Electron Commerce Res (2007) 7: 341–365
DOI 10.1007/s10660-007-9010-9

A quantitative foundation for defining and manipulating deals to facilitate automated e-commerce

Boaz Golany · Oded Shmueli

Published online: 24 October 2007

Abstract We propose a goal programming framework that aims at automating e-commerce transactions. This framework consists of three basic layers: deal definition—defining the deal's parameters and associated constraints (e.g., item, price, delivery dates); deal manipulation—a collection of procedures for shaping deals to attain desired goals (e.g., earliest delivery and minimum price) and an applications layer that employs these procedures within some negotiations settings (e.g., an auction-related application presents a "better offer" while bidding on a contract). Our proposed foundation is rich enough to support a wide array of applications ranging from 1-1 and 1-n negotiations (auctions) to deal valuation and deal splitting. Whereas the techniques are appropriate to a multitude of settings, we shall mainly present them in the context of business-to-business (B2B) commerce where we see the greatest short term benefits.

Keywords Goal programming · Deal manipulation · Automated e-commerce · Automated negotiations

O. Shmueli's work is partially supported by the Fund for the Promotion of Research at the Technion.

B. Golany (✉)
Industrial Engineering and Management Faculty, Technion—Israel Institute of Technology, Haifa 32000, Israel
e-mail: golany@ie.technion.ac.il

O. Shmueli
Computer Science Faculty, Technion—Israel Institute of Technology, Haifa 32000, Israel
e-mail: oshmu@cs.technion.ac.il

1 Introduction

1.1 Overview

Recently, we have witnessed a massive growth in electronic commerce (EC). Yet, to a large extent, EC is still confined to applications that tightly involve direct human participation. In addition, most applications are fairly simple—buying a book or other product on-line, price comparison, or participating in an auction. Most of these activities are still carried out by humans that use electronic means primarily for communication (with automated trading being an exception). This is obviously time-consuming and highly limiting, especially as trade is carried over multiple time zones and at all hours. So, there is a growing recognition that substantial activities (such as deal valuation, negotiations and splitting a deal among a number of parties) need to be automated.

To automate, partially or fully, the kind of EC activities we envision, one needs a formal and well understood concept of a deal. For example, if a corporate buyer employs an electronic purchasing agent to negotiate with a provider the purchase of a list of supplies, a certain foundation is necessary. First, an agreed-upon vocabulary for describing building supplies must be established. With the emergence of web services [13], UDDI [32, 33], WSDL [4] and semantic web ontlogy languages such as OWL [34] we may assume that such vocabularies will become available and utilized [18, 29]. Second, one needs to be able to define constraints on acceptable deals. Such constraints may involve financial, timing and geographic considerations. Again, with an agreed-upon terminology, this does not present major complications. Third, one must define for the purchasing agent what one would consider a "good deal". This part is more complex, as deals may have a number of dimensions (price, date, quality, etc.). So, we need a formalism to express preferences and tradeoffs among various parameters.

A large early body of work on electronic negotiations was reviewed by Teich et al. [30] which mentions proposals for a multiple-criteria methodology as the basic foundation for software-based negotiations tools (e.g., [12, 23]. The idea of constructing software agents that will be able to define and evaluate deals and later negotiate on them has started to emerge only in the last decade. Another systematic approach for e-negotiations was proposed and demonstrated in a series of papers by Kersten and his colleagues (e.g., [2, 3, 16, 17]).

INSPIRE [3] provides web-based negotiation support. In the pre-negotiation stage the problem is defined: a list of issues and discrete options per issues. Piecewise linear interpolation between discrete options is also possible. A utility function based on a hybrid conjoint analysis is constructed. Issues are rated for importance and within each issue the discrete options for the issue are also rated. Then, the overall rating is verified based on the user's evaluation of complete packages.

The utility contribution of each issue is assumed to be independent of the others, so no inter issue tradeoffs are expressible. Then, negotiations take place by exchanging offers, accompanied by optional text. Preferences may change during negotiations. The system neither form offers to specification nor does it provide advice on negotiation tactics. Once the parties agree, the system switches to a post negotiation phase in order to achieve Pareto optimality. This phase is similar to the negotiation phase.

In contrast, the foundations proposed in this paper allow for a wide array of applications (including 1-1 negotiation as in [3]), provide a more expressive preference specification (including tradeoffs), deals thoroughly with offer manipulation, and addresses the issue of the state of knowledge during negotiations. The area of automated negotiations has recently received much attention. Foundational work in this area includes [5, 6, 11, 19, 25]. Because of space limitations, we cannot develop in depth the multitude of issues associated with automated negotiations. Instead, we refer the reader to a survey of automated negotiation procedures in [1] and a review of the issues involved in designing multi-criteria deals in e-auctions [31].

In this paper we present a mechanism for the formal definition of deals. The mechanism, a technology developed by Dealigence Inc. [28], is based on goal programming (GoP [8–10, 20, 24, 26])—a branch of linear programming [14, 15]. It enables the definition of constraints, preferences and tradeoffs among parameters of complex deals. Although numeric in nature, it has the ability to express "soft constraints" as well as discrete choices. An important capability of this mechanism is that it allows the comparison of 'apples and oranges', assigning a unique value to a multi-dimensional deal.

Once the purchasing agent is equipped with a well-defined deal formalism, it can manipulate (or shape) deals. For example, given an offer by a provider, the purchasing agent may want to demand 'better terms'. However, as deals are complex and involve many attributes, the agent needs a method of forming such deals. We have constructed a wide array of deal manipulation operators. Interestingly, most operators can be 'compiled' into goal programs (GPs) as well. This makes GoP a uniform foundation for both specifying and manipulating deals.

We exhibit that GoP is a robust foundation for automated agents. For this, a collection of tools is not enough. The agent needs a plan of action, or a strategy, for carrying out its actions. For example, suppose a provider offers the purchasing agent a deal d. The agent would like to ask for a better deal. Further, suppose the agent can compute f and h, the worst and best deals, respectively. Suppose further, that the agent can construct a deal to deal-value specification anywhere between f and h. Should the agent ask for a high-valued deal (close to that of h) or a lower-valued one (closer to f), or perhaps simply accept d (provided it has more value than f)? This is clearly a question of *strategy* that falls in the domain of game theory [21, 22]. Strategy is out of the scope of this paper; nevertheless, we show how the operators of the deal manipulation layer can facilitate the realization of application layer mechanisms implementing 1-1 and 1-n negotiations as well as deal splitting [27, 28].

Figure 1 illustrates the basic concepts. A human defines deal parameters via a graphical user interface (GUI). While the GUI is important in enabling human-machine interaction, we will not address GUI-related issues in this paper. We concentrate on the "post-GUI" stage. The first post-GUI step is deal compilation, namely the process of formulating a deal (specified in human-oriented terminology) as a GP. We use GoP as a formalism for representing (deal definition layer) and manipulating (deal manipulation layer) deals. Finally, we use deal manipulation operators within game-theoretic inspired mechanisms (application layer). As we show, in addition to traditional EC applications, applications such as catalog choosing [7], self-service and deal-splitting [28] can also utilize the GoP formulation. Our approach should be

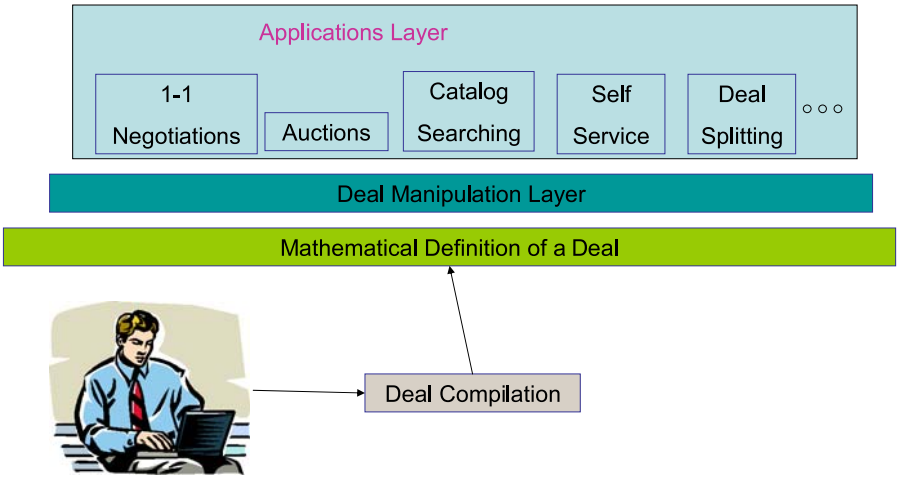

Fig. 1 Basic architecture

contrasted with ad-hoc solutions that do not rely on a 'universal' deal representation. Such solutions are not extensible and tend to 'mix' levels of abstractions (deal optimization and mechanisms, for example).

We foresee the greatest short term benefits for our techniques in the context of business-to-business (B2B) commerce. However, our techniques are not aimed at commodity markets (e.g., wheat, oil) with established exchanges but rather at ad-hoc deals with interdependent constraints.

The rest of the paper is structured as follows. In Sect. 1 we explain how deals can be represented as goal programs. In Sect. 2 we explain how to compile goal programs. The deal manipulation layer is described in Sect. 3. Selected applications are described in Sect. 4 and in Sect. 5 we present our conclusions.

1.2 Representing a deal as a goal program

The specification of a deal may be obtained from a user in a number of ways. Typically, the user is presented with a GUI that may be pre-populated with various pieces of information, ranging from identification numbers to recommended or constrained values for certain fields. For example, a user may be presented with a GUI as displayed in the table below and asked to fill in the missing fields.

This deal consists of buying two items—bricks and iron. The deal attributes are bricksDays (number of days till delivery), bricksNumber (number of bricks), ironQuantity (in kilograms) and overallPrice (in US$). The specification columns allow the buyer to express various constraints and preferences. Consider bricksDays. The buyer indicates a hard constraint—delivery cannot be done in less than 5 days (e.g., the storage area may not be ready) and cannot be done in more than 25 days (presumably, this would cause unbearable delay). That said, the buyer indicates a preference for delivery in 10–14 days and expresses indifference to values within this range. Next, the buyer explains how deviations from the 10–14 desired interval should be

Table 1 An example of a deal specification GUI

Item	Attribute	Specification						
Bricks	Days	Min	Max	Left	Right	Left penalty	Right penalty	Importance
		5	25	10	14	M	H	2
Bricks	Number	Min	Max	Left	Right	Left penalty	Right penalty	Importance
		1000	1300	1100	1100	H	L	2
Iron	Quantity	Min	Max	Left	Right	Left penalty	Right penalty	Importance
		5K	6K	6K	6K	L		1
Overall	Price	Min	Max	Left	Right	Left penalty	Right penalty	Importance
		17K	25K	20K	20K	M	H	1

treated. A "right" penalty of H (high) indicates that values greater than 14 are highly undesirable. A "left" penalty of L (low) indicates also that values smaller than 10 are not desirable but are only lightly so. Finally, given a complex deal, one needs to compare the importance of bricksDays related preferences to other preferences. This is done via the importance column in which values in the 1–5 range are specified (1 is very important and 5 is slightly important).

The ironQuantity specification introduces a situation in which the desired interval consists only of the rightmost allowed value of 6 tons. Note that there is no right penalty as deviations to the right are simply not allowed. This is an example of a one-sided goal (more on this later). Finally, the overallPrice indicates an interesting situation. A minimum price of US$17K is indicated. Apparently the buyer suspects that cheaper deals are not realizable or that they indicate an illegal aspect. This perception is reinforced by actually penalizing deals that are between US$17–20K.

Generalizing this simple example, a deal instance is characterized by the items (products or services), the attributes of these items (such as weight, color and quality), overall deal attributes (such as delivery date), hard constraints, soft constraints (delivery in 10–14 days with allowed deviations), desired targets (e.g., price US$20K), and tradeoffs (e.g., for each day of early delivery an additional US$100 in payment, not illustrated in this example). In addition, when describing soft constraints, a penalty for each deviation is specified. Finally, not all preferences are equally important and the degree of relative importance is indicated.

Regardless of the GUI, it is necessary for the deal specification to be translated, or compiled, into a mathematical object so as to facilitate comparisons and manipulations of deals. This can be done in many ways ranging from logic-based formalisms to differential equations. We have chosen GoP, a type of linear programming, for that purpose. The motivation is threefold. First, goal programs (GPs) are designed specifically to handle imprecise statements and for comparing 'apples and oranges'. Second, GPs lend themselves to practical efficient solutions by a wide array of linear solvers. Third, they are a natural formalism to express inter-parameter tradeoffs.

Table 2 The GP for the iron and brick example

LEX MIN
Level 1: $0.6\beta^- + 1.0\beta^+ + 0.10\delta^-$
Level 2: $0.6\pi_1^- + 1.0\pi_2^+ + 0.002\epsilon^- + 0.001\epsilon^+$
GOAL CONSTRAINTS:
bricksDays $+\pi_1^- - \pi_1^+ = 10$
bricksDays $-\pi_2^+ + \pi_2^- = 14$
bricksNumber $+\epsilon^- - \epsilon^+ = 1100$
ironQuantity $+\delta^- = 6000$
overallPrice $+\beta^- - \beta^+ = 20000$
HARD CONSTRAINTS:
$5 \leq$ brickDays ≤ 25
$1000 \leq$ brickNumber ≤ 1300
$5000 \leq$ ironQuantity ≤ 6000
$17000 \leq$ overallPrice ≤ 25000

In the next section we describe various aspects of deal compilation. Clearly, this is not a trivial process as it translates the rather informal GUI description into a very formal mathematical object. Suppose we have carried out this translation for our purchasing example. The resulting program would look as follows.

In specifying brickDays $+\pi_1^- - \pi_1^+ = 10$, the user indicates a desire for a value of 10 for the decision variable brickDays (since π_1^- and π_1^+ are negative and positive deviation variables that are minimized in the objective function). Generally in GPs, deviation variables are to be minimized. The user has specified that deviations associated with bricksDays and with bricksNumber are at the same level of importance. However, the associated ranges are quite different (5–25 for bricksDays and 1100–1300 for bricksNumber). Hence, the compilation process needs to adjust the coefficients so as to be able to have a single objective function corresponding to both deviations. We shall discuss this issue in greater detail later on.

Lastly, the LEX MIN operator indicates how the GP is to be solved. It means we start by minimizing only the first level objective. This provides values for the variables mentioned in the program. If there is only one level of objective functions, we are done. Otherwise, the first level objective function, f_1, is replaced by a constraint. If the optimal value obtained for the first level objective function is v, the constraint that replaces it takes the form $f_1 \leq v$. Then, the second level objective function is solved with respect to this newly added constraint and the original GP. This process repeats until all levels are solved. Intuitively, the added constraints ensure that the values of the solutions obtained at higher levels will not deteriorate by optimizing the lower levels; however, flexibility is retained so as to choose particular values for the decision variables that maintain the values of the higher level objectives and also meet the preferences of the lower level as much as possible.

One can naturally expect that a GP describing a deal will have a single level (as those at lower levels are infinitely less important, which means that they would probably not be specified). However, as we shall see, the ability to optimize level-wise is a key feature in using GPs to encode complex business desires. One limitation of GPs

Table 3 Evaluating two deal offers

Item	Attribute	Value for offer 1	Value for offer 2
Bricks	Days	17	5
Bricks	Number	1200	1000
Iron	Quantity	5000	5000
Overall	Price	19000	20625
Objective		(700, 3.1)	(725, 3.2)

is their linearity (for example, one cannot write itemUnitPrice · Quantity ≤US$20K. However, certain standard procedures may be used to handle many such situations essentially by breaking intervals into sub-intervals.

Once deals are expressed as GPs, we can compare and manipulate them. For example consider the purchasing deal we described earlier. Suppose we present this deal to two suppliers and obtain the offers shown in Table 3. Determining which of these offers is better is not a trivial question and humans often find it quite challenging to solve.

Further, suppose we wish to present a counteroffer to offer 1. Clearly, we would like to improve the situation for our side, but by how much? And how should we go about producing the counteroffer? A similar situation occurs when we consider an auction whose subject is a complex deal. For example, suppose the current offer of supplier S_1 to the bricks and iron deal is offer 1. Further, assume the buyer runs an auction among the potential suppliers. Currently a better offer was presented to the buyer by supplier S_2. Now, S_1 would like to present yet a better offer to the buyer. Again the same issues arise, by how much and how to generate the new offer. As we shall see in Sect. 3, once we have a GP as the mathematical description of a deal, the offer generation problem can be done quite efficiently. The decision on the degree of improvement is in the domain of game theory and will be discussed in Sect. 4.

2 Compiling deal specifications into a GP

Decision variables get their bounds either through the GUI level, or by interacting with the user and external sources. It is essential that each decision variable be bounded. Even if a certain variable is in principle unlimited from above, one should look for a reasonably large number to limit it. These bounds play an important role in defining the feasible region for the mathematical programs that are used to express the negotiation process. Unnecessarily large bounds may lead to overly large regions, which in turn extend the required computation time. On the other hand, tight bounds may limit the feasible region so severely as to preclude the possibility of finding an optimal solution. We usually use L (resp., U) to denote the lower (resp., upper) bound.

In this section we explain how the user's specification is compiled into a GP. As the number of constructs specifiable by the user is rather large, we'll focus on a few cases illustrating the basic ideas.

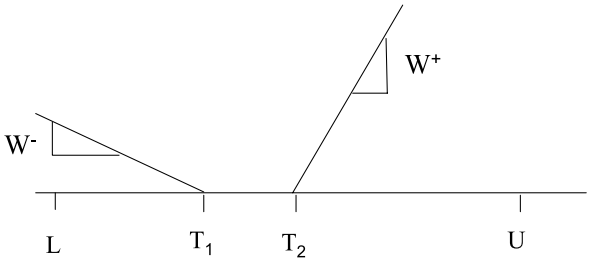

Fig. 2 Two-sided goals with interval range

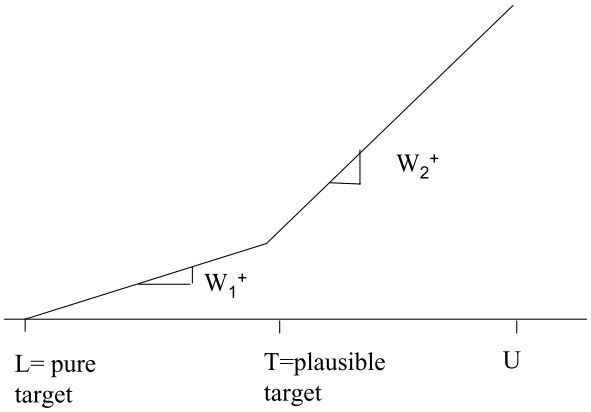

Fig. 3 Single-sided goals with multiple slopes

2.1 Two-sided goals with interval range

Here, we have a target range as depicted in Fig. 2. As a special case, when the interval is such that $T_1 = T_2$ we have a point interval. This case is formulated by:

$$\begin{aligned} \text{Min } & W^- \cdot \delta_1^- + W^+ \cdot \delta_2^+ \\ \text{s.t. } & X + \delta_1^- - \delta_1^+ = T_1, \\ & X + \delta_2^- - \delta_2^+ = T_2, \\ & L \le X \le U, \quad \delta_1^-,\ \delta_1^+, \delta_2^-, \delta_2^+ \ge 0, \\ & \text{other constraints.} \end{aligned}$$

2.2 Single-sided goals with multiple slopes

Unlike the previous case, here the user is capable of specifying the degree by which he actually prefers deviations from the theoretical target L up to the plausible target T vis-à-vis the deviations from the plausible target.[1] This case is depicted in Fig. 3.

[1] In general, we can also model multiple linear segments with various associated angles.

$$
\begin{aligned}
&\text{Min } W_1^+ \cdot \delta_1^+ + W_2^+ \cdot \delta_2^+ \\
&\text{s.t. } X - \delta_1^+ - \delta_2^+ = L, \\
&\quad \delta_1^+ \le T - L, \\
&\quad L \le X \le U, \quad \delta_1^+, \delta_2^+ \ge 0, \\
&\quad \text{other constraints.}
\end{aligned}
$$

2.3 Scaling

The scaling of a GP constraint aims to handle two problems (see [24, pp. 35–36]). The first problem concerns goals that deal with very different numerical values (e.g., price in millions of dollars vs. number of days for delivery). In such cases, we face difficulties in combining the corresponding deviation variables within the same objective function level. The second problem concerns objective functions that mix deviations that are associated with both discrete and continuous variables. To express ordinal preferences in the dimension of a discrete variable we use internal weights that are arbitrary with respect to the other variables that share the same level of the objective function. Hence, such mixing must be handled with care.

Scaling could be done by targets. That is, one can scale the deviation variables such that they express the amount of *relative* deviation from the target values of each goal. However, scaling by target, while reasonable, suffers from certain deficiencies (for example, inability to scale targets that are near zero). Therefore, we shall not explore it further in this paper.

Scaling can be done in interval terminology rather than by target. The advantage is that we can naturally handle a target of 0. For example suppose that D measures delay and we prefer zero delay. Suppose that D is in the range [700,850] where the target is 800. The corresponding GP fragment is shown below. Observe that the question to which the answer is $W = 5$ (as shown in the objective function below) is "what penalty would you assign to a deviation the size of the entire interval?"

$$
\begin{aligned}
&\text{Min } 5 \cdot \delta_p^- + \cdots \quad 5 \quad \text{is penalty for a deviation of 150 units} \\
&\text{s.t. } \frac{D}{150} + \delta_p^- - \delta_p^+ = \frac{800}{150} \quad \text{deviation measured as fraction of interval,} \\
&\quad 700 \le D \le 850.
\end{aligned}
$$

Effective intervals for negotiations Consider any form of negotiations between two parties. Each party specifies the importance in "interval units" prior to matching with other parties. The *effective interval* is the intersection of the intervals specified by the parties. Observe that when specifying, a party does not necessarily know which other parties' intervals will be encountered. We differentiate between resulting point intervals (consisting of a single point), small intervals (that is small compared to the original ranges), and normal intervals. In this paper we shall only cover normal intervals. Nevertheless, we point out that even if each party, separately, specifies a large interval, the effective one might be quite small in comparison.

Normal intervals Suppose a party specified an importance factor of 5 and its interval at specification time had 150 units as in the example above. Suppose the size of the intersection with another party's interval has just 50 units. First, if the "old" target is outside the intersection (e.g., the intersection is [705,755]), we "push" it towards the closest intersection boundary (e.g. from 800 to 755) thereby creating a new target. If the "old target" is within the new interval, we leave the target unchanged. Next, we modify the goal constraints and objective function to reflect the new interval. The resulting compilation, assuming a new interval of [705,755] and following the scheme above, is depicted below. Observe the changes in the target in the goal constraint and the new bounds on the interval. Also note the addition of the term $5{\cdot}(45/150)$ to the objective function due to a target shift from 800 to 755. This constant term does not affect the optimization and we may remove it in treating the goal program. It should be considered only when the original goal program is used for *listing offers*,[2] especially when comparing offers originating in negotiation sessions with different parties.

$$\text{Min } 5 \cdot \frac{45}{150} + 5 \cdot \delta_p^- + \cdots \quad 5 \text{ is the penalty for a deviation of 150 units}$$

$$\text{s.t.} \quad \frac{D - 705}{150} + \delta_p^- - \delta_p^+ = \frac{755 - 705}{150} \quad \text{deviation measured as fraction of interval,}$$

$$700 \le D \le 850.$$

Point intervals Point intervals are obtained when the intersection of the parties' intervals is a point. In this case, we substitute for the variable in question its value (the point) in the constraints. This also gives values to the deviations. We introduce the resulting constants into the objective functions. As before, we have the option of removing these constants altogether.

Small intervals The resulting interval may be "small". However, smallness is a relative term and what is small for party A may be large for party B. If both parties agree that the interval is small, then they may simply choose a mid-point between their targets (shifted into an interval boundary point if needed) and we are back to the case of a point interval. If the interval is large for both, we treat it as normal. If one party considers it small and the other as large, we treat it as normal.

2.4 Tradeoffs

Certain mixes of deviations from individual goals that belong to the same level of the objective function, associated with either penalties or rewards, may lead to tradeoff relations among combinations of variables (not necessarily pairs). For example, consider two variables X and Y, with targets T_x and T_y, respectively. There is a penalty on X values exceeding T_x and a bonus for values short of it. Conversely, there is a bonus on Y values exceeding T_y and a penalty for values short of it. In this case, we have two tradeoff expressions that naturally exist between X and Y. Notice that

[2] Note that we can do away with the -705 term that appears in both sides of the equation

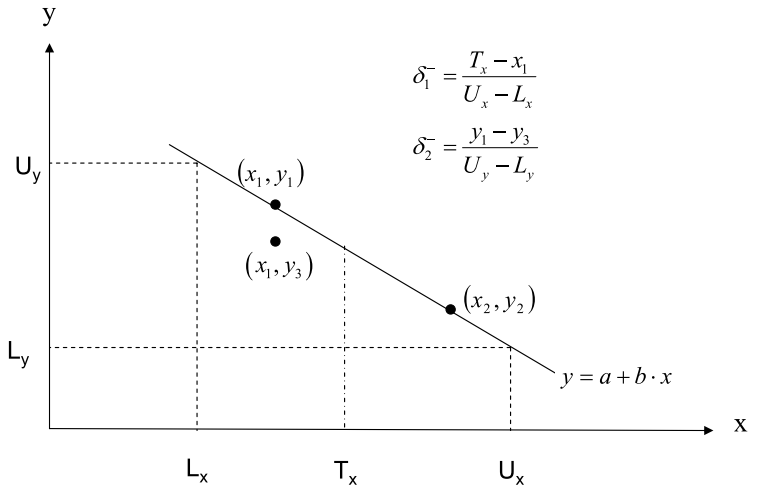

Fig. 4 Tradeoffs

there is no need to solicit separate information on tradeoffs or to explore all possible tradeoffs since the weighted deviation terms that are built into the objective function will naturally define tradeoffs where they are relevant.

There are other ways of expressing tradeoffs. Suppose that we have information about the target for X and the weights for deviations from it, while for Y the information is given in terms of a linear tradeoff relation with $X : T_y = a + b \cdot X$ along with the weights (either penalty or bonus) for being above or below the tradeoff line. This means that the target for Y is not static (as is the target T_x for X). Instead, the tradeoff relations define a "dynamic" target that depends on the value x_1 that variable X will assume. Given this target, deviations will lead to either a penalty or a bonus according to the original definition. In other words, we have transformed this case to the same case we handled earlier—where we have two variables with single-dimensional targets and deviations.

Inputs in this case are represented by Fig. 4. Suppose L_x and U_x (respectively, L_y and U_y) are the upper and lower limits of X (respectively, Y). The parameters a and b may be determined from the user specified points (x_1, y_1) and (x_2, y_2) as depicted in Fig. 4. For a point (x_1, y_3) that lies below the tradeoff line, the figure also depicts the relevant deviations: δ_1^- and δ_2^-. After proper scaling, the relevant elements in the GP should look like:

$$
\begin{aligned}
&\text{Min } W_1^- \cdot \delta_1^- + W_1^+ \cdot \delta_1^+ + W_2^- \cdot \delta_2^- + W_2^+ \cdot \delta_2^+ \\
&\text{s.t. } \frac{X}{U_x - L_x} + \delta_1^- - \delta_1^+ = \frac{T_x}{U_x - L_x}, \\
&\quad \frac{Y}{U_y - L_y} - \frac{b \cdot X}{U_y - L_y} + \delta_2^- - \delta_2^+ = \frac{a}{U_y - L_y}, \\
&\quad L_x \le X \le U_x, \\
&\quad L_y \le Y \le U_y.
\end{aligned}
$$

Inconsistencies When users specify pair-wise tradeoffs, there is a chance that inconsistent relations will enter the system. For example, the user states that the slope of the tradeoff between variables X and Y, and between X and Z are $+2$ and $+3$, respectively. Then, he specifies the tradeoff slope between Y and Z as $+5$. But, the slope implied through the first two tradeoffs is $+6$. There are ways to handle potential inconsistencies which are described in [28].

2.5 Discrete variable compilation

In the preceding subsections we described ordinary goals and tradeoffs associated with continuous variables only. Here, we deal with the problem of representing discrete-valued variables. In particular, we are interested in representing *string values*, such as the names of hotels in a city, or colors of the product, etc.

Let the feasible values for a variable t be $\{T_1 \dots T_m\}$ and the preference weights for every feasible value of t are $\{W_1 \dots W_m\}$, respectively. Suppose we are at the Lth level of the objective function and the relative importance of t within the level is V_t. First, we represent t through the binary variables, $b_i \in \{0, 1\}$, $1 \le i \le m$, as: $t = \sum_{i=1}^{m} T_i \cdot b_i$. This definition is kept *outside* the GP. We use it only for the purpose of translating the GP outputs in order to present them to the user (or to forward them to the agent of the other party). The other variables we define below will play a role in the GP.

$$\text{Hard constraints:} \quad x_t = \sum_{i=1}^{m} W_i \cdot b_i, \quad x_t \text{ is an auxiliary variable.}$$

$$\sum b_i = 1, \quad b_i \in \{0, 1\},\ 1 \le i \le m.$$

For a particular situation, be it 1-1 negotiations, an auction or some other application, we *relax* the model by replacing the binary variables b_i in the hard constraints above with continuous variables $0 \le c_i \le 1$, $1 \le i \le m$ and omitting the constraint that $b_i \in \{0, 1\}$. This results in a goal constraint and objective function as follows:

$$\text{Goal constraint (one-sided) } t\text{:} \quad x_t / W^{\max} + \delta^- - \delta^+ = 1.$$

The factor $W^{\max}$ is the maximal weight in $\{W_1 \dots W_m\}$. It also bounds x_t.

$$\text{Objective (at level L):} \quad \text{Min } Z = V_t \cdot \delta^- + \cdots.$$

Example Suppose that the variable t represents the color of bricks, where the colors Red, Brown, Gray, Yellow and Black are represented using the values one to five, respectively. A seller can deliver only Gray, Yellow, and Black, with the following preference weights: $\{W_{\text{Gray}} = 1,\ W_{\text{Yellow}} = 8,\ W_{\text{Black}} = 3\}$. Also suppose that the seller asked for some level of objective L, and some relative importance within that

level V_t, and that the seller cannot provide bricks of any of the other colors.

Definition of t (color):	$t = 3 \cdot b_1 + 4 \cdot b_2 + 5 \cdot b_3$,
Definition of x (desirability):	$x = 1 \cdot c_1 + 8 \cdot c_2 + 3 \cdot c_3, \quad 0 \le x \le 6$,
Definition of c's:	$c_1 + c_2 + c_3 = 1, \quad 0 \le c_i \le 1, \ i = 1, 2, 3$,
Goal constraint:	$x/8 + \delta^- = \delta^+ = 1$,
Objective (level L) :	Min $Z = V_t \cdot \delta^-$.

Note that in this GP we are using continuous variables (c_i's). Upon completion of negotiations, the continuous c_i's need be converted into binary b_i's. The conversion details are intricate and will be omitted here, the underlying principle is straightforward. We view the conversion as an optimization problem in which the b_i's need be assigned 0-1 values in such a way as to preserve as much as possible the objective functions' values achieved by the negotiators.

2.6 Hannan's method and non-zero deviation pairs

This method (see [8, 10]) tries to maximize the values of the deviation-variables for which there is an *implicit preference* to move in their respective direction (syntactically, these are deviation variables that do not participate in any objective function). We may consider the Lex Min objective-vector as a requirement to minimize the cost of deviating from the targets set on the goal-constraints in the *undesirable* direction (minus or plus). Thus, Hannan's method tries to maximize the benefit of deviating from the target in the *desirable* direction. Therefore the "Hannan objective", denoted $Hanan$, is of the form: $\{-\sum \delta_i^-\} + \{-\sum \delta_j^+\}$ for those δ_i^-, δ_j^+ that did not appear in any objective function of the original GP problem.

In some GPs, it is possible for a pair of deviation variables that correspond to the same decision variable to concurrently be assigned non-zero values. Logically, deviation cannot be both ways and indeed, in regular GPs it will never happen since the columns that correspond to each pair of deviation variables are linearly dependent. However, if a member of the pair appears (alone) in another constraint or in the objective function, such a problem may occur. So, we have devised constraints, to prevent such irregular cases. Intuitively, one may think that Hannan's method may provide a remedy to the non-zero deviation pairs problem. However, we can show that using Hanan's objective does not necessarily prevent the problem. Hence, both Hannan's objective and the non-zero deviation constraints need to be explicitly enforced.

2.7 OR conditions (disjunctive compilation)

The user may specify several disjunctive constraints that need to be compiled into the GP model. This is a challenging task since all the constraints of a GP model are, by definition, implicitly conjuncts. Therefore, we need to provide a translation method for a disjunctive expression so that we can require the satisfaction of an arbitrary subset of the set of constraints; i.e, enabling solutions to problems such as asking to satisfy exactly three constraints out of seven or asking to satisfy at least two such constraints out of five, etc. Moreover, this translation must consist of only linear expressions, as we are restricting ourselves to integer and linear-programming capabilities. Solutions are presented in [28].

3 Deal manipulation

This section describes an array of operators for manipulating deals. The motivation is to use these operators within EC applications (e.g., deal evaluation, deal comparison, producing a deal to specifications, choosing from a catalog, etc.). Generally, the manipulations support an application in which a deal is being negotiated. We use the names "Bob" and "Sue" to denote the negotiating agents. Nevertheless, the procedures may apply to arbitrary parties, buyers, sellers and also symmetric agents (e.g., bartering).

3.1 Notation

x—A vector of decision variables.
X—A vector of values for decision variables.
d—A vector of deviation variables of a goal program.
D—A vector of values for deviation variables of a goal program.
$[X|D]$—A combined presentation of X and D above.
$\alpha(\bar{\alpha})$—Prioritized vector $(\alpha_1, \ldots, \alpha_k)$ of objective functions (values) for Bob where α_i is the objective expression at level i.
$\beta(\bar{\beta})$—Prioritized vector of objective functions (values) for Sue.
F_α (F_β)—Set of goal constraints that link the elements of x with their respective deviation variables in Bob's (Sue's) program. By writing $x \in F_\alpha$ we loosely mean that x and the associated deviation variables must satisfy this set of constraints.
S_B (S_S)—Set of hard constraints in Bob's (Sue's) program.
$P^{\text{type}}_{\text{agent}}$—A proposal of type $\in$ {"new", "last"} made by either by Bob or by Sue. It contains (1) A *value tuple* $t = [X|D]$ of values for the decision, X, and deviation variables, D, and (2) A *tuple* $\bar{\alpha}$ of the objective-functions values for α.
$X^{\text{type}}_{\text{agent}}$—A vector of values of the decision variables associated with $P^{\text{type}}_{\text{agent}}$.
$\alpha^{\text{type}}_{\text{agent}}$, $\beta^{\text{type}}_{\text{agent}}$—A vector of values for the priority levels, of Bob and Sue respectively, associated with $P^{\text{type}}_{\text{agent}}$.
α^*, β^*—A best possible vector of values for the priority levels of Bob and Sue respectively.
α_*, β_*—A worst possible vector of values for the priority levels of Bob and Sue respectively.
ρ_{SS}—Proportion of improvement in Sue's utility (β) with respect to (β^{last}_S).
ρ_{BS}—Proportion of improvement in Bob's utility (α) with respect to (α^{last}_S).

3.2 Solving a GP (*solve*)

This basic utility procedure employs an ordinary linear programming procedure (e.g., the Simplex algorithm) to solve a general-purpose GP. Its input is a GP and its output is a proposed solution $P^{\text{new}}_{\text{agent}}$ which consists of a *tuple* $\bar{\alpha}$ of the optimal objective-function values and a *tuple* $t = [X \mid D]$ of the corresponding values for the decision and deviation variables, respectively.

3.3 Building a deal to specification (*suggest)*

The utility takes as inputs a *mode* parameter: $mode \in \{Optimal, Worst, Any, Percent, Average\}$ and a GP. If $mode = Percent$, then it takes as additional inputs a vector $\bar{\alpha}^0$ of objective function values to improve upon and a percentage ρ. Its output is a proposed solution P_B^{new}, consisting of a *tuple* $t = [X \mid D]$ of values for the decision and deviation variables, and a *tuple* $\bar{\alpha}$ of the objective-function values.

The function suggests a solution, according to the mode of operation, as follows:

Optimal The optimal solution for the GP, denoted as $t^* = [D^* \mid X^*]$, is obtained by calling (GP).

Worst Here the utility returns the worst possible soultion (e.g., as the first offer for the other party). To do so, it first inverts the objective functions in the GP from Min to Max (done simply by multiplying each of these functions by -1). Then, it calls *solve*(GP) and returns its outputs: $t_* = [D_* \mid X_*]$ and the corresponding $\bar{\alpha}$.

Any Here, the utility derives an arbitrary feasible solution. It does so by randomly choosing one out of every pair of deviation variables of the original GP and randomly assigning it a coefficient in the range $[0, b]$ where b is a system parameter. It creates a new objective function as follows; it inserts the deviation variable multiplied by its coefficient into the objective function. The coefficient for the other member of the pair is set to zero. Finally, it calls *solve* (GP) and returns its output.

Percent This mode derives a solution that is worse than $\bar{\alpha}^0$ values by no more than $\rho\%$. The proposed solution reduces each objective at a time, using the same percentage $\rho\%$ for all objectives. To obtain the solution we add to the GP the following goal-constraints and bounds. For every objective function α_i and its value $\bar{\alpha}_i^0$, we add the goal constraint: $\alpha_i + \delta_i^- - \delta_i^+ = \bar{\alpha}_i^0 + \rho \cdot |\bar{\alpha}_i^0|$. We replace the objective-function vector of the GP by inserting the function $\sum_{i=1}^{k} (\omega_i^+ \delta_i^+ + \omega_i^- \delta_i^-)$ as the first priority level; here the weights ω_i^+, ω_i^- are such that the lower the index i, the higher the weight, e.g., use powers of ten, $w_i = 10^{-i}$. Afterwards, we set the rest of the priority levels according to the original α objective function (that is, each function in the original objective is pushed one level down). Finally, we call *solve*(GP) and return its outputs.

Average This utility derives a solution that is as close as possible to the average (between the optimal and the worst solutions) of the objective function values. The derived solution will be feasible since it is generated by a GP formulation where feasibility is guaranteed. The utility first finds the optimal solution X^*, $\bar{\alpha}^*$ and the worst solution X_*, $\bar{\alpha}_*$. Then, it adds for every objective function α_i in the GP the following goal-constraints and bounds, $\alpha_i + \delta_i^- - \delta_i^+ = (\bar{\alpha}_i^* + \bar{\alpha}_{i*})/2$. At this stage the utility replaces the objective-function vector of GP as done in *Percent* mode above.

3.4 Ranking solutions (*rank*)

This procedure is used for comparing assignments to decision variables (referred to as *offers*). Its inputs are a GP and a set of h tuples of values $\{t^j = [X^j | D^j]\}$[3] $1 \leq j \leq h$. The outputs include a sorted set of h tuples of values, $\{t^{j'} = [X^{j'} | D^{j'}]\}$, each with its rank number. The utility calculates the objective function values for each tuple, and performs a lexicographical ordering of the objective vectors to sort the tuples. As a result, the tuples are ordered from most preferred to least preferred. Observe that by employing *rank* we can implement an operator that returns the best m vectors, and for $m = 1$ the best vector (of course, in that case we can simply solve the optimization problem). We shall skip the details of the GP construction for *rank*.

An advantage of lexicographically ordering goals is that ties at a high level are often broken at lower levels. In the unlikely case that ties between assignments are not broken, the ordering is arbitrary. This is justified since such assignments, from the decision maker point of view, represent points along an indifference curve. Of course, one can add additional rules for tie breaking.

3.5 Initializing deals in 1-1 negotiations (*initialize*)

One-to-one (1-1) negotiations are a fundamental building block in EC. Conceptually, it is carried out between agents that agree on the definition of decision variables. Each agent has a specification (a GP in our case) of its constraints, preferences and tradeoffs. Each agent may also possess knowledge regarding the other agent. The particular knowledge may be complete (*informed*), empty (*ignorant*) or somewhere in between. We shall treat explicitly the operators for "ignorant" and "informed". The ideas can be extended to handle arbitrary partial knowledge states regarding the other agent's GP. One assumption we make is that negotiations proceed level-by-level, that is, first both parties negotiate over their respective first (and most important) level, then they proceed to their respective second level, and so forth. Switching levels is a delicate operation which we shall examine more closely. Below we explain how to formulate the *first* offer in a negotiation session. We describe how an "informed" Sue generates an offer for Bob (of type "any", i.e. either "ignorant" or "informed").

3.5.1 First-offer-at-level-k (agent is "informed", opponent is "any") (Sue is constructing the offer)

The inputs are the agent's GP (GP_β), the opponent's GP (GP_α) and the new level, k, of negotiation. The outputs are the proposed solution P_S^{new}, which consists of a *tuple* $t' = \lfloor X' \mid D' \rfloor$ of values for the variables and a *tuple* $\bar{\alpha}'$ of the corresponding objective function values (for Bob).

This utility is used by an "informed" agent to construct the very first offer and additional first offers upon switching levels in the objective function. Essentially, the utility combines the constraints of the GPs of both agents. It finds the best offer for

[3] The tuples may be partial or complete.

Sue given these combined constraints and Sue's objective functions. It then incorporates Sue's achievements as new constraints and solves these modified constraints and Bob's objective functions for the *worst* offer for Bob. The precise scheme in which this is done is as follows: invoke *solve* for best and worst solutions (level-by-level), working on the first effective level of Sue's objective ((k), that is specified in the input), followed by the first effective level of Bob's objective, then the second effective level of both objectives, etc.[4]

3.5.2 First-offer-at-level-k (agent is "ignorant", opponent is "any") (Sue is constructing the offer)

When the agent is "ignorant", it computes the very first offer of the whole negotiation by calling the *Suggest*(*Optimal*) utility. In contrast to the "informed" case, special care need be taken when switching levels since Sue doesn't know Bob's objective functions.

When switching from one level to the next there is a need to update some of the system parameters in light of what was agreed upon in the previous level(s). Specifically, the My_best and My_worst values for the present lexicographical level (denoted as α^* and α_*) might change as a result of the $\bar{\alpha}$ values that were achieved at higher levels and the presence of hard constraints that link variables across levels.

To find the updated values of α^* and α_* for an "ignorant" agent we add an equality-type constraint rather than an inequality-type constraint (i.e., at the kth level, add the constraint $\alpha_{k-1} = \bar{\alpha}_{k-1}$). This is necessary since our ignorant agent (Sue) can not incorporate constraints that represent the targets of her opponent (Bob). Hence, we must force Sue to stick to the $\bar{\alpha}_{k-1}$ value that was achieved earlier since otherwise she will try to improve it and may end up without any effective changes in the α^* and α_* of her current level. This difficulty is not present when the agent is "informed" since the latter incorporates the constraints representing his own $\bar{\alpha}_k$ value and the $\bar{\beta}_k$ value of his opponent in his GP. Thus, in the "informed"—improve implementation, presented later on, both of these additional constraints can be written as inequalities.

3.6 Manipulation of deals (improve utilities)

These utilities are intricate methods for shaping deals that take into account the state of negotiations, for example previous offers made by the parties. In these operators, again, we differentiate between versions specialized to parties' state of knowledge. The layer that calls these improve operators decides which improvement style is preferable. Intuitively, an "ignorant" party may find it difficult to improve on his previous offer for the other party due to his ignorance as to the other party's GP. The opposite holds for an "informed" party. However, an informed party will usually not blindly improve the offer for the other party. Rather, he is likely to apply some constraints on the worsening of his own position, as judged by his own GP. Note that these improve procedures may be turned into "worsening" procedures by changing the directions of the objective functions; for brevity we do not detail these changes.

[4] We assume that when the function is called after agreeing on the $(k-1)$ level, then both GPs already contain the achievement values of the previous k objective levels in the form of constraints.

3.6.1 Improve (agent is "ignorant", opponent is "any") (Bob's viewpoint)

The inputs include: Bob's GP; P_B^{last}—the previous offer Bob made to Sue which serves as a reference point; a tuple X_S^{last} of values for the decision variables that appeared in Sue's last offer; a percentage ρ; α^*—the optimal objective values of Bob; α_*—the worst objective values of Bob; *gap_flag*—a flag to determine the way to calculate the new objective function values (possible values are: *fixed, dynamic*); *Max_gap*—the maximal value allowed for changing the objective function; *Min_gap*—the minimal value allowed for changing the objective function; gc—the gap constant for the fixed *gap_flag*; i—the current objective level under negotiation.

The output is a proposed solution P_B^{new}, which consists of: a *tuple* $t' = \lfloor X' \mid D' \rfloor$ of values for the decision and deviation variables; a *tuple* $\bar{\alpha}'$ of the corresponding objective-function values.

Given the input objective values (corresponding to objective function α on P_B^{last}), this utility finds a new objective function value $\bar{\alpha}'$, that worsens the values of P_B^{last} for Bob by at most $\rho\%$.[5]

We illustrate this utility via an example. Recall the construction GP of Sect. 2. Assume that this is Bob's GP. Suppose Bob solves this GP and produces his optimal offer (14, 1100, 6000, 20000) for (bricksDays, bricksNumber, ironQuantity, overallPrice). This offer, offer-1, has the two objective functions at 0. Suppose that Sue responds with a counteroffer: (17, 1000, 5000, 24000). In Bob's GP, Sue's offer rates as (4100, 3.2) for the two objective levels. We now explain how Bob constructs a new GP and then responds based on its solution.

The new GP is based on two principles. First, Bob is ready to worsen his position, by a certain percentage, in order to reach a deal with Sue. This is manifested in the first objective level of the new GP. Second, Bob is not knowledgeable regarding Sue's preferences. Therefore, Bob uses Sue's last offer as a hint as to the kind of values that are acceptable to Sue, and all else being equal, Bob attempts to have the values of his counteroffer close to the corresponding values in Sue's last offer. This is manifested in the second objective level. The new GP is illustrated in Table 4. The two objective functions reflect the above two principles. The original GP objectives are now stated as goals. The first one repeats the formulation of Bob's original top level objective and attempts to arrive at a value of 410. That value, 410, is obtained as ρ times the difference between the objective value in Bob's first offer, namely zero, and the first level rating of Sue's counteroffer, namely 4100, where ρ is taken as 10%. In addition, four "stay close" goals, one for each decision variable, are added. In general, the coefficients of the γ deviation variables may be different in the second level objective based on the (Bob's perceived) importance of issues for Sue. The coefficients of the first level objective give preference to progressing towards Sue while worsening.

By solving the new GP, Bob generates a new offer, offer-2, (17, 1000, 5000, 20305). In generating this offer, the new GP first level evaluated to zero as it succeeded in reaching 410 for the oldTopObjective goal without assigning nonzero values to the Z deviation variables. Recall that Bob's offer-1 was (14, 1100, 6000, 20000).

[5] The same percentage may be used for all levels of the objectives list.

Table 4 The new GP for constructing a counteroffer

LEX MIN

Level 1: $0.6Z^- + 1.0Z^+$

Level 2: $\gamma_1^- + \gamma_1^+ + \gamma_2^- + \gamma_2^+ \gamma_3^- + \gamma_3^+ + \gamma_4^- + \gamma_4^+$

GOAL CONSTRAINTS:

oldTopObjective: $0.6\beta^- + 1.0\beta^+ + 0.10\delta^- + Z^- + Z^+ = (4100 - 0) * (\rho = 0.1) = 410$

oldSecondObjective: $0.6\pi_1^- + 1.0\pi_2^+ + 0.001\epsilon^- + 0.002\epsilon^+ + M^- + M^+ = 0$

stayClose-1: brickDays $+\gamma_1^- - \gamma_1^+ = 17$

stayClose-2: brickNumber $+\gamma_2^- - \gamma_2^+ = 1000$

stayClose-3: ironQuantity $+\gamma_3^- - \gamma_3^+ = 5000$

stayClose-4: overallPrice $+\gamma_4^- - \gamma_4^+ = 24000$

brickDays $+\pi_1^- - \pi_1^+ = 10$

brickDays $-\pi_2^+ + \pi_2^- = 14$

brickNumber $+\epsilon^- - \epsilon^+ = 1100$

ironQuantity $+\delta^- = 6000$

overallPrice $+\beta^- - \beta^+ = 20000$

HARD CONSTRAINTS:

$5 \leq$ brickDays ≤ 25

$1000 \leq$ brickNumber ≤ 1300

$5000 \leq$ ironQuantity ≤ 6000

$17000 \leq$ overallPrice ≤ 25000

Suppose Sue reacts now with (17, 1000, 5500, 22000). For the first level objective function in Bob's original GP, the value is 2050. Previously Bob aspired to a value of 410 for this function and in fact his offer-2 had precisely this value. Sue's current offer sets a value of 2050 to Bob's original GP first level objective. The difference is therefore $2050 - 410 = 1640$. Bob is therefore ready to worsen his position from 410 to $410 + 0.1 \cdot 1640 = 574$. So, Bob solves again his new GP, with the following differences: (1) the value for oldTopObjective is now 574, (2) the "stay close" goals now reflect Sue's latest offer—(17, 1000, 5500, 22000). Solving the modified new GP, Bob obtains (17, 1000, 5500, 20524), as offer-3, which actually achieves 574 without using nonzero values for the Z deviation variables.

Suppose Sue reacts with (17, 1000, 5400, 20525) yielding a value of 585 for the first objective function in Bob's original GP. Previously, Bob aspired to a value of 574 for this function (in fact, his offer-3 had precisely this value). The difference between these two values is $585 - 574 = 11$. Bob is now ready to worsen his position from 574 to $574 + 0.1 \cdot 11 = 575.1$. So, Bob solves again a new GP, with the following differences: (1) the value for oldTopObjective is now 575.1, (2) the "stay close" goals reflect Sue's latest offer—(17, 1000, 5400, 20525). Solving the modified GP, Bob obtains (17, 1000, 5400, 20515.1) as offer-4. Sue accepts this offer and the negotiations terminate.

3.6.2 Improve (agent is informed, opponent is any) (Sue's viewpoint)

The inputs to this utility include: Sue's GP, (GP_β); P_S^{last}, the *reference proposal*, that is, the previous offer Sue made which serves as a reference point; The opponent's (Bob's) GP;[6] GP_α; the opponent agent's (Bob's) optimal result α^*;[7] a tuple X_B^{last} of values for the decision variables (Bob's latest offer); a percentage ρ_{BS} for the maximum improvement to the opponent (Bob) objective; a percentage ρ_{SS} for the maximum worsening of the agent (Sue) objective; Sue's optimal result β^*; i—the current level under negotiation. The outputs are a proposed solution P_S^{new}, which consists of: a *tuple* $t' = \lfloor X' \mid D' \rfloor$ of values for the variables and a *tuple* $\bar{\beta}'$ of the corresponding objective-function values.

We shall not detail the precise process of forming a new GP but merely sketch it. Given input values (corresponding to Sue's objective function β), this utility constructs a new objective function β' that attempts to improve the values of the objective function value for Bob by at least $\rho\%$. Taking into consideration Bob's constraints and goals as represented by his GP, it tries to improve the value of the opponent's objectives in a controlled manner.

In both *improve* procedures given above there is a reference to the proposing agent's prior offer and to its opponent's prior offer. A natural question is how are these values initialized the first time during a negotiation session when such prior offers do not actually exist. We omit the details of this fairly straightforward initialization.

4 The application layer

This layer uses the GP foundation described in the previous sections to realize efficient commerce applications. In this paper we address the simpler problem of negotiating a single, yet multi-dimensional, item. The more general problem, that of negotiating a collection of items, is similar in nature but more complex mathematically. We first address auction mechanisms. Specifically, we treat the well-known second-price auction by using the basic operator, *suggest* (described in the previous section), to help bidders shape their multi-dimensional offers. Next, we address the very complex, and less understood theoretically, topic of 1-1 negotiations. We deliberately avoid the description of the *strategic level*, that is when and by how much to modify offers in response to the other party's actions as this subject is out of the scope of this paper.

4.1 One-to-N mechanisms

Here we discuss procedures for markets characterized by a single seller and several buyers where the goal is to maximize the seller's revenue. The analog mirror images

[6] The information that is known about the opponent is subject to the opponent's strategy. For example, the opponent may not provide all of his preferences (constraints or objectives), or even may not provide true preferences!

[7] Calculated according to Sue's knowledge about Bob.

of these procedures are suitable for the case of one buyer and a few sellers. We assume the private value model (that is, the valuation of offers by an agent is independent of the way other agents view the offer).[8]

In a *second price auction* [35] the seller chooses a reservation price r, which is then revealed to all buyers. The buyers simultaneously submit bids. We denote by p_j the bid of buyer j, $j = 1, 2, \ldots, N$. If all bids are below the seller's reservation price, then the procedure is terminated with no trade. Otherwise, the winner is the bidder whose bid was the highest (if there is a tie then the winner is chosen by a random device). If buyer k is the winner then he pays $\text{Max}[\text{Max}_{j \neq k}\{p_j\}, r]$ to the seller and the other buyers pay zero. In case more than one item, say l items, are the subject of the auction (either sold or bought by the auctioneer) while each bidder supplies/buys a single item, the price is that of the losing bidder with the highest bid.

In case the only issue to agree upon is price, then all parties understand the preferences of the others. The seller is interested in a higher price and the buyers prefer a lower price. This is not the case when we have multi-dimensional deals. There, we denote a buyer's value function by $f(\cdot)$ and that of the seller by $g(\cdot)$ where both $f(\cdot)$ and $g(\cdot)$ represent penalties (i.e., weighted deviations from targets). So, in terms of these penalty functions, "less means better". In such a case, it is important to first reveal the value function $g(\cdot)$ of the seller to the bidders. One possibility is to reveal to the bidders the true value of function $g(\cdot)$, which was submitted by the seller.[9]

First, the seller reveals his $g(\cdot)$ (by publishing his constraints, preferences and objectives possibly arranged in $K \leq 2$ levels in lexicographical order) and his reservation price (given in terms of maximal allowed values for his objective function levels—g_k). The limitation on K is because the auction utility (to be explained shortly) concentrates on the first level objective function and hence actually works in a mode where all important issues, properly weighted, should be at level 1, and all the other issues at level 2.

Then, the buyers bid. Let $b = (b_1, \ldots, b_n)$ denote the vector of submitted bids where each bid is a value for the seller's first level objective function. Let $f_{i1}(\cdot)$ be bounds (worst acceptable values) on the first level value function of bidder i, GP_i be his corresponding goal program, with objective function α_i and GP_A be the goal program, with objective functions β, of the auctioneer. Then, to generate a bid, bidder i solves a program that maximizes the seller's utility subject to his own constraints as well as those of the seller (i.e., both GP_i and GP_A) and while not crossing the threshold values specified for his own objective function at level 1 as well as the thresholds (i.e., reservation bounds) for level 1 that were specified by the seller (g_1):

$$
\begin{aligned}
&\text{Lex Min } \{\beta_1\}, \{\beta_2\}, \\
&\qquad \alpha_{i1} \leq f_{i1}, \\
&\qquad \beta_1 \leq g_1, \\
&\qquad \text{constraints of } \text{GP}_i, \\
&\qquad \text{constraints of } \text{GP}_A.
\end{aligned}
$$

[8] Mechanisms to support auctions when values are interdependent are presented in [28].

[9] Another possibility, which we shall not discuss here, is to give the seller an option to reveal a modified value function according to his strategic choice.

The first hard constraint forces the solution to meet the bidder's threshold specifications (f_{i1}) at the 1st level of his objective function. The second constraint restricts the bid so that it gives the auctioneer at least what he announced as his threshold level g_1. In the objective function of the program above, the bidder tries to respond (the response being values for the seller's two objective functions), as best as he can under the constraints, to the auctioneer's two objectives (according to their lexicographical order).

The auctioneer selects the winning bid as the one associated with the smallest value for $g(\cdot)$ (the lexicographic objective function). Let bidder m be the winner. That is $b_m = \text{argmin}_i[g(b_i)]$. Let s denote the second winning bidder. That is, $b_s = \text{argmin}_{i \neq m}[g(b_i)]$. The traded deal, denoted by d, is then obtained by solving (1 and 2 denote levels, A is the auctioneer and b_m is the wining bidder):

$$\begin{aligned} \text{Lex Min } & \alpha_{m1}, \alpha_{m2}, \beta_{A1}, \beta_{A2}, \\ & \beta_{A1} \leq g_1(b_s), \\ & \text{constraints of GP}_m, \\ & \text{constraints of GP}_A. \end{aligned}$$

In other words, the winning buyer selects values for the vector of decision variables x so as to optimize his own objective function, subject to the constraint that the selected values will give the seller at level 1 at least the utility obtained through b_s. Following that, there will be an attempt to optimize the seller's utility, at the next priority levels.

4.2 One-to-one negotiations

One-to-one (1-1) is a complex trading mechanism as compared to one-to-N (auctions). Part of this complication is due to issues of symmetry—who starts, who responds, at what intervals, how to respond and when and how to terminate. The actual negotiation behavior of an agent is determined by its *negotiation classification parameters* which include various negotiation-oriented parameters and constants and its *operational profile* which contains its strategic-level scenario. This information is used to dictate how to respond to the other side's offers (for example, percentage of offer improvement). Here, the other side's offers are essentially values for the decision variables. An agent's *profile* may be constructed in many ways (e.g., choose from a pre-determined set of profiles or by soliciting answers to certain questions).

We assume that the goal of each negotiating agent is to reach a deal with the highest possible value of his value function (in the context of GP, the highest possible value is achieved when the objective functions are minimized). The knowledge each party possesses regarding the other party is an essential factor in negotiations. If each agent may be either "informed" or "ignorant" of the other side's value function, there are four basic knowledge state possibilities. Considering the seller's problem, we have a *value function* instead of price. (When negotiations proceed level-by-level, this value function is specific to the current level that is being discussed.) In general, we have a buyer, say Bob, with a value function α and a seller, say Sue, with a value function β (where α and β are not necessarily identical). Previously we described various operators that may be used to form specific offers based on the agent's profile

and its knowledge state. These operators are used handle various situations that occur within 1-1 negotiations.

Rules for starting the negotiation process Negotiations commence by the party who wishes to start (or at random if both parties were willing to start). The initial proposal is designed to present a firm opening by, at each lexicographic level, obtaining the best for the opening side and the worst for the opponent. The underlying idea is that compromise mechanisms embedded in the profiles will soften this harsh opening. The initial values in the first offer are specified by repeated calls to the *solve* routine. A knowledgable agent will iterate between solving for the best value for itself and solving for the worst value for the other side (for each priority level) while an ignorant agent will lexicographically solve for its best value along the various priority levels. An ignorant agent may utilize operators that attempt to respond cooperatively, sya using the "stay close" goals.

Rules for ending the negotiation process Negotiations may be stopped either due to a successful conclusion (a deal is reached), or due to technical reasons such as the passage of time, the 'end of the profile' or the lack of progress. We outline a representative collection of criteria for either accepting an offer or for stopping the exchange of offers and counteroffers. In case of stopping, we have the option of considering the negotiations as failed or, alternatively, we may generate a compromise offer based on the current state of the negotiations as well as the original intentions of the parties.

Acceptance criteria for proposition acceptance Accepting the other party's offer can be determined through several criteria. For example, the difference between Sue's next offer to Bob's last offer is smaller than a pre-defined threshold; Bob's last offer is better (for Sue) than the offer Sue is about to suggest; the quality of the offer that Sue received is higher than the quality of offers Sue was ready to accept at this point in the negotiation.

Stopping rules Stopping the exchange of offers can also be based on a collection of criteria such as: the previous K offers Sue has received are similar (according to a predefined parameter) to each other; Sue has reached a deadline that she has specified for this negotiation process or for the entire deal negotiation; there may also be other user directive, either human generated or tool generated.

5 Conclusions and future directions

Having a common framework to describe commercial transactions ("deals") is highly important in understanding and implementing current applications as well as a foundation for future ones. It is analogous to defining atoms (GPs describing constraints, preferences and tradeoffs) in Chemistry, then forming various molecules (operators such as *suggest* and *improve*) and then discussing compounds (EC applications).

This paper outlines a comprehensive framework for supporting many EC applications. The formalism is based on GoP, a well-researched branch of Operations

Research. We showed that GoP is highly suitable for describing, manipulating and implementing various negotiation moves. Here the term 'negotiation' is used in a very general sense to describe many EC activities such as deal comparison, choosing from a catalog, auctions, 1-1 negotiations, deal splitting and more.

In fact, we have defined many more operators and showed how to use them in applications [28]. Nevertheless, much work remains. First, on the basic technical level, GPs are linear. Some applications are non-linear in nature and can only be approximated using the GoP formalism. It will be interesting to analyze a useful class of non-linear GPs. We handle non-linearity in goals by using piecewise linear approximations (see Fig. 3). When non-linearity occurs in the constraints (e.g., $X \cdot Y = K$), a similar technique can be used (it may yield a more extensive computational effort). Second, substantial experimentation with systems built on this methodology is needed. Third, as web services become prevalent, introducing these techniques at the architectural level is an important issue, some preliminary work is reported in [29].

Acknowledgement The authors thank Professor Motty Perry of the Hebrew University in Jerusalem for many illuminating discussions.

References

1. Beam, C., & Segev, A. (1997). Automated negotiations: a survey of the state of the art. *Wirtschafts-Informatik*, *3*(97), 263–267.
2. Chen, E., Kersten, G. E., & Vahidov, R. (2003). An e-marketplace for agent-supported commerce negotiations. In *INR08/03, InterNeg*.
3. Kersten, G. E., & Noronha, S. J. (1999). WWW-based negotiation support: design implementation, and use. *Decision Support Systems*, *25*, 135–154.
4. Christensen, E., Curbera, F., Meredith, G., & Weerawarana, S. (2001). *Web services description language (WSDL) 1.1*. W3C Note 15, March 2001. http://www.w3.org/TR/2001/NOTE-wsdl-20010315.
5. Faratin, P. (2000). *Automated service negotiation between autonomous computational agents*. PhD thesis, University College London, London, UK.
6. Faratin, P., Sierra, C., & Jennings, N. R. (2002). Using similarity criteria to make issue trade-offs in automated negotiations. *Artificial Intelligence*, *142*(2), 205–237.
7. Golany, B., & Shmueli, O. (2005). Catalog-based purchasing: illustrating a quantitative approach to electronic commerce. In *IEEE CEC 2005, 7th international IEEE conference on e-commerce technology 2005*, Technische Universitat Munchen, Germany, 19–22 July 2005.
8. Hannan, E. L. (1980). Non-dominance in goal programming. *INFOR*, *18*(4), 300–309.
9. Hannan, E. L., & Narasimhan, R. (1981). On fuzzy goal programming. *Decision Sciences*, *12*(3), 522–531.
10. Hannan, E. L. (1985). An assessment of some criticisms of goal programming. *Computers & Operations Research*, *12*(6), 525–541.
11. Heiskanen, P. (1999). Decentralized method for computing Pareto solutions in multi-party negotiations. *European Journal of Operational Research*, *117*(3), 578–590.
12. Hammond, K. R. (1965). New directions in research on conflict resolution. *The Journal of Social Issues*, *21*, 44–66.
13. IBM (2000). *Web services architecture overview. the next stage of evolution for e-business*. IBM Web Services Architecture Team, September 2000. http://www106.ibm.com/developerworks/webservices/library/.
14. Ignizio, J. P. (1976). *Goal programming and extensions*. Lexington Books.
15. Ignizio, J. P., & Cavalier, T. M. (1994). *Linear programming*. New York: Prentice-Hall.
16. Kersten, G. E., & Szpakowicz, S. (1998). Modeling business negotiations for e-commerce. In *INR05/98, InterNeg*.

17. Kersten, G. E., Strecker, S. E., & Law, K. P. (2004). Protocols for electronic negotiation systems: Theoretical foundations and design issues. In *Lecture notes in computer science* (Vol. 3182, pp. 106–115).
18. Konopnicki, D., Leiba, L., Shmueli, O., & Sagiv, Y. (2002). A formal yet practical approach to electronic commerce. *International Journal of Cooperative Information Systems*, *11*(1–2), 93–117.
19. Kraus, S., & Lehmann, D. (1995). Designing and building a negotiating automated agent. *Computational Intelligence*, *11*(1), 132–171.
20. Lee, S. M. (1972). *Goal programming for decision analysis*. Philadelphia: Auerbach.
21. Myerson, R. B. (1997). *Game theory—analysis of conflict*. Cambridge: Harvard University Press.
22. Osborn, M. J., & Rubinstein, A. (1999). *A course in game theory*. Cambridge: The MIT Press.
23. Raiffa, H. (1982). *The art and science of negotiations*. Cambridge: Harvard University Press.
24. Romero, C. (1991). *Handbook of critical issues in goal programming*. Elmsford: Pergamon Press.
25. Rosenschein, J. S., & Zlotkin, G. (1994). *Rules of encounter: designing conventions for automated negotiation among computers*. Cambridge: MIT Press.
26. Schniederjans, M. J. (1995). *Goal programming methodology and applications*. Dordrecht: Kluwer Academic.
27. Shachnai, H., Shmueli, O., & Sayegh, R. (2004). Approximation schemes for deal splitting and covering integer programs with multiplicity constraints. In *ALGO 2004 (WAOA)*, Norway, September 2004.
28. Shmueli, O., Golany, B., Sayegh, R., Shachnai, H., Perry, M., Gradovitch, N., & Yehezkel, B. (2004). *Negotiation platform*. International patent application WO 02077759, 2001-2, US patent application 20040133526.
29. Shmueli, O. (2001). Architectures for internal web services deployment. In *VLDB 2001* (pp. 641–644), Rome, Italy, September 2001.
30. Teich, J. E., & Wallenius, H. (1994). Advances in negotiation sciences. *Transactions in Operational Research,*, *6*, 55–94.
31. Teich, J. E., Wallenius, H., Wallenius, J., & Koppius, O. R. (2004). Emerging multiple issue e-auctions. *European Journal of Operational Research*, *159*, 1–16.
32. UDDI technical white paper. (2000). http://www.uddi.org/whitepapers.html.
33. UDDI programmer's API specification. (2000). http://www.uddi.org/specification.html.
34. Web Ontology Language (OWL05) (2005). *W3c*. http://www.w3.org/2004/OWL/.
35. Vickrey, W. (1961). Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, *16*, 8–37.