant performance ratio does not guarantee that the heuristic often outputs good-quality solutions, see, e. g., the discussion of the DMST heuristic below.

Domination Analysis (DA) (for surveys, see [22,24]) provides an alternative and a complement to approximation analysis. In DA, we are interested in the domination number or domination ratio of heuristics. *Domination number* (*ratio*) of a heuristic H for a combinatorial optimization problem P is the maximum number (fraction) of all solutions that are not better than the solution found by H for any instance of P of size n. In many cases, DA is very useful. For example, we will see later that the greedy algorithm has domination number 1 for many CO problems. In other words, the greedy algorithm, in the worst case, produces the unique worst possible solution. This is in line with latest computational experiments with the greedy algorithm, see, e. g., [25], where the authors came to the conclusion that the greedy algorithm 'might be said to self-destruct' and that it should not be used even as 'a general-purpose starting tour generator'.

The *Asymmetric Traveling Salesman Problem (ATSP)* is the problem of computing a minimum weight tour (Hamilton cycle) passing through every vertex in a weighted complete digraph K_n^* on n vertices. The *Symmetric TSP (STSP)* is the same problem, but on a complete undirected graph. When a certain fact holds for both ATSP and STSP, we will simply speak of *TSP*. Sometimes, the maximizing version of TSP is of interest, we denote it by *Max TSP*.

APX is the class of CO problems that admit polynomial time approximation algorithms with a constant performance ratio [3]. It is well known that while Max TSP belongs to APX, TSP does not. This is at odds with the simple fact that a 'good' approximation algorithm for Max TSP can be easily transformed into an algorithm for TSP. Thus, it seems that both Max TSP and TSP should be in the same class of CO problems. The above asymmetry was already viewed as a drawback of performance ratio already in the 1970's, see, e. g. [11,28,33]. Notice that from the DA point view Max TSP and TSP are equivalent problems.

Zemel [33] was the first to characterize measures of quality of approximate solutions (of binary integer programming problems) that satisfy a few basic and natural properties: the measure becomes smaller for better solutions, it equals 0 for optimal solutions and it is the same for corresponding solutions of equivalent instances. While the performance ratio and even the relative error (see [3]) do not satisfy the last property, the parameter *1-r*, where *r* is the domination ratio, does satisfy all of the properties.

Local Search (LS) is one of the most successful approaches in constructing heuristics for CO problems. Recently, several researchers investigated LS with Very Large Scale Neighborhoods (see, e. g., [1,12,24]). The hypothesis behind this approach is that the larger the neighborhood the better quality solution are expected to be found [1]. However, some computational experiments do not support this hypothesis; sometimes an LS with small neighborhoods proves to be superior to that with large neighborhoods. This means that some other parameters are responsible for the relative power of neighborhoods. Theoretical and experimental results on TSP indicate that one such parameter may well be the domination number of the corresponding LS.

In our view, it is advantageous to have bounds for both performance ratio and domination number (or, domination ratio) of a heuristic whenever it is possible. Roughly speaking this will enable us to see a 2D rather than 1D picture. For example, consider the double minimum spanning tree heuristic (DMST) for the Metric STSP (i. e., STSP with triangle inequality). DMST starts from constructing a minimum weight spanning tree T in the complete graph of the STSP, doubles every edge in T, finds a closed Euler trail E in the 'double' T, and cancels any repetition of vertices in E to obtain a TSP tour H. It is well-known and easy to prove that the weight of H is at most twice the weight of the optimal tour. Thus, the performance ratio for DMST is bounded by 2. However, Punnen, Margot and Kabadi [29] proved that the domination number of DMST is 1. Interestingly, in practice DMST often performs much worse than the well-known 2-Opt LS heuristic. For 2-Opt LS we cannot give any constant approximation guarantee, but the heuristic is of very large domination number [29].

The above example indicates that it makes sense to use DA to rank heuristics for the CO problem under consideration. If the domination number of a heuristic $\mathcal{H}$ is larger than the domination of a heuristic $\mathcal{H}'$ (for all or 'almost all' sizes n), we may say that $\mathcal{H}$ is better than $\mathcal{H}'$ in the worst case (from the DA point of view). Berend, Skiena and Twitto [10] used DA to rank some well-known heuristics for the Vertex Cover problem (and, thus, the Independent Set and Clique problems). The three problems and the heuristics will be defined in the corresponding subsection of the Cases section. Ben-Arieh et al. [7] studied three heuristics for the *Generalized TSP*: the vertices of the complete digraph are partitioned into subsets and the goal is to find a minimum weight cycle containing exactly one vertex from each subset. In the computational experiment in [7] one of the heuristics was clearly inferior to the other two. The best two behaved very similarly. Nevertheless, the authors of [7] managed to 'separate' the two heuristics by showing that one of the heuristics was of much larger domination number.

One might wonder whether a heuristic $\mathcal{A}$, which is significantly better that another heuristic $\mathcal{B}$ from the DA point of view, is better that $\mathcal{B}$ in computational experiments. In particular, whether the ATSP greedy algorithm, which is of domination number 1, is worse, in computational experiments, than any ATSP heuristic of domination number at least $(n-2)!$? Generally speaking the answer to this natural question is negative. This is because computational experiments and DA indicate different aspects of quality of heuristics. Nevertheless, it seems that many heuristics of very small domination number such as the ATSP greedy algorithm perform poorly also in computational experiments and, thus, cannot be recommended to be widely used in computational practice.

The rest of the entry is organized as follows. We give additional terminology and notation in the section Definitions. In the section Methods, we describe two powerful methods in DA. In the section Cases, we consider DA results for some well-known CO problems.

Definitions

Let $\mathcal{P}$ be a CO problem and let $\mathcal{H}$ be a heuristic for $\mathcal{P}$. The *domination number* $\text{domn}(\mathcal{H}, \mathcal{I})$ of $\mathcal{H}$ *for an instance* $\mathcal{I}$ of $\mathcal{P}$ is the number of solutions of $\mathcal{I}$ that are not better than the solution s produced by $\mathcal{H}$ including s itself. For example, consider an instance $\mathcal{T}$ of the STSP on 5 vertices. Suppose that the weights of tours in $\mathcal{T}$ are 2,5,5,6,6,9,9,11,11,12,12,15 (every instance of STSP on 5 vertices has 12 tours) and suppose that the greedy algorithm computes the tour T of weight 6. Then $\text{domn}(\texttt{greedy}, \mathcal{T}) = 9$. In general, if $\text{domn}(\mathcal{H}, \mathcal{I})$ equals the number of solutions in $\mathcal{I}$, then $\mathcal{H}$ finds an optimal solution for $\mathcal{I}$. If $\text{domn}(\mathcal{H}, \mathcal{I}) = 1$, then the solution found by $\mathcal{H}$ for $\mathcal{I}$ is the unique worst possible one.

The *domination number* $\text{domn}(\mathcal{H}, n)$ of $\mathcal{H}$ is the minimum of $\text{domn}(\mathcal{H}, \mathcal{I})$ over all instances $\mathcal{I}$ of size n. Since the ATSP on n vertices has $(n-1)!$ tours, an algorithm for the ATSP with domination number $(n-1)!$ is exact. The domination number of an exact algorithm for the STSP is $(n-1)!/2$. If an ATSP heuristic $\mathcal{A}$ has domination number equal 1, then there is an assignment of weights to the arcs of each complete digraph K_n^*, $n \geq 2$, such that $\mathcal{A}$ finds the unique worst possible tour in K_n^*.

While studying TSP we normally consider only feasible solutions (tours), for several other problems some authors take into consideration also infeasible solutions [10]. One example is the *Maximum Independent Set* problem, where given a graph G, the aim is to find an independent set in G of maximum cardinality. Every non-empty set of vertices is considered to be a solution by Berend, Skiena and Twitto [10]. To avoid dealing with infeasible solutions (and, thus, reserving the term 'solution' only for feasible solutions) we also use the notion of the *blackball number* introduced in [10]. The *blackball number* $\text{bbn}(\mathcal{H}, \mathcal{I})$ of $\mathcal{H}$ *for a an instance* $\mathcal{I}$ of $\mathcal{P}$ is the number of solutions of $\mathcal{I}$ that are better than the solution produced by $\mathcal{H}$. The *blackball number* $\text{bbn}(\mathcal{H}, n)$ of $\mathcal{H}$ is the maximum of $\text{domn}(\mathcal{H}, \mathcal{I})$ over all instances $\mathcal{I}$ of size n.

When the number of solutions depends not only on the size of the instance of the CO problem at hand (for example, the number of independent sets of vertices in a graph G on n vertices depends on the structure of G), the domination ratio of an algorithm $\mathcal{A}$ is of interest: the *domination ratio* of $\mathcal{A}$, $\mathrm{domr}(\mathcal{A}, n)$, is the minimum of $\mathrm{domn}(\mathcal{A}, \mathcal{I})/\mathrm{sol}(\mathcal{I})$, where $\mathrm{sol}(\mathcal{I})$ is the number of solutions of $\mathcal{I}$, taken over all instances $\mathcal{I}$ of size n. Clearly, domination ratio belongs to the interval $(0, 1]$ and exact algorithms are of domination ratio 1.

Methods

Currently, there are two powerful methods in DA. One is used to prove that the heuristic under consideration is of domination number 1. For this method to be useful, the heuristic has to be a greedy-type algorithm for a CO problem on independence systems. We describe the method and its applications in the subsection Greedy-Type Algorithms. The other method is used prove that the heuristic under consideration is of very large domination number. For many problems this follows from the fact that the heuristic always finds a solution that is not worse than the average solution. This method is described in the subsection Better-Than-Average Heuristics.

Greedy-Type Algorithms

The main practical message of this subsection is that one should be careful while using the classical greedy algorithm and its variations in combinatorial optimization (CO): there are many instances of CO problems for which such algorithms will produce the unique worst possible solution. Moreover, this is true for several well-known optimization problems and the corresponding instances are not exotic, in a sense. This means that not always the paradigm of greedy optimization provides any meaningful optimization at all.

An *independence system* is a pair consisting of a finite set E and a family $\mathcal{F}$ of subsets (called *independent sets*) of E such that (I1) and (I2) are satisfied.

(I1) the empty set is in $\mathcal{F}$;
(I2) If $X \in \mathcal{F}$ and Y is a subset of X, then $Y \in \mathcal{F}$.

All maximal sets of $\mathcal{F}$ are called *bases*. An independence system is *uniform* if all its bases are of the same cardinality.

Many combinatorial optimization problems can be formulated as follows. We are given an independence system $(E,\mathcal{F})$, a set $W \subseteq \mathbb{Z}_+$ and a weight function w that assigns a weight $w(e) \in W$ to every element of E ($\mathbb{Z}_+$ is the set of non-negative integers). The weight $w(S)$ of $S \in \mathcal{F}$ is defined as the sum of the weights of the elements of S. It is required to find a base $B \in \mathcal{F}$ of minimum weight. We will consider only such problems and call them the *(E,F,W)-optimization problems*.

If $S \in \mathcal{F}$, then let $I(S) = \{x : S \cup \{x\} \in \mathcal{F}\} - S$. This means that $I(S)$ consists of those elements from E-S, which can be added to S, in order to have an independent set of size $|S| + 1$. Note that by (I2) $I(S) \neq \emptyset$ for every independent set S which is not a base.

The greedy algorithm tries to construct a minimum weight base as follows: it starts from an empty set X, and at every step it takes the current set X and adds to it a minimum weight element $e \in I(X)$, the algorithm stops when a base is built. We assume that the greedy algorithm may choose any element among equally weighted elements in $I(X)$. Thus, when we say that the greedy algorithm *may construct* a base B, we mean that B is built provided the appropriate choices between elements of the same weight are made.

An *ordered partitioning* of an ordered set $Z = \{z_1, z_2, \dots, z_k\}$ is a collection of subsets $A_1, A_2, \dots, A_q$ of Z such that if $z_r \in A_i$ and $z_s \in A_j$ where $1 \leq i < j \leq q$ then $r < s$. Some of the sets A_i may be empty and $\cup_{i=1}^{q} A_i = Z$.

The following theorem by Bang-Jensen, Gutin and Yeo [6] characterizes all uniform independence systems $(E,\mathcal{F})$ for which there is an assignment of weights to the elements of E such that the greedy algorithm solving the $(E, \mathcal{F}, \{1, 2, \dots, r\})$-optimization problem may construct the unique worst possible solution.

Theorem 1 *Let $(E,\mathcal{F})$ be a uniform independence system and let $r \geq 2$ be a natural number. There exists a weight assignment $w : E \to \{1, 2, \dots, r\}$ such that the greedy algorithm may produce the unique worst possible base if and only if $\mathcal{F}$ contains some base B with the property that for some ordering $x_1, \dots, x_k$ of the elements of B and some ordered partitioning $A_1, A_2, \dots, A_r$ of $x_1, \dots, x_k$ the following holds for every base $B' \neq B$ of $\mathcal{F}$:*

$$\sum_{j=0}^{r-1} |I(A_{0,j}) \cap B'| < \sum_{j=1}^{r} j \cdot |A_j| \,, \qquad (1)$$

where $A_{0,j} = A_0 \cup \dots \cup A_j$ and $A_0 = \emptyset$.

The special case $r = 2$ has an 'easier' characterization also proved in [6].

Theorem 2 *Let $(E,\mathcal{F})$ be a uniform independence system. For every choice of distinct natural numbers a,b there exists a weight function $w\colon E \to \{a, b\}$ such that the greedy algorithm may produce the unique worst base if and only if $\mathcal{F}$ contains a base $B = \{x_1, x_2, \dots, x_k\}$ such that for some $1 \le i < k$ the following holds:*

(a) *If B' is a base such that $\{x_1, \dots, x_i\} \subseteq B'$ then $B' = B$.*
(b) *If B' is a base such that $\{x_{i+1}, \dots, x_k\} \subseteq B'$ then $B' = B$.*

Using Theorem 1, the authors of [6] proved the following two corollaries.

Corollary 3 *Consider STSP as an $(E,\mathcal{H},W)$-optimization problem. Let $n \ge 3$.*

(a) *If $n \ge 4$ and $|W| \le \lfloor \frac{n-1}{2} \rfloor$, then the greedy algorithm never produces the unique worst possible base (i. e., tour).*
(b) *If $n \ge 3$, $r \ge n - 1$ and $W = \{1, 2, \dots, r\}$, then there exists a weight function $w\colon E \to \{1, 2, \dots, r\}$ such that the greedy algorithm may produce the unique worst possible base (i. e., tour).*

Corollary 4 *Consider ATSP as an $(E,\mathcal{H},W)$-optimization problems. Let $n \ge 3$.*

(a) *If $|W| \le \lfloor \frac{n-1}{2} \rfloor$, then the greedy algorithm never produces the unique worst possible base (i. e., tour).*
(b) *For every $r \ge \lceil \frac{n+1}{2} \rceil$ there exists a weight function $w\colon E(K_n^*) \to \{1, 2, \dots, r\}$ such that the greedy algorithm may produce the unique worst possible base (i. e., tour).*

Let $\mathcal{F}$ be the sets of those subsets X of $E(K_{2n})$ which induce a bipartite graph with at most n vertices in each partite set. Then $(E(K_{2n}), \mathcal{F})$ is a uniform independence system and the bases of $(E(K_{2n}), \mathcal{F})$ correspond to copies of the complete balanced bipartite graph $K_{n,n}$ in K_{2n}. The $(E(K_{2n}), \mathcal{F}, \mathbb{Z}_+)$-optimization problem is called the *Minimum Bisection Problem*. Theorem 2 implies the following:

Corollary 5 [6] *Let $n \ge 4$. The greedy algorithm for the $(E(K_{2n}), \mathcal{F}, W)$-optimization problem may produce the unique worst solution even if $|W| = 2$.*

For $W = \mathbb{Z}_+$, the following sufficient condition can often be used:

Theorem 6 [21] *Let $(E,\mathcal{F})$ be an independence system which has a base $B' = \{x_1, x_2, \dots, x_k\}$ such that the following holds for every base $B \in \mathcal{F}$, $B \ne B'$,*

$$\sum_{j=0}^{k-1} |I(x_1, x_2, \dots, x_j) \cap B| < k(k + 1)/2 .$$

Then the greedy algorithm for the $(E, \mathcal{F}, \mathbb{Z}_+)$-optimization problem may produce the unique worst solution.

Gutin, Yeo and Zverovich [23] considered the well-known *nearest neighbor (NN)* TSP heuristic: the tour starts at any vertex x of the complete directed or undirected graph; we repeat the following loop until all vertices have been included in the tour: add to the tour a vertex (among vertices not yet in the tour) closest to the vertex last added to the tour. It was proved in [23] that the domination number of NN is 1 for any $n \ge 3$.

Bendall and Margot [8] studied greedy-type algorithms for many CO problems. Greedy-type algorithms were introduced in [18]. They include NN and were defined as follows. A *greedy-type* algorithm $\mathcal{H}$ is similar to the greedy algorithm: start with the partial solution $X = \emptyset$; and then repeatedly add to X an element of minimum weight in $I_{\mathcal{H}}(X)$ (ties are broken arbitrarily) until X is a base of $\mathcal{F}$, where $I_{\mathcal{H}}(X)$ is a subset of $I(X)$ that does not depend on the cost function c, but only on the independence system $(E,\mathcal{F})$ and the set X. Moreover, $I_{\mathcal{H}}(X)$ is non-empty if $I(X) \ne \emptyset$, a condition that guarantees that $\mathcal{H}$ always outputs a base. Bendall and Margot [8] obtained complicated sufficient conditions for an independent system $(E,\mathcal{F})$ that ensure that every greedy-type algorithm is of domination number 1 for the $(E, \mathcal{F}, \mathbb{Z}_+)$-optimization problem.

The conditions imply that every greedy-type algorithm is of domination number 1 for the following classical CO problems [8]: (1) The Minimum Bisection Problem; (2) The *k-Clique Problem*: find a set of k vertices in a complete graph so that the sum of the weights of the edges between them is minimum; (3) ATSP; (4) STSP; (5) The *MinMax Matching Subgraph Problem*:

find a maximal (with respect to inclusion) matching so that the sum of the weights of the edges in the matching is minimum; (6) The *Assignment Problem*: find a perfect matching in a weighted complete bipartite graph so that the sum of the weights of the edges in the matching is minimum.

Better-Than-Average Heuristics

The idea of this method is to show that a heuristic is of very large domination number if it always produces a solution that is not worse than the average solution. The first such result was proved by Rublineckii [31] for the STSP.

Theorem 7 *Every STSP heuristic that always produces a tour not worse than the average tour (of the instance) is of domination number at least* $(n-2)!$ *when n is odd and* $(n-2)!/2$ *when n is even.*

The following similar theorem was proved by Sarvanov [32] for *n* odd and Gutin and Yeo [20] for *n* even.

Theorem 8 *Every ATSP heuristic that always produces a tour not worse than the average tour (of the instance) is of domination number at least* $(n-2)!$ *for each* $n \neq 6$.

The two theorems has been used to prove that a wide variety of TSP heuristics have domination number at least $\Omega((n-2)!)$. We discuss two families of such heuristics.

Consider an instance of the ATSP (STSP). Order the vertices $x_1, x_2, \dots, x_n$ of K_n^* (K_n) using some rule. The *generic vertex insertion algorithm* proceeds as follows. Start with the cycle $C_2 = x_1x_2x_1$. Construct the cycle C_j from C_{j-1} ($j = 3, 4, 5, \dots, n$), by inserting the vertex x_j into $C_{j\text{-}1}$ at the optimum place. This means that for each arc $e = xy$ which lies on the cycle $C_{j\text{-}1}$ we compute $w(xx_j) + w(x_jy) - w(xy)$, and insert x_j into the arc $e = xy$, which obtains the minimum such value. Here *w(uv)* denotes the weight of an arc *uv*. E.M. Lifshitz (see [31]) was the first to prove that the generic vertex insertion algorithm always produces a tour not worse than the average tour. Thus, we have the following:

Corollary 9 *The generic vertex insertion algorithm has domination number at least* $(n-2)!$ $(n \neq 6)$.

In TSP local search (LS) heuristics, a neighborhood *N(T)* is assigned to every tour *T*; *N(T)* is a set of tours in some sense close to *T*. The *best improvement* LS proceeds as follows. We start from a tour T_0. In the *i*'th iteration ($i \geq 1$), we search in the neighborhood $N(T_{i-1})$ for the best tour T_i. If the weights of $T_{i\text{-}1}$ and T_i do not coincide, we carry out the next iteration. Otherwise, we output T_i.

The *k-Opt*, $k \geq 2$, neighborhood of a tour *T* consists of all tours that can be obtained by replacing a collection of *k* edges (arcs) by a collection of *k* edges (arcs). It is easy to see that one iteration of the best improvement *k*-Opt LS can be completed in time $O(n^k)$. Rublineckii [31] showed that every local optimum for the best improvement 2-Opt or 3-Opt LS for STSP is of weight at least the average weight of a tour and, thus, by Theorem 7 is of domination number at least $(n-2)!/2$ when *n* is even and $(n-2)!$ when *n* is odd. Observe that this result is of restricted interest since to reach a *k*-Opt local optimum one may need exponential time (cf. Section 3 in [26]). However, Punnen, Margot and Kabadi [29] managed to prove that, in polynomial time, the best improvement 2-Opt and 3-Opt LS's for STSP produce a tour of weight at least the average weight of a tour. Thus, we have the following:

Corollary 10 *For the STSP the best improvement 2-Opt LS produces a tour, which is not worse than at least* $\Omega((n-2)!)$ *other tours, in at most* $O(n^3 \log n)$ *iterations.*

Corollary 10 is also valid for the best improvement 3-Opt LS and some other LS heuristics for TSP, see [24,29]. In the next section, we will give further examples of better-than-average heuristics for problems other than TSP.

Cases

Traveling Salesman Problem

In the previous sections, we discussed several TSP heuristics. However, there are many more TSP heuristics and, in this subsection, we consider some of them. In the next subsection, some general upper bounds are given on the domination number of TSP heuristics.

Gutin, Yeo and Zverovich [23] considered the *repeated nearest neighbor (RNN)* heuristic, which is the following variation of the NN heuristic: construct *n* tours by starting NN from each vertex of the complete (di)graph and choose the best tour among the *n* tours. The authors of [23] proved that for the ATSP, RNN always produces a tour, which is not worse than at least $n/2-1$ other tours, but for some instance it finds a tour, which is not worse than at most *n-2* other tours, $n \geq 4$. We also show that, for some instance of the STSP on $n \geq 4$ vertices, RNN produces a tour not worse than at most 2^{n-3} tours.

Another ATSP heuristic, max-regret-fc (fc abbreviates First Coordinate), was first introduced by Ghosh et al. [13]. Extensive computational experiments in [13] demonstrated a clear superiority of max-regret-fc over the greedy algorithm and several other construction heuristics from [14]. Therefore, the result of Theorem 11 obtained by Gutin, Goldengorin and Huang [15] was somewhat unexpected.

Let K_n^* be a complete digraph with vertices $V = \{1, 2, \ldots, n\}$. The weight of an arc *(i,j)* is denoted by w_{ij}. Let Q be a collection of disjoint paths in K_n^*. An arc *a=(i,j)* is a *feasible addition* to Q if Q+*a* is either a collection of disjoint paths or a tour in K_n^*. Consider the following two ATSP heuristics: max-regret-fc and max-regret.

The heuristic *max-regret-fc* proceeds as follows. Set $W = T = \emptyset$. While $V \neq W$ do the following: For each $i \in V \setminus W$, compute two lightest arcs *(i,j)* and *(i,k)* that are feasible additions to T, and compute the difference $\Delta_i = |w_{ij} - w_{ik}|$. For $i \in V \setminus W$ with maximum Δ_i choose the lightest arc *(i,j)*, which is a feasible addition to T and add *(i,j)* to M and i to W.

The heuristic *max-regret* proceeds as follows. Set $W^+ = W^- = T = \emptyset$. While $V \neq W^+$ do the following: For each $i \in V \setminus W^+$, compute two lightest arcs *(i,j)* and *(i,k)* that are feasible additions to T, and compute the difference $\Delta_i^+ = |w_{ij} - w_{ik}|$; for each $i \in V \setminus W^-$, compute two lightest arcs *(j,i)* and *(k,i)* that are feasible additions to T, and compute the difference $\Delta_i^- = |w_{ji} - w_{ki}|$. Compute $i' \in V \setminus W^+$ with maximum $\Delta_{i'}^+$ and $i'' \in V \setminus W^-$ with maximum $\Delta_{i''}^-$. If $\Delta_{i'}^+ \geq \Delta_{i''}^-$ choose the lightest arc (i', j'), which is a feasible addition to T and add (i', j') to M, i' to W^+ and j' to W^-. Otherwise, choose the lightest arc (j'', i''), which is a feasible addition to T and add (j'', i'') to M, i'' to W^- and j'' to W^+.

Notice that in max-regret-fc, if $|V \setminus W| = 1$ we set $\Delta_i = 0$. A similar remark applies to max-regret.

Theorem 11 *The domination number of both max-regret-fc and max-regret equals 1 for each* $n \geq 2$.

Upper Bounds for Domination Numbers of ATSP Heuristics

It is realistic to assume that any ATSP algorithm spends at least one unit of time on every arc of K_n^* that it considers. We use this assumption in this subsection.

Theorem 12 [17] *Let $\mathcal{A}$ be an ATSP heuristic of running time t(n). Then the domination number of $\mathcal{A}$ does not exceed* $\max_{1 \leq n' \leq n}(t(n)/n')^{n'}$.

Corollary 13 [17] *Let $\mathcal{A}$ be an ATSP heuristic of complexity t(n). Then the domination number of $\mathcal{A}$ does not exceed* $\max\{e^{t(n)/e}, (t(n)/n)^n\}$, *where e is the basis of natural logarithms.*

The next assertion follows directly from the proof of Corollary 13.

Corollary 14 [17] *Let $\mathcal{A}$ be an ATSP heuristic of complexity t(n). For* $t(n) \geq en$, *the domination number of $\mathcal{A}$ does not exceed* $(t(n)/n)^n$.

We finish this subsection with a result from [17] that improves (and somewhat clarifies) Theorem 20 in [29].

Theorem 15 *Unless P = NP, there is no polynomial time ATSP algorithm of domination number at least* $(n-1)! - \lfloor n - n^\alpha \rfloor!$ *for any constant* $\alpha < 1$.

Multidimensional Assignment Problem (MAP)

In case of s dimensions, MAP is abbreviated by s-AP and defined as follows. Let $X_1 = X_2 = \cdots = X_s = \{1, 2, \ldots, n\}$. We will consider only vectors that belong to the Cartesian product $X = X_1 \times X_2 \times \cdots \times X_s$. Each vector e is assigned a weight *w(e)*. For a vector e, e_j denotes its jth coordinate, i. e., $e_j \in X_j$. A *partial assignment* is a collection $e^1, e^2, \ldots, e^t$ of $t \leq n$ vectors such that $e_j^i \neq e_j^k$ for each $i \neq k$ and $j \in \{1, 2, \ldots, s\}$. An *assignment* is a partial assignment with n vectors. The *weight* of a partial assignment $A = \{e^1, e^2, \ldots, e^t\}$ is $w(A) = \sum_{i=1}^t w(e^i)$. The objective is to find an assignment of minimum weight. Notice that s-AP has $(n!)^{s-1}$ solutions (assignments).

s-AP can be considered as the $(X, \mathcal{F}, \mathbb{Z}_+)$-optimization problem. ($\mathcal{F}$ consists of partial assignments including the empty one.) This allows us to define the greedy algorithm for s-AP and to conclude from Theorem 6 that the greedy algorithm is of domination number 1 (for every fixed $s \geq 3$).

In the subsection Traveling Salesman Problem, we considered the *max-regret-fc and max-regret heuristics*. In fact, max-regret was first introduced for 3-AP by Balas and Saltzman [4]. (See [15] for detailed description of the s-AP max-regret-fc and max-regret heuristics for each $s \geq 2$.) In computational experiments, Balas and Saltzman [4] compared the greedy algorithm with max-regret and concluded that max-regret is superior to the greedy algorithm with respect to the quality of solutions. However, after conducting wider computational experiments, Robertson [30] came to a different conclusion: the greedy algorithm and max-regret are of similar quality for 3-AP. Gutin, Goldengorin and Huang [15] share the conclusion of Robertson: both max-regret and max-regret-fc are of domination number 1 (similarly to the greedy algorithm) for s-AP for each $s \geq 3$. Moreover, there exists a family of s-AP instances for which all three heuristics will find the unique worst assignment [15] (for each $s \geq 3$).

Similarly to TSP, we may obtain MAP heuristics of factorial domination number if we consider better-than-average heuristics. This follows from the next theorem:

Theorem 16 [15] *Let $\mathcal{H}$ be a heuristic that for each instance of s-AP constructs an assignment of weight at most the average weight of an assignment. Then the domination number of $\mathcal{H}$ is at least* $((n-1)!)^{s-1}$.

Balas and Saltzman [4] introduced a *3-Opt heuristic* for 3-AP which is similar to the 3-Opt TSP heuristic. The *3-Opt neighborhood* of an assignment $A = \{e^1, e^2, \dots, e^n\}$ is the set of all assignments that can be obtained from A by replacing a triple of vectors with another triple of vectors. The 3-Opt is a local search heuristic that uses the 3-Opt neighborhood. It is proved in [15] that an assignment, that is the best in its 3-Opt neighborhood, is at least as good as the average assignment. This implies that 3-Opt is of domination number at least $((n-1)!)^2$. We cannot guarantee that 3-Opt local search will stop after polynomial number of iterations. Moreover, 3-Opt is only for 3-AP. Thus, the following heuristic introduced and studied in [15] is of interest.

Recursive Opt Matching (ROM) proceeds as follows. Compute a new weight $\bar{w}(i,j) = w(X_{ij})/n^{s-2}$, where X_{ij} is the set of all vectors with last two coordinates equal i and j, respectively. Solving the 2-AP with the new weights to optimality, find an optimal assignment $\{(i, \pi_s(i)) : i = 1, 2, \dots, n\}$, where π_s is a permutation on X_s. While $s \neq 1$, introduce *(s-1)*-AP with weights given as follows: $w'(f^i) = w(f^i, \pi_s(i))$ for each vector $f^i \in X'$, where $X'n = X_1 \times X_2 \times \dots \times X_{s-1}$, with last coordinate equal i and apply ROM recursively. As a result we have obtained permutations $\pi_s, \pi_{s-1}, \dots, \pi_2$. The output is the assignment $\{(i, \pi_2(i), \dots, \pi_s(\pi_{s-1}(\dots(\pi_2(i)))\dots)) : i = 1, 2, \dots, n\}$.

Clearly, ROM is of running time $O(n^3)$ for every fixed $s \geq 3$. Using Theorem 16, it is proved in [15] that ROM is of domination number at least $((n-1)!)^{s-1}$.

Minimum Partition and Multiprocessor Scheduling Problems

In this subsection, N always denotes the set $\{1, 2, \dots, n\}$ and each $i \in N$ is assigned a positive integral weight $\sigma(i)$. $\mathcal{A} = (A_1, A_2, \dots, A_p)$ is a *p-partition* of N if each $A_i \subseteq N$, $A_i \cap A_j = \emptyset$ for each $i \neq j$ and the union of all sets in $\mathcal{A}$ equals N. For a subset A of N, $\sigma(A) = \sum_{i \in A} \sigma(i)$. The *Minimum Multiprocessor Scheduling Problem* (MMS) [3] can be stated as follows. We are given a triple (N, σ, p), where p is an integer, $p \geq 2$. We are required to find a p-partition C of N that minimizes $\sigma(\mathcal{A}) = \max_{1 \leq i \leq p} \sigma(A_i)$ over all p-partitions $\mathcal{A} = (A_1, A_2, \dots, A_p)$ of N.

Clearly, if $p \geq n$, then MMS becomes trivial. Thus, in what follows, $p < n$. The size s of MMS is $\Theta(n + \sum_{i=1}^{n} \log \sigma(i))$. Consider the following heuristic $\mathcal{H}$ for MMS. If $s \geq p^n$, then we simply solve the problem optimally. This takes $O(s^2)$ time, as there are at most $O(s)$ solutions, and each one can be evaluated and compared to the current best in $O(s)$ time. If $s < p^n$, then we sort the elements of the sequence $\sigma(1), \sigma(2), \dots, \sigma(n)$. For simplicity of notation, assume that $\sigma(1) \geq \sigma(2) \geq \dots \geq \sigma(n)$. Compute $r = \lceil \log n / \log p \rceil$ and solve MMS for $(\{1, 2, \dots, r\}, \sigma, p)$ to optimality. Suppose we have obtained a p-partition $\mathcal{A}$ of $\{1, 2, \dots, r\}$. Now for i from *r+1* to n add i to the set A_j of the current p-partition $\mathcal{A}$ with smallest $\sigma(A_j)$. The following result was proved by Gutin, Jensen and Yeo [16].

Theorem 17 *The heuristic $\mathcal{H}$ runs in time $O(s^2 \log s)$ and* $\lim_{s \to \infty} \operatorname{domr}(\mathcal{H}, s) = 1$.

The *Minimum Partition Problem (MP)* is MMS with *p=2*. Alon, Gutin and Krivelevich [2] proved Theorem 17 for MP with s replaced by n.

Max Cut Problem

The *Max Cut (MC)* is the following problem: given a weighted graph *G=(V,E)*, find a bipartition (a *cut*) *(X,Y)* of *V* such that the sum of weights of the edges with one end vertex in *X* and the other in *Y*, called the *weight of the cut (X,Y)*, is maximum. For this problem, there are some better-than-average heuristics. The simplest is probably the following greedy-like heuristic *C*; order the vertices arbitrarily and put each vertex in its turn either in *X* or in *Y* in order to maximize in each step the total weight of crossing edges.

Using an advanced probabilistic approach Alon, Gutin and Krivelevich [2] proved that the heuristic *C* is of domination ratio larger than 0.025. For the unweighted MC (all weights are equal), a better quality algorithm can be designed as described in [2]. Its domination ratio is at least $1/3 - o(1)$.

Constraint Satisfaction Problems

Let r be a fixed positive integer, and let $\mathcal{F} = \{f_1, f_2, \ldots, f_m\}$ be a collection of Boolean functions, each involving at most r of the n variables, and each having a positive weight $w(f_i)$. The *Max-r-Constraint Satisfaction Problem* (or *Max-r-CSP*, for short), is the problem of finding a truth assignment to the variables so as to maximize the total weight of the functions satisfied. Note that this includes, as a special case, the Max Cut problem. Another interesting special case is the Max-*r*-SAT problem, in which each of the Boolean functions f_i is a clause of at most r literals.

Alon, Gutin and Krivelevich [2] proved the following:

Theorem 18 *For each fixed integer $r \geq 1$ there exists a linear time algorithm for the Max-r-CSP problem, whose domination ratio exceeds $\frac{1}{2^{4/3} \cdot 2^{6r}}$.*

Vertex Cover, Independent Set and Clique Problems

A *clique* in a graph *G* is a set of vertices in *G* such that every pair of vertices in the set are connected by an edge. The *Maximum Clique Problem* (*MCl*) is the problem of finding a clique of maximum cardinality in a graph. A *vertex cover* in a graph *G* is a set *S* of vertices in *G* such that every edge is incident to a vertex in *S*. The *Minimum Vertex Cover Problem* (*MVC*) is the problem of finding a minimum cardinality vertex cover. An *independent set* in a graph is a set *S* of vertices such that no edge joins two vertices in *S*. The *Maximum Independent Set Problem* (*MIS*) is the problem of finding a minimum cardinality independent set in a graph. It is easy to see that the number of cliques and independent sets in a graph depends on its structure, and not only on the number of vertices. The same holds for vertex covers.

Notice that if *C* is a vertex cover of a graph *G*, then $V(G) \setminus C$ is an independent set in *G*; if *Q* is a clique in *G*, then *Q* is an independent set in the complement of *G*. These well-known facts imply that if there is a heuristic for one of the problem of domination ratio at least *r(n)*, all other problems admit a heuristic of domination ratio at least *r(n)*.

MCl, MIS and MVC are somewhat different from the previous problems we have considered. Firstly, the number of feasible solutions, for an input of size *n*, depends on the actual input, and not just its size. The second difference is that the three problems do not admit polynomial-time heuristics of domination ratio at least $1/p(n)$ for any polynomial *p(n)* in *n* unless P=NP. This was proved by Gutin, Vainshtein and Yeo [19].

Because of the first difference, it is better to compare heuristics for the problems using the blackball number rather than domination number. Since a heuristic for MVC can be easy transformed into a heuristic for the other two problems, we restrict ourselves only to MVC heuristics.

The *incremental deletion heuristic* starts with an arbitrary permutation π of vertices of *G* and an initial solution *S=V(G)*. We consider each vertex of *G* in turn (according to π), deleting it from *S* if the resulting subset remains a (feasible) solution. A seemingly better heuristic for MVC is obtained by ordering the vertices by degree (lower degrees first), and then applying the incremental deletion heuristic. We call it the *increasing-degree deletion heuristic*. The well-known *maximal matching heuristic* constructs a maximal matching *M* and outputs both end-vertices of all edges in *M* as a solution. Berend, Skiena and Twitto [10] proved that the incremental deletion heuristic (increasing-degree deletion heuristic, maximal matching heuristic) is of blackball number $2^{n-1} - n$ (of blackball number larger than $2 - \epsilon)^n$ for each $\epsilon > 0$, of blackball number approximately 1.839^n). Clearly, the maximal matching heuristic is the best among the three heuristics from the DA point of view.

Quadratic Assignment Problem

The *Quadratic Assignment Problem* (*QAP*) can be formulated as follows. We are given two $n \times n$ matrices $A = [a_{ij}]$ and $B = [b_{ij}]$ of integers. Our aim is to find a permutation π of $\{1, 2, \ldots, n\}$ that minimizes the sum

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{\pi(i)\pi(j)} \, .$$

Gutin and Yeo [23] described a better-than-average heuristic for QAP and proved that the heuristic is of domination number at least $n!/\beta^n$ for each $\beta > 1$. Moreover, the domination number of the heuristic is at least $(n-2)!$ for every prime power n. These results were obtained using a group-theoretical approach.

See also

► Traveling Salesman Problem

References

1. Ahuja RK, Ergun Ö, Orlin JB, Punnen AP (2002) A survey of very large-scale neighborhood search techniques. Discret Appl Math 123:75–102
2. Alon N, Gutin G, Krivelevich M (2004) Algorithms with large domination ratio. J Algorithms 50:118–131
3. Ausiello G, Crescenzi P, Gambosi G, Kann V, Marchetti-Spaccamela A, Protasi M (1999) Complexity and Approximation. Springer, Berlin
4. Balas E, Saltzman MJ (1991) An algorithm for the three-index assignment problem. Oper Res 39:150–161
5. Bang-Jensen J, Gutin G (2000) Digraphs: Theory, Algorithms and Applications. Springer, London
6. Bang-Jensen J, Gutin G, Yeo A (2004) When the greedy algorithm fails. Discret Optim 1:121–127
7. Ben-Arieh D, Gutin G, Penn M, Yeo A, Zverovitch A (2003) Transformations of Generalized ATSP into ATSP: experimental and theoretical study. Oper Res Lett 31:357–365
8. Bendall G, Margot F (2006) Greedy Type Resistance of Combinatorial Problems. Discret Optim 3:288–298
9. Berge C (1958) The Theory of Graphs. Methuen, London
10. Berend B, Skiena SS, Twitto Y, Combinatorial dominance guarantees for heuristic algorithms. ACM Trans Algorithm (to appear)
11. Cornuejols G, Fisher ML, Nemhauser GL (1977) Location of bank accounts to optimize float; an analytic study of exact and approximate algorithms. Manag Sci 23:789–810
12. Deineko VG, Woeginger GJ (2000) A study of exponential neighbourhoods for the traveling salesman problem and the quadratic assignment problem. Math Prog Ser A 87:519–542
13. Ghosh D, Goldengorin B, Gutin G, Jäger G (2007) Tolerance-based greedy algorithms for the traveling salesman problem. Commun DQM 41:521–538
14. Glover F, Gutin G, Yeo A, Zverovich A (2001) Construction heuristics for the asymmetric TSP. Eur J Oper Res 129: 555–568
15. Gutin G, Goldengorin B, Huang J (2006) Worst Case Analysis of Max-Regret, Greedy and Other Heuristics for Multidimensional Assignment and Traveling Salesman Problems. Lect Notes Comput Sci 4368:214–225
16. Gutin G, Jensen T, Yeo A (2006) Domination analysis for minimum multiprocessor scheduling. Discret Appl Math 154:2613–2619
17. Gutin G, Koller A, Yeo A (2006) Note on Upper Bounds for TSP Domination Number. Algorithm Oper Res 1:52–54
18. Gutin G, Vainshtein A, Yeo A (2002) When greedy-type algorithms fail. Unpublished manuscript
19. Gutin G, Vainshtein A, Yeo A (2003) Domination Analysis of Combinatorial Optimization Problems. Discret Appl Math 129:513–520
20. Gutin G, Yeo A (2002) Polynomial approximation algorithms for the TSP and the QAP with a factorial domination number. Discret Appl Math 119:107–116
21. Gutin G, Yeo A (2002) Anti-matroids. Oper Res Lett 30: 97–99
22. Gutin G, Yeo A (2005) Domination Analysis of Combinatorial Optimization Algorithms and Problems. In: Golumbic M, Hartman I (eds) Graph Theory, Combinatorics and Algorithms: Interdisciplinary Applications. Springer, New York, pp 152–176
23. Gutin G, Yeo A, Zverovitch A (2002) Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP. Discret Appl Math 117:81–86
24. Gutin G, Yeo A, Zverovitch A (2002) Exponential Neighborhoods and Domination Analysis for the TSP. In: Gutin G, Punnen AP (eds) The Traveling Salesman Problem and its Variations. Kluwer, Dordrecht, pp 223–256
25. Johnson DS, Gutin G, McGeoch LA, Yeo A, Zhang X, Zverovitch A (2002) Experimental Analysis of Heuristics for ATSP. In: Gutin G, Punnen AP (eds) The Traveling Salesman Problem and its Variations. Kluwer, Dordrecht, pp 445–487
26. Johnson DS, McGeoch LA (1997) The traveling salesman problem: A case study in local optimization. In: Aarts EHL, Lenstra JK (eds) Local Search in Combinatorial Optimization. Wiley, Chichester, pp 251–310
27. Johnson DS, McGeoch LA (2002) Experimental Analysis of Heuristics for STSP. In: Gutin G, Punnen AP (eds) The Traveling Salesman Problem and its Variations. Kluwer, Dordrecht, pp 369–443
28. Kise H, Ibaraki T, Mine H (1979) Performance analysis of six approximation algorithms for the one-machine maximum

lateness scheduling problem with ready times. J Oper Res Soc Japan 22:205–223

29. Punnen AP, Margot F, Kabadi SN (2003) TSP heuristics: domination analysis and complexity. Algorithmica 35:111–127
30. Robertson AJ (2001) A set of greedy randomized adaptive local search procedure implementations for the multidimentional assignment problem. Comput Optim Appl 19:145–164
31. Rublineckii VI (1973) Estimates of the accuracy of procedures in the Traveling Salesman Problem. Numer Math Comput Tech 4:18–23 (in Russian)
32. Sarvanov VI (1976) The mean value of the functional of the assignment problem. Vestsi Akad Navuk BSSR, Ser Fiz-Mat Navuk 2:111–114 (in Russian)
33. Zemel E (1981) Measuring the quality of approximate solutions to zero-one programming problems. Math Oper Res 6:319–332

Duality Gaps in Nonconvex Optimization

PANOS PARPAS, BERÇ RUSTEM
Department of Computing, Imperial College, London, UK

MSC2000: 90B50, 78M50

Article Outline

Abstract

Duality gaps in optimization problems arise because of the nonconvexities involved in the objective function or constraints. The Lagrangian dual of a nonconvex optimization problem can also be viewed as a two-person zero-sum game. From this viewpoint, the occurrence of duality gaps originates from the order in which the two players select their strategies. Therefore, duality theory can be analyzed as a zero-sum game where the order of play generates an asymmetry. One can conjecture that this asymmetry can be eliminated by allowing one of the players to select strategies from a larger space than that of the finite-dimensional Euclidean space. Once the asymmetry is removed, then there is zero duality gap. The aim of this article is to review two methods by which this process can be carried out. The first is based on randomization of the primal problem. The second extends the space from which the dual variables can be selected. Duality gaps are important in mathematical programming and some of the results reviewed here are more than 50 years old, but only recently methods have been discovered to take advantage of them. The theory is elegant and helps appreciate the game-theoretic origins of the dual problem and the role of Lagrange multipliers.

Background

We discuss how duality gaps arise, and how they can be eliminated in nonconvex optimization problems. A standard optimization problem is stated as follows:

$$\begin{aligned} \min \quad & f(x)\,, \\ & g(x) \leq 0\,, \\ & x \in X\,, \end{aligned} \tag{1}$$

where $f\colon \mathbb{R}^n \to \mathbb{R}$ and $g\colon \mathbb{R}^n \to \mathbb{R}^m$ are assumed to be smooth and nonconvex. The feasible region of (1) is denoted by $\mathcal{F}$, and it is assumed to be nonempty and compact. X is some compact convex set.

In order to understand the origins of duality in mathematical programming, consider devising a strategy to determine whether a point, say y, is the globally optimal solution of (1). Such a strategy can be concocted as follows: if $f(y)$ is the global solution of (1) then the following system of inequalities

$$\begin{aligned} & f(x) < f(y)\,, \\ & g(x) \leq 0\,, \\ & x \in X \end{aligned} \tag{2}$$

will not have a solution. We can reformulate (2) in a slightly more convenient framework. Indeed, suppose that there exist m positive scalars λ_i, $i = 1, \dots, m$, such that

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) < f(y) \tag{3}$$